

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Брестский государственный технический университет»
Кафедра интеллектуальных информационных технологий

Учреждение образования
Белорусский государственный университет информатики и радиоэлектроники
Кафедра электронно-вычислительных машин

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, Р.Х. Садыхов

ОСНОВЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Методическое пособие по дисциплинам
«Основы искусственного интеллекта»
«Автоматизация проектирования вычислительных машин и систем»
«Архитектура вычислительных машин и систем»

*Рекомендовано Советом Брестского государственного
технического университета для студентов специальностей:*

1-40 03 01 «Искусственный интеллект»

1-40 02 01 «Вычислительные машины, системы и сети»

1-53 01 02 «Автоматизированные системы обработки информации»

Брест 2013

УДК 004.8 (07)
ББК 32 813
М 33

Рецензенты:

заведующий кафедрой интеллектуальных информационных технологий,
доктор технических наук, профессор **В.В. Голенков**;

доцент кафедры «Экономика и управление научными исследованиями,
проектированием и производством», кандидат экономических наук **П.В. Мелюшин**.

М 33 Головки В.А., Дудкин А.А., Матюшков Л.П., Садыхов Р.Х.
Основы вычислительных систем: методическое пособие. – Брест: Издательство
УО «БрГТУ», 2013. – 148 с.

ISBN 978-985-493-241-5

В данном методическом пособии рассмотрены основные 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 и специалистам, интересующимся проблемами современных информационных технологий в построении вычислительных машин, систем и сетей и их использования в различных приложениях.

УДК 004.8 (07)
ББК 32 813

ISBN 978-985-493-241-5

© Коллектив авторов, 2013
© Издательство БрГТУ, 2013

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
Глава 1. АЛГОРИТМЫ И ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ	5
1.1. Автоматы и алгоритмы	5
1.2. Средства описания алгоритмов	12
1.3. Методы реализации алгоритмов	25
1.4. Ассоциативные исчисления и нормальный алгоритм Маркова	33
1.5. Конечные автоматы	37
1.6. Архитектура вычислительных систем	47
Контрольные вопросы	49
Глава 2. ЭЛЕМЕНТЫ ТЕОРИИ ФОРМАЛЬНЫХ ЯЗЫКОВ	50
2.1. Синтаксис и семантика формальных языков	50
2.2. Алгоритмические языки	55
2.3. Операционные системы	58
2.4. Использование формальных языков для поиска информации	60
2.5. Базы данных и знаний	64
2.6. Языки в проектировании вычислительных систем	69
Контрольные вопросы	79
Глава 3. МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОЕКТИРОВАНИИ	80
3.1. Проблемы создания искусственного интеллекта	80
3.2. Диалоговые методы решения задач и экспертные системы	85
3.3. Распознавание образов и обработка изображений	90
3.4. Нейронные сети	101
Контрольные вопросы	106
Глава 4. ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ	107
4.1. Основные понятия интеллектуальной информационной технологии	107
4.2. Нейронные системы и нейрокомпьютеры	112
4.3. Системы технического зрения	123
4.4. Мультиагентные системы	129
4.5. Защита информации	131
Контрольные вопросы	141
ЗАКЛЮЧЕНИЕ	142
ЛИТЕРАТУРА	143
ГЛОССАРИЙ	145

ВВЕДЕНИЕ

Современные информационные технологии проникли во все области деятельности человека. С каждым годом наблюдается рост «интеллектуализации» различных автоматизированных систем в технике, управлении и быту. Поэтому крайне необходимо для специалистов всех профилей владеть общими подходами в использовании современных информационных технологий в конкретных областях деятельности.

Знание общих закономерностей в создании и адаптации различных автоматизированных систем крайне необходимо на современном этапе развития общества всем специалистам, ориентирующимся на использование информатики в своей практической деятельности. Владение общими подходами и закономерностями в проектировании информационных систем также позволит повысить их эффективность за счет квалифицированного выбора при приобретении готовых систем или заказе индивидуальных проектов.

В данном пособии на доступном уровне для специальностей всех профилей излагаются сведения об основных элементах и подходах, применяемых в проектировании, конструировании и эксплуатации современных интеллектуальных технологий и систем. Тому ярким 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 и т.д. Особую роль 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. Задача данного пособия также состоит в том, чтобы в краткой и доступной форме ознакомить потребителей с основами этих знаний. Их составляют теория алгоритмов и автоматов, формальные языки, базы данных и знаний, операционные системы и системы управления базами данных, нейронные сети и нейро-ЭВМ; сети ЭВМ и современные интеллектуальные технологии. Цель книги – ознакомить читателя с основными идеями и положениями этих теорий.

Материал пособия подготовлен группой авторов: Головкин В.А. – параграфы 3.4, 4.1, 4.2 (совместно с А.А. Дудкиным), заключение; Дудкин А.А. – 1.2, 1.6, 2.6, 4.3; 1.3 и 1.5 (совместно с Л.П. Матюшковым); Матюшков Л.П. – 1.1, 1.4, 1.5, 2.1-2.5, 3.2, 4.5; Садыхов Р.Х. – введение, параграфы 4.4, 2.3 (совместно с А.А. Дудкиным), параграф 3.1 (совместно с Головкин В.А. и Матюшковым Л.П.).

Глава 1. АЛГОРИТМЫ И ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ

1.1. Автоматы и алгоритмы

Общие понятия о цифровом автомате и алгоритме. 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) цифрового 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 строгом 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 однозначной 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 машин; 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 обозначить, но что значит «рассмотреть» бесконечную последовательность?

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

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

Интуитивное определение алгоритма. Существует несколько определений термина «алгоритм». Вообще под алгоритмом часто понимается некоторое формальное предписание, действуя согласно которому, можно получить нужное решение задачи. Формулировка соответствует 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. Процесс вычислений останавливается. Если нет, то ПКУ4.
4. Если первое обозреваемое число меньше второго, то переставь их местами. ПКУ3.
5. Вычитай второе число из первого и обозревай два числа: вычитаемое и остаток. ПКУ2 (процесс повторяется по вышеназванной схеме).

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

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

5. $15 - 5 = 10$. ПКУ2.

2. $5 < 10$. ПКУ3.

3. ПКУ4.

4. 10, 5. ПКУ5.

5. $10 - 5 = 5$. ПКУ2.

2. $5 = 5$. ПКУ3.

3. Остановка процесса, результат 5.

По определению акад. А.Н. Колмогорова, алгоритм или алгоритм – это всякая система вычислений, 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.

Решим задачу: начав поиск с площадки A , выяснить, достижима ли произвольная площадка Y .

Если она достижима, то найти путь из A в Y , а в ином случае после поиска вернуть-

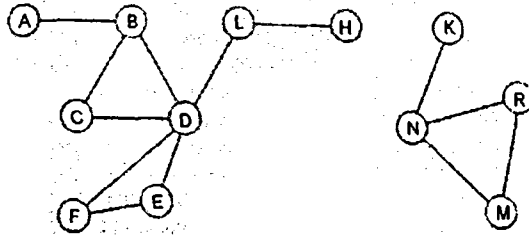


Рисунок 1.1.1 – Модель лабиринта

ся в А. Устройство лабиринта заранее неизвестно. Под решением задачи понимается указание общего метода поиска для любого расположения площадок А и У. Рассмотрим один из возможных методов поиска. Предположим, что в начале и конце проходимых коридоров можно делать любые виды пометок, по которым легко узнать, проходил ли данный коридор и сколько раз. Коридоры, которые не проходились ни разу, будут без пометок; пройденным один раз будем присваивать пометку 1, а пройденным дважды – пометку 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 пометкой 1, заменив пометку коридора знаком 2. Находясь на какой-либо площадке, можно различать следующие признаки:

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

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

Таблица 1.1.1 – Метод поиска пути из А в У

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

Относительно рассматриваемого метода справедливы следующие утверждения:

1. При любом расположении А и У за конечным числом ходов наступит остановка либо в А, либо в У.
2. Если остановка на площадке А, то площадка У недостижима.

Рассмотрим два небольших примера (рис. 1.1.2).

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.

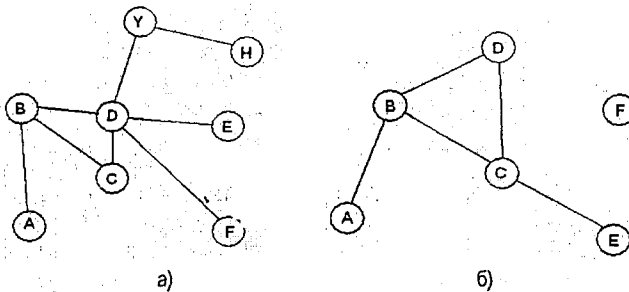


Рисунок 1.1.2 – Лабиринты: а) с достижимыми площадками; б) с недостижимыми площадками

Таблица 1.1.2 – Поиск достижимой площадки F

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

Таблица 1.1.3 – Недостижимость площадки F

Номер хода	Признак	Направление	Пометка коридора
		движения	
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. Напр., в алгоритме Евклида операции, сделанные разными вычислителями, совпадают во всех деталях. Здесь же от одной площадки отходит несколько коридоров, и выбор можно делать произвольно. Алгоритмом обычно называют строго детерминированную систему. Поэтому дополним алгоритм требованием выбирать первый коридор по часовой стрелке при возникновении неоднозначного продолжения.

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

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

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

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

Направленность алгоритма. Если способ получения последующей величины 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 – рекурсивные функции – является исторически первой формализацией понятия алгоритма.

Второй тип основан на представлении алгоритма как некоторого детерминированного устройства, 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 дискретном времени так, что в каждый следующий момент времени структура данных получается по заданным правилам по структуре данных величин, имевшихся в предыдущий момент. Структура элементарных шагов в определении алгоритма не оговаривается, поэтому, правила преобразования величин могут допускать или предписывать параллельную их обработку. Алгоритм может быть параллельным, когда некоторые шаги обработки могут выполняться 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, выполняют разные функции, 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 ещё 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 коммуникационных сетей. Масштабируемость зависит также от параллельного алгоритма: алгоритм, проверенный и хорошо работающий на вычислительной системе 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.1.1.3). Вначале нужно вычислить произведения $a \cdot b$ и $c \cdot d$, затем извлечь корень, и, наконец,

выполнить сложение. Описанный алгоритм можно изобразить следующим образом:

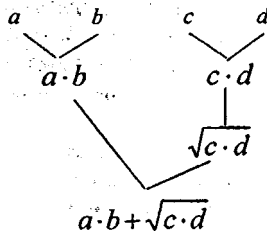


Рисунок 1.1.3 – Схема параллельного алгоритма

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 вычислительной системе 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. Средства описания 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 вычислений одного из них обеспечивало возможность реализации следующей 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 аргументов, точность вычислений.

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

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

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

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

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

- 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 минимизации 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 операторы 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 перейти к 4, иначе к 6.
4. $B := B + 1$.
5. Перейти к 7.
6. $M := M + 1$.
7. Если $i < n$, то перейти к 8, иначе к 10.
8. $i := i + 1$.
9. Перейти к 3.
10. Конец.

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

Эта последовательность операций и представляет собой запись алгоритма для человека, знающего алгебру и правила 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. Суть 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 заносится 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 подчеркнуть 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 .

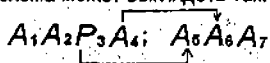
Мы остановимся более детально на операторной форме представления алгоритмов и их схем, так как предварительная подготовка задачи для программирования крайне редко обходится без укрупненной разбивки на операторы и описания связей между ними на языке схем. По мере усложнения задач и характера вычислительного 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 в одну строку и руководствуются следующими правилами:

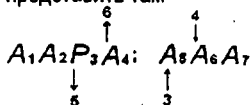
1. Порядковый номер оператора в данной схеме изображается нижним индексом оператора. Нумерация операторов сквозная.
2. Если оператор зависит от параметра, то этот параметр изображается верхним индексом при его буквенном обозначении.
3. Если знаки двух операторов располагаются в схеме рядом, то оператор, 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:



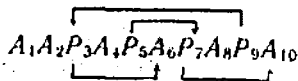
Для удобства записи горизонтальную часть стрелки можно опускать. При этом начало стрелки отмечается малой вертикальной стрелкой, направленной от оператора и снабженной номером оператора, которому передается управление с указанием номера, передающего управление оператору:

Тогда эту же схему можно представить так:



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 приведенными в этом параграфе алгоритмы в операторной форме можно записать так:



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

С помощью операторной схемы алгоритма составляется схема решения задачи, которая дополняется операторами, свойственными машинному вычислительному процессу. Для каждого оператора в порядке его записи затем составляется своя частная программа. Совокупность этих частных программ и дает программу решения задачи. Использование операторной схемы алгоритма 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 перерыва или поручать продолжение начатой работы другому лицу.

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

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

Схема алгоритма, с одной стороны, служит как формальный язык для записи содержания и логических связей задачи, а с другой – она сравнительно легко преобразуется в программу для ЭВМ.

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

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

Поясним понятие схемы алгоритма. Пусть требуется вычислить функцию

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

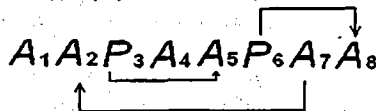
для всех дискретных значений x с шагом 0,1 из сегмента $[-7; 5]$.

Так как y представлен через аналитические выражения x^2 и $x^2 + \sqrt{x}$, по которым могут быть произведены вычисления в результате подстановки в них значений x , то алгоритм для вычисления 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. 1.1.4).

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 $\{\bar{e}_k = e_{i,j} = (v_i, v_j)\} \in E$, где $k = 1, \dots, M, i = 1, \dots, N, j = 1, \dots, N, i \neq j$, 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.

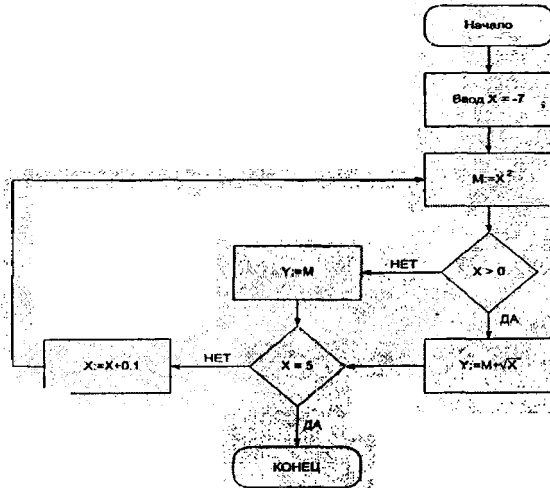


Рисунок 1.1.4 – Схема алгоритма

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).

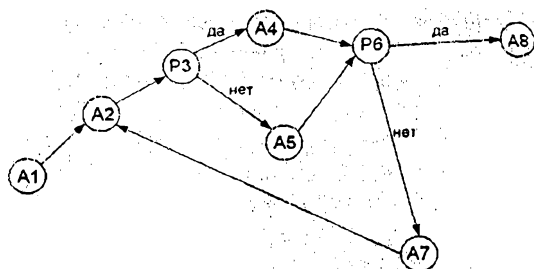


Рисунок 1.1.5 – Граф-схема алгоритма

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 исходных 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 алгоритм 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 в зависимости от типа обрабатываемого объекта. Система обработки способна выполнять множество операций обработки $O = \{o_1, \dots, o_k\}$, $k > m$. Сценарий обработки отдельного типа кадров содержит подмножество операций $O_j = \{o_{j_1}, \dots, o_{j_k}\}$, $\bigcup_{j=1}^n O_j = O$. Операции в каждом сценарии упорядочены отношениями следования $o_{ja} \succ o_{jb}$, т. е. для кадра с типом j операция a выполняется перед операцией b . Каждая операция обработки o_i характеризуется сложностью вычислений w_{o_i} , 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 $\{e_{i,j} = (v_i, v_j)\} \in E$, где $i = 1, \dots, N, j = 1, \dots, N, i \neq j$.

Дуга представляет собой отношение предшествования между операциями алгоритма и определяет передачу 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 формат данных, поэтому для всех сценариев и типов кадров дуги имеют равную стоимость.

Построение приложения для параллельной обработки потока данных состоит из нескольких шагов: 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 $((O), (P)) \rightarrow \bigcup_{PEP} (O_p)$, где

$\forall O_k, O_p : O_k \neq O_p \Rightarrow O_k \cap O_p = \emptyset$. Каждое подмножество операций O_p размещается на выбранном узле кластера.

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

$$T_{opt} = \min\{\max T_N\}, \quad (1.1)$$

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

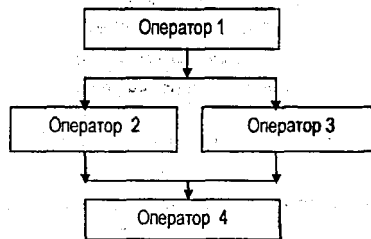


Рисунок 1.1.6 – Фрагмент параллельной ГСА

Сети Петри. Среди многих методов описания и анализа дискретных параллельных систем выделяют подход, который основан на использовании сетевых моделей, относящихся к сетям специального вида, предложенных Карлом Петри для моделирования асинхронных информационных потоков в системах преобразования данных. Сети Петри обеспечивают описание как параллельных алгоритмов, так и собственно вычислительных систем и их устройств.

Структура сети представляется ориентированным двудольным графом. Множество V вершин графа разбивается на два подмножества T и P , $V = T \cup P$, $T \cap P = \emptyset$. Дугами могут связываться вершины из множеств P и T . Динамика развития процессов отражается в вершинах P метками (марками). Распределение меток по вершинам P называют маркированием 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 инцидентно какому-либо переходу ($T \times P \rightarrow \{0, 1\}$ – функция следования), каждый переход инцидентен какому-либо месту ($P \times T \rightarrow \{0, 1\}$ – функция предше-

ствования), т.е. она задает множества дуг $\{t_j, p_i\}$ и $\{p_i, t_j\}$; $M_0: P \rightarrow N_0$ – начальное маркирование (состояние) сети; N_0 – множество положительных целых чисел.

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

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

Маркирование сети представляется вектором $M = \{m(p_i)\}$, где $m(p_i)$ – число меток в позиции p_i . Переход t_j возбужден при маркировании M и может сработать, если выполняется условие $\forall p_i \in Pre(t_j) [m(p_i) - O(p_i, t_j) \geq 0]$, то есть 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 $p_i \in Pre(t_j)$ теряет $O(p_i, t_j)$ меток, а каждая из позиций $Post(t_j)$ получает $J(t_j, p_i)$ меток. Каждое изменение маркировки называют *событием*, причем каждое событие связано с определенным переходом. Считается, что события происходят мгновенно и одновременно при выполнении некоторых условий.

Каждому условию в сети Петри соответствует определенная позиция. Совершенно событию соответствует *срабатывание* (возбуждение или запуск) перехода, при котором маркеры из входных позиций этого перехода перемещаются в выходные позиции. Последовательность событий образует моделируемый процесс. Изображение позиции и перехода на графе показано на рисунке 1.1.7.

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).

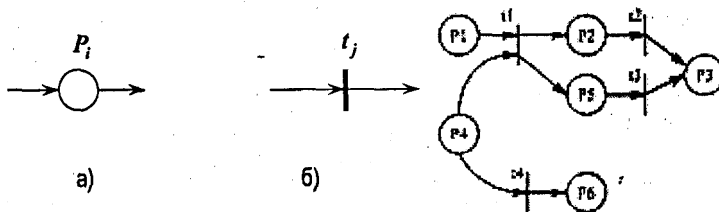


Рисунок 1.1.7 - Графическое изображение сети Петри: а) – позиции; б) – перехода

Правила срабатывания переходов конкретизируют 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 выходной позиции увеличивается на M_j , где M_j – число дуг, связывающих переход с j -й позицией. Пример смены разметки показан на рис. 1.1.9. Процесс срабатывания переходов и смены разметки называют работой сети Петри.



Рисунок 1.1.9 – Смена разметки сети Петри: а) до срабатывания перехода t_j ; б) после срабатывания перехода t_j .

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 ветвей алгоритма, при выполнении параллельной части алгоритма управления. Множество переходов 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 последовательностей срабатывающих переходов.

В качестве примера сети Петри рассмотрим вычислительную систему (ВС) (рис. 1.1.10), содержащую процессор (П) и устройство связи с объектом (УСО), которые могут работать параллельно, и запоминающее устройство (ЗУ). Процессы в данной ВС порождаются периодическими и инициативными запросами от ОУ и инициативными запросами от оператора. Рассмотрим очередь запросов оператора, которые включают выполнение на П двух операций. Первая из них использует текущие данные из ЗУ. Вторая операция выполняется после первой и после завершения работы УСО с ОУ и ЗУ.

Сеть Петри, описывающая данный процесс, приведена на рис. 1.1.11.

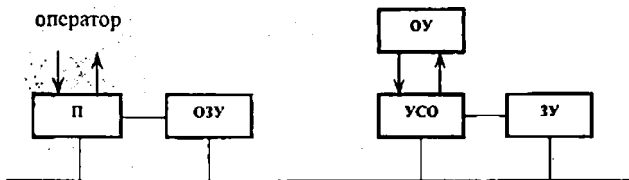


Рисунок 1.1.10 – Однопроцессорная вычислительная система

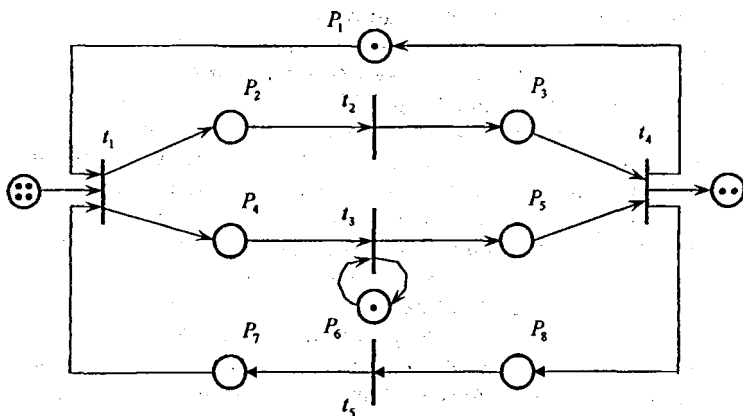


Рисунок 1.1.11 – Модель процесса в виде сети Петри

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

Позиции: P_1 – П свободен; P_2, P_4 – ввод данных с ЗУ в ОЗУ 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 ЗУ. Возможность параллельной работы процессора и УСО отражена параллельными ветвями с 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.

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) их 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 цифрой в кружке, а связи между операторами – цифрами слева и справа от этого кружка. В левой части прямоугольника-оператора будем записывать номера предшест-

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

Наиболее распространенными подклассами графов, применяемых в спецификациях ПО, являются деревья, конечно – автоматные диаграммы, обобщенные диаграммы переходов, сети 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 эффективного 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 тестирования, анализа производительности ПО, а также статическом контроле структуры программы. Для перечисленных целей в математической модели программы достаточно задать порядок следования действий, определяемый 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 x , графа взвешивается числом p_x , указывающим на длину (число команд, операций) участка.

Анализ графовой модели на соответствие формальным правилам построения позволяет без исполнения программы выявить такие ошибки в ее структуре, как наличие вершин, не достижимых из начальной ни по одному из путей ("лишние" вершины), а также вершин, из которых не достижима конечная вершина ("тупиковые" 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.

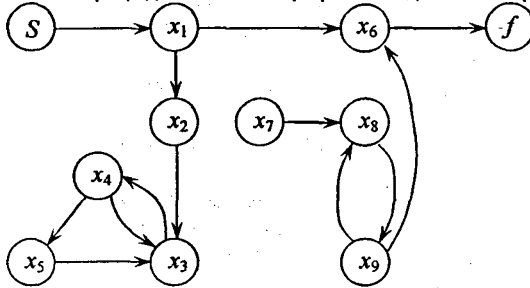


Рисунок 1.1.12 – Графовая модель программы

Вершины x_1 , x_f соответствуют формальному началу и концу программы. При определении структурных ошибок по матрице R проверяются следующие соотношения:

$$\sum_{j=1}^n r_{sj} = n-1; \quad \sum_{j=1}^n r_{jf} = n-1. \quad (1.2)$$

Соотношения (1.2) указывают на отсутствие лишних и тупиковых компонент в графовой модели программы. При невыполнении соотношений локализация ошибочных компонент выполняется проверкой условий: $r_{sj} = 0$ и $r_{jf} = 0$.

Анализ матрицы достижимости графа программы (рис. 1.1.12) показывает на наличие "лишних" x_7 , x_8 , x_9 ($r_{s7} = r_{s8} = r_{s9} = 0$) и "тупиковых" компонент x_2 , x_3 , x_4 , x_5 ($r_{2f} = r_{3f} = r_{4f} = r_{5f} = 0$). Графовая модель программы может использоваться для определения ее статистических характеристик и показателей надежности.

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, определяющих 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 аппроксимации непрерывных методов (моделей) вычислений дискретными операциями. Погрешности представления данных и накопления 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 погрешности 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 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 $(j + 1)$ 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) $y_1 = 0,5 (2 + 2:2) \approx 1,5$, $|1,5 - 2| > 0,01$

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

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

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

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

Ответ: $y = 1,41$.

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

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

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

Ответ: $y = 1,41$.

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

1) необходимость сведения ее к выполнению последовательности арифметических действий (практически можно использовать и другие элементарные действия, для которых уже 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 с 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 приведет к связности алгоритма, равной $(n+1)$.

Но эти вычисления можно упростить по следующим формулам:

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

После выполнения всех вычислений получим результат $y = y_n$.

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, функций 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 бесконечного числа операций, на практике сводятся к способам с конечным числом операций. Например, задание функции в виде суммы членов сходящегося степенного ряда можно отнести к первому случаю, т. к. в памяти машины хранится конечное число первых членов ряда и отбрасываются остальные. Точнее, программа вычисления функции с заданной точностью реализована на базе необходимого числа членов ряда.

Выбор способа задания функций и соответственно вычисления ее значений базируются 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 ($|x|$, x^n , $\ln x$, $\sin x$ 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 заменяются 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 подбора соответствующего многочлена.

Степень многочлена подбирается так, чтобы выполнялись условия:

- 1) степень должна быть возможно более низкой;
- 2) значение многочлена должно отличаться от соответствующего значения функции на величину, меньшую, чем некоторое малое число E , характеризующее допустимую погрешность получаемых значений функции.

Если $P_i(x)$ – многочлен, то на соответствующем подинтервале должно выполняться неравенство $|\tilde{f}(x) - P_i(x)| < E$.

В этом случае в память вместо большой таблицы значений функции или большой по объему 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 вычислительных систем и их компонентов на 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, то есть более сложные, чем NP -полные.

Все задачи решаются методами комбинаторного поиска, основанными на построении дерева поиска. Эти методы порождают ветвящиеся процессы последовательного конструирования искомого решения, реализуя многоступенчатые переходы от начальной ситуации, отражающей 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 или множество конечных вершин.

Разработка методов решения конкретных задач связывается с конкретизацией понятий "ситуация" и "элементарный шаг", с определением способов последовательного конструирования решений, с введением некоторых оптимизирующих правил. Общая схема процесса решения задач может быть представлена деревом поиска следующим образом. Корню дерева поиска ставится в соответствие исходная (начальная) ситуация, а остальным вершинам – ситуации, которые могут быть достигнуты в процессе поиска. Пути соответствуют шагам решающего процесса и связывают вершины, соответствующие

щие 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 (любой 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 дерева поиска дает оптимальное решение. Затем формируются некоторые усеченные деревья поиска, точнее описываются правила их построения, обход которых дает 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 априорно, 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 входов 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 $Q = 0$ 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 из Q со строками из R . Для начальной ситуации матрица S не содержит ни одной строки. Мощность 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. Ему будет соответствовать строка $r \in R$, для которой число совместимых с ней миноров из Q будет 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 $R = 0$) $|Q| < q$, то Q запоминается в качестве текущего решения и $q = |Q|$.

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

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

Описанный алгоритм является точным, так как в каждой ситуации рассматривается полная 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 разбиение заданного свойства.

Правило расщепления 2. В матрице S находим минимальный по числу единиц строку. Ему 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, i -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 разрешимости решаемых задач такого рода достигается разработкой приближенных алгоритмов, которые не гарантируют получения минимального группирования, но 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 будет цепным (без возврата) – основанным лишь на редуцировании текущей ситуации. Очевидно, наиболее приемлемыми с точки зрения практики являются именно такие алгоритмы, или алгоритмы с сильно ограниченным перебором. При их построении основное внимание уделяется нахождению эвристических правил построения пути из корня дерева поиска в его концевую 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. Неразрешимой является проблема выводимости для исчисления предикатов, 10-я 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 той или иной серии задач, стало возможным благодаря определению алгоритма через точное математическое описание класса вычислимых функций натурального аргумента (частично рекурсивные и рекурсивные функции) и благодаря уточнению математического определения класса вычислительных процессов, реализуемых абстрактными вычислителями (машины 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 отдельного случая оценки существует определенный набор данных, который называется входными данными. Размер входных данных определяется объемом памяти необходимым для размещения этих данных. Для геометрических задач, это – обычно число точек и координат фигуры или сторон во входном наборе. 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, то есть 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 наихудшего случая из всех допустимых вариантов входного набора размера n . Точно также вводится функция сложности по $SA(N)$ для каждого алгоритма. Сложность для худшего случая это пессимистическая оценка сложности алгоритма. В алгоритмическом анализе сложность по времени и по объему памяти выражается как функции от размерности входного набора, умноженная на постоянный коэффициент. $O(f(N))$ обозначает 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 $N \geq N_0$.

Мы говорим, что алгоритм A имеет полиномиальную сложность P по времени, если $TA(n) = O(n^c)$, где c – некоторая постоянная.

Полиномиально разрешимые задачи можно успешно решать на компьютере и даже в тех случаях, когда они имеют большую размерность. Для других задач не удается найти полиномиальный алгоритм. поэтому их называют трудноразрешимыми. К классу трудноразрешимых задач относится 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 $O(2^N)$.

Алгоритмы, которые имеют полиномиальную сложность по времени, являются эффективными. Класс задач, который не допускает поиска эффективных алгоритмов, известен как класс NP -полных задач. Но, может быть, возможно, найти приближенное к оптимальному решение 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 (приближения).

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 нами при решении задачи поиска пути в лабиринте. На абстрактном уровне вычислительная машина всегда выполняет операции по переработке исходного текста (набор букв и чисел, задающих условие задачи) в некоторый заключительный текст (описание результата решения задачи). В машине буквы и цифры задаются в виде 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 этом алфавите. Например, в алфавите $A = \{a, b, c\}$ словами будут последовательности $ab, ac, abac, bbbb$ и т.п.

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

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

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

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

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

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

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

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

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

Пример.

$\{a, b, c, d, e\}$ – алфавит,

$ac - ca; abac - abacc;$

$ad - da; eca - ae;$

$bc - cb; eda - be;$

$bd - db; edb - be.$

} – подстановки

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

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

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

Иногда рассматривается специальный вид ассоциативного исчисления, которое задается алфавитом и системой ориентированных подстановок типа $H \rightarrow M$. 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 – слово).

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.

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

$n = 2^{a_1} \cdot 3^{a_2} \cdot 5^{a_3} \cdot \dots \cdot P_{m-1}^{a_{m-1}}$, где $P_0 = 2, P_1 = 3, \dots, P_{m-1}$, т.е. P_m — m -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 $n = 60$, имеем $60 = 2^2 \cdot 3^1 \cdot 5^1$ т.е. $a_1 = 2; a_2 = 1; a_3 = 1$.

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

С помощью этого способа (геделизации) можно нумеровать 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 q , 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 обычным способом геделизацию, 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. Конечные автоматы

Понятие автомата. Типы автоматов. Автомат — это алгоритм, определяющий некоторое множество 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 цепочка заданному множеству L ;

2) преобразователи — автоматы с выходом, которые преобразуют входную цепочку x в цепочку 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. Лента автомата бесконечна, но занята 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.

Слово P : $11 + 11 + 111$.

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

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

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

$$P_2 = + 1111 + 111 \quad P_7 = + 1111111$$

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

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

Мы видим, что этот алгоритм суммирует количество единиц. Следует обратить особое внимание на элементарность и однозначность шагов, выполняемых при реализации алгоритма. Естественно, что представления любого алгоритма в такой форме достаточно, чтобы поручить его реализацию некоторому автомату. Нормальный алгоритм Маркова можно рассматривать как стандартную форму для задания любого алгоритма. Такая форма на практике используется лишь в теоретических построениях. В процессе построения теории численных 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 запись условий A_n в запись решений B_m , можно свести к вычислению значений некоторой числовой 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. Теперь 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 ей номер n' , затем вычислить $m^* = \Phi(n')$ и по m^* определить запись решения.

X – входной алфавит;

Y – выходной алфавит;

δ – функция переходов, задающая отображение $K \times X \rightarrow K$; γ – функция выходов, задающая отображение $K \times X \rightarrow Y$.

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

Пусть на некотором такте t управляющее устройство находится в состоянии q , а из 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.:

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

Если же команда $qx \rightarrow py$ отсутствует, то автомат оказывается блокированным 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, то такой автомат называют инициальным, т.е. $q(0) = q_0$.

Инициальный автомат – это шестерка вида $A = (K, X, Y, \delta, \gamma, q_0)$, 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 $Q \times X \rightarrow Y$);

q_0 – начальное состояние.

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

Закон функционирования автомата Мили задается уравнениями:

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

Закон функционирования автомата Мура задается уравнениями:

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

Из сравнения законов функционирования видно, что, в отличие от автомата Мили, 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 дополнительно 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 или 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.к. 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

Входная головка в каждый момент времени читает (обозревает) одну ячейку входной ленты. За один такт работы автомата входная головка может сдвинуться на одну ячейку вправо 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) машина Тьюринга (МТ);
- 2) линейно-ограниченный автомат (ЛОА);
- 3) автомат с магазинной памятью (МП-автомат);
- 4) конечный автомат (КА).

Сложность рабочей ленты определяет сложность автомата. Так, например:

- 1) машина Тьюринга имеет 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;
- 4) у конечного автомата рабочая лента отсутствует.

Формальное определение автомата. Неинициальный автомат – это пятерка вида

$A = (K, X, Y, \delta, \gamma)$, где

K – множество состояний (алфавит состояний);

X – входной алфавит;
 Y – выходной алфавит;
 δ – функция переходов, задающая отображение $K \times X \rightarrow K$; γ – функция выходов, задающая отображение $K \times X \rightarrow Y$.

Функционирование автомата можно задать множеством команд вида $qx \rightarrow ru$, где $q, r \in K, x \in X, u \in Y$.

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

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

Если же команда $qx \rightarrow ru$ 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, то такой автомат называют инициальным, т.е. $q(0) = q_0$.

Инициальный автомат – это шестерка вида $A = (K, X, Y, \delta, \gamma, q_0)$, 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 $Q \times X \rightarrow Y$);

q_0 – начальное состояние.

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, 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 дополнительно 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 или 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.к. 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 Мура может быть задан как входонезависимый автомат Мили. Однако это не соответствует способу 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 Мили, у которого выход, соответствующий некоторому 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 значений входов X в данный момент времени и не зависят от входных воздействий в предшествующие моменты времени, называется комбинационным (однотактным), или автоматом без памяти. В структурных моделях автоматы этого класса называются комбинационными схемами (КС). Функционирование КС описывается математической моделью вида $Y = f(X)$. Функция, описывающая это соотношение для одного выхода, называется булевой функцией (БФ). БФ задаются с помощью полностью или не полностью определенных таблиц, которые называются таблицами истинности (ТИ) и таблицами решений соответственно. В табл. 1.5.1 представлены ТИ для ряда БФ двух переменных x_1 и x_2 . Более компактной формой представления является задание булевых функций в виде булевых формул, которые в отличие от таблиц всегда определены на всех 2^n (n – число входных переменных) наборах. Примеры также приведены в табл. 1.5.1

Таблица 1.5.1 – Логические функции

Комбинации аргументов x_1, x_2	Функция И ($y = x_1 \cdot x_2$)	Функция ИЛИ ($y = x_1 + x_2$)	Функция И-НЕ ($y = x_1 \cdot \bar{x}_2$)	Функция ИЛИ-НЕ ($y = x_1 + \bar{x}_2$)	Функция исключающее ИЛИ ($y = x_1 \oplus x_2$)
0 0	0	0	1	1	0
0 1	0	1	1	0	1
1 0	0	1	1	0	1
1 1	1	1	0	0	0

При представлении логической функции математическим выражением используют два вида ее представления.

Совершенной дизъюнктивной нормальной формой (СДНФ) называется логическая сумма элементарных логических произведений, в каждое из которых аргумент или его отрицание входят один раз. СДНФ может быть получена из таблицы истинности следующим образом: для каждого набора аргументов, на котором функция равна «1», записывают элементарные произведения переменных, причем переменные, значение которых равно нулю, записывают с инверсией. Полученные произведения, которые носят название конъюнкты единицы, суммируют. Далее по законам алгебры логики это выражение минимизируется с целью сокращения в нём как количества элементарных произведений, так и некоторых переменных в них, что упрощает техническую реализацию КС.

Совершенной конъюнктивной нормальной формой (СКНФ) называется логическое произведение элементарных сумм, в каждую из которых аргумент или его отрицание входят один 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 конъюнкт нуля, объединяют операцией логического умножения, полученная СКНФ далее по законам алгебры логики это выражение минимизируется.

Задание БФ в виде графа переходов нерационально, так как для этого класса 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 использование ЭВМ проявилось в автоматизации умственной деятельности человека, но вместе с миниатюризацией элементов ЭВМ появилась возможность сочетания автоматизации производственных процессов с одновременной автоматизацией управления ими непосредственно на объекте. С этого момента начались интенсивные исследования и эксперименты, ставящие своей целью создание машин, обладающих не только мускулами, но и интеллектом. Такие машины, выполняющие некоторые функции человека, стали называться роботами.

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

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

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

Робот должен обладать способностью распознавать обстановку и автоматически вырабатывать решения о своих дальнейших действиях для выполнения нужных технологических операций в неопределенной или меняющейся ситуации. Еще более интеллектуальным должен быть робот для ведения домашнего хозяйства и имитации сложных механических действий человека. Зарубежные специалисты ожидают его появления на рынке по относительно приемлемой цене на уровне стоимости легкового автомобиля. 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 их 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 отрезков 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 эти целочисленные значения. Моменты времени $0, 1, 2, \dots, i$ назовем тактами.

Текущий такт, соответствующий настоящему моменту времени, обозначим через i , а предшествующие 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, описываемых 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», т.е. 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^1 и C^0 можно отыскать C^1 , приняв $i = 1$. Далее по C^1 и P^2 отыскать C^2 и т.д. конечных множества символов $\{C\}$ и $\{P\}$ и рекуррентное соотношение $C = F(C^{i-1}, P^i)$, связывающее переменные C и P .

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

Фиксируя разные символы из множества $\{P\}$, получим g автономных автоматов. Задания g автономных автоматов достаточно, чтобы описать поведение конечного автомата. Автомат будем считать полностью заданным, если известны $\{P\}$, $\{C\}$ и $F(C^{i-1}, P_C)$. Функцию F можно задавать различными способами.

Чаще всего задание различных функций F осуществляется с помощью таблиц.

Пусть множество входов $\{P\}$ состоит из g символов. Тогда таблица функций переходов состояний строится следующим образом: вычерчивается прямоугольник, содержащий $(r+1) \cdot (k+1)$ клеток. В первую клетку в верхнем левом углу прямоугольника соответственно записываются в верхней и нижней частях ее наименования координат P_i, C^{i-1} . Вправо от нее по строке поклеточно 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 $C^{i-1} = C_m$. Для конкретизации характера таких функций будем задавать их таблицами.

В итоге получим таблицу, которую обычно называют основной таблицей конечного автомата (табл. 1.5.2).

Таблица 1.5.2 – Основная таблица конечного автомата

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

Размещение данных в таблице таково, что пользование ею не вызывает затруднений.

Пусть $P^i = P_2$, а $C^{i-1} = C_k$. Тогда искомое значение функции $F(C_k, P_2)$ отыскивается на 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 .

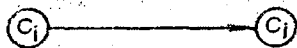


Рисунок 1.5.1 – Задание конечного автомата автономными автоматами

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

$C^0 \backslash P^i$	P_1	P_2
C_1	C_2	C_3
C_2	C_1	C_2
C_3	C_4	C_3
C_4	C_2	C_1

Чтобы различить, для какого фиксированного входа рассматривается 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 рисунке 1.5.2. Можно заметить, что от каждого из кружков на рисунке 1.5.3. отходит только по одной стрелке. Этот факт следует из определения детерминированного конечного автомата.

Для большей компактности таблицу 1.5.2. можно представить одним объединенным графом, который называется диаграммой состояний конечного автомата. Как видно из рисунка 1.5.3, на объединенном графе над каждой стрелкой записывается тот вход P , который вызывает переход из исходного состояния в то, к которому направлена стрелка. Диаграмма состояний автомата весьма наглядна. Допустим, необходимо установить, каким будет внутреннее состояние автомата A на третьем такте при входной последовательности $\{P\} = \{P_1, P_1, P_2\}$ и начальном состоянии $C^0 = C_3$. 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.

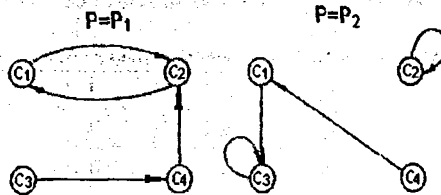


Рисунок 1.5.2 – Задание автомата диаграммой

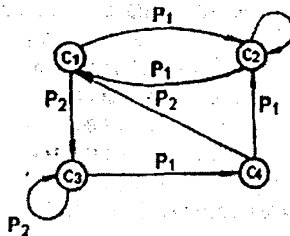


Рисунок 1.5.3 – Объединенный граф автомата

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 квадратную матрицу, состоящую из нулей и единиц. Пронумеруем все возможные состояния S цифрами 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 S должна отходить одна и только одна стрелка, и, естественно, в строке m матрицы $A(P_c)$ найдется один элемент $a_{mj}(P_c) = 1$, а остальные будут равны нулю.

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

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

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

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

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

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

з начальному состоянию $S^0 = S_3$ — матрица $A(P^0) = (0010)$. Тогда, используя формулу, получим искомым результат: $A_3(P^3) = (0100)$.

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

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

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

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

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

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

Пусть задан конечный автомат для управления роботом, предназначенным для обхода замкнутых контуров из отрезков прямых целочисленной длины, соединенных так, что они образуют внутренние углы замкнутого многоугольника в 90 или 270 градусов. Автомат служит управляющей системой для робота, который может двигаться за один такт на единичный шаг вперед, либо поворачиваться на 90 градусов по часовой стрелке, либо захватывать деталь, либо отпускать деталь, либо стоять. Иными словами, по одной команде «двигаться вперед» робот проходит единичный отрезок, а по команде «поворот» – поворачивается на девяносто градусов 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 для исполнительного механизма из-за разных состояний конечного автомата. Покажем, что конечный автомат А (задается конструктором), приведенный в нижеприведенной таблице, достаточен для решения поставленной задачи.

Таблица 1.5.3 – Управляющий автомат А

$C \backslash P$	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 детали, 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 выгрузить, а затем снова вернуться в точку $(X_0; Y_0)$. Ориентация робота в направлении возможного движения по прямой указана стрелкой на его кожухе.

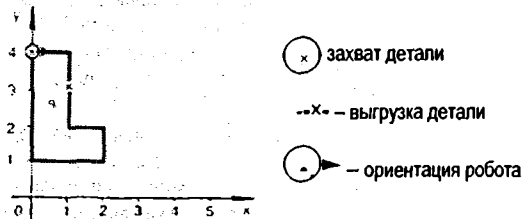


Рисунок 1.5.4 – Схема движения робота

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 которое шло входное воздействие символов P в виде цепочки смены состояний):

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

Чтение записи происходит на базе таблицы автомата A и рисунка контура. В таблице мы 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 контуру (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 покоя C_1 во второй последовательности выделено особо (над символом C_1 нет никакого входного 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 нами класса. Таким образом, на этом небольшом примере можно проиллюстрировать главное преимущество гибких автоматизированных производств с использованием роботов: для избранного класса задач меняются только управляющие программы, а оборудование остается тем же. Не составляет труда сделать и ряд других интерпретаций возможных функций робота, например: сварка деталей по контуру или какой-то линии, шлифовка заусенцев, покраска поверхностей, сверление отверстий в заданных точках и т.п.

При объяснении процесса движения робота по контуру нам пришлось прибегнуть 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 математической символики для записи различных формул и при доказательстве теорем. Эти символические записи могут читать люди или машины, которые в состоянии интерпретировать их смысл. Понятно, что последовательность символов не может быть любой, часто имеются ограничения на порядок записи символов в зависимости от порядка их следования. Таким образом, мы невольно пользуемся искусственными языками для точного описания различных процессов и объектов. Языки такого типа используются и при описании алгоритмов. Остановимся ниже на особенностях их построения и использования.

1.6. Архитектура вычислительных систем

Классификация вычислительных систем. Одним из наиболее распространенных способов классификации ЭВМ является систематика Флинна (Flynn), в рамках которой основное внимание при анализе архитектуры вычислительных систем 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 обычные 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 нет единого мнения – ряд специалистов говорят, что примеров конкретных ЭВМ, соответствующих данному типу вычислительных систем, не существует, и введение подобного класса предпринимается для полноты системы классификации; другие же относят к данному типу, например, систолические вычислительные системы 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 вычислительных систем.

Принципы построения параллельных вычислительных систем. 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 вычислительной системы – к устройствам ввода/вывода, к обрабатывающим процессорам и к устройствам памяти;

– избыточность элементов вычислительной системы – организация избыточности может осуществляться в следующих формах:

– использование специализированных устройств, таких, как отдельные процессоры для целочисленной и вещественной арифметики, устройства многоуровневой памяти (регистры, кэш);

– дублирование устройств ЭВМ путем использования нескольких однотипных обрабатывающих процессоров или нескольких устройств оперативной памяти.

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 приводит к существенным временным задержкам и как результат, эффективная обработка данных при данном способе организации вычислений возможна только для параллельных алгоритмов с низкой интенсивностью потоков межпроцессорных передач данных.

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

Примером параллельных вычислительных систем 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 (операционного) устройства и устройства управления.

Любое действие, выполняемое в операционном блоке, описывается некоторой микропрограммой и реализуется за один или несколько тактов. Элементарная функциональная операция, 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 это 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,

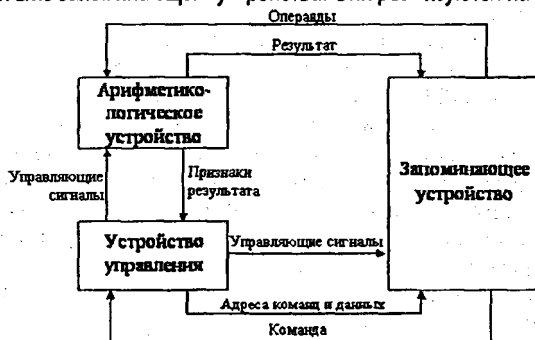


Рис. 1.6.1 – Структура классической ЭВМ

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 соответствие микропрограмма.

Литература

[2, 9, 13, 21-23, 29, 33, 34, 36, 37]

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 доказательства 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. Назовите основные типы архитектур вычислительных систем и их особенности.

Глава 2. ЭЛЕМЕНТЫ ТЕОРИИ ФОРМАЛЬНЫХ ЯЗЫКОВ

2.1. Синтаксис и семантика формальных языков

Формальный и естественный языки. Интерес к применению обычного разговорно-языка в диалоговом общении человека с ЭВМ и к подготовке программ объясняется целым рядом причин. Исходя из практики общения между людьми, пользователи ЭВМ, конечно, хотели бы говорить с ней на обычном языке: подавать устные команды голосом, давать читать 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 совпадения этих языков с естественным не удается достигнуть, и поэтому языкам приходится обучаться.

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 простейших средств такого типа является управление работой программ на уровне «меню», т.е. машина дает рецепты человеку, а он выбирает нужный из них для своего случая. Но все же есть 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 последующей 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 (их смысл). Форме конкретного предложения не всегда соответствует единственное содержание. Например, фраза «он встретил ее на поляне с цветами» не является определенной (она была 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 их 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 без противоречия ни истинным, ни ложным).

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

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, в дальнейшем, чтобы избежать путаницы, мы будем четко различать метаязыки (внешние языки) и язык-объект.

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

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

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

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

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

конечного числа предложений, называемых металингвистическими формулами Бэкуса. При их построении используются два универсальных метасимвола: «:: = » и «|»; первый можно читать как «по определению есть», второй – как «или». Остальные метасимволы выбираются произвольно разработчиком формального языка, который является также и разработчиком, внешнего языка.

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

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

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

<ненулевая цифра> :: = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

<цифра> :: = 0 | <ненулевая цифра>

<четная цифра> :: = 0 | 2 | 4 | 6 | 8 |

<начало> :: = <ненулевая цифра> |

<начало> <цифра>

<четное число> :: = <четная цифра> | <начало> <четная цифра>

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

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

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

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

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

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

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

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

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

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

Синтаксические правила описывают связи между частями предложения. Применение правила $\phi \rightarrow \Phi$ 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. Языки, порождаемые такими грамматиками, называются бесконтекстными языками.

В грамматике непосредственно составляющих (НС – грамматике) или контекстной грамматике синтаксические правила имеют вид $vAv \rightarrow v\Phi v$. В НС – грамматике каждый шаг вывода состоит в замене одного вхождения символа A вхождением цепочки Φ , причем замена обусловлена наличием контекстов V и W . В НС – грамматиках на каждом шаге вывода заменяется только один символ, поэтому в них 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: $I \rightarrow AAB$; $A \rightarrow a$; $B \rightarrow C$; $C \rightarrow c$ (основные символы – a, b, c ; A, B, C – вспомогательные символы, I – начальный символ), то 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.

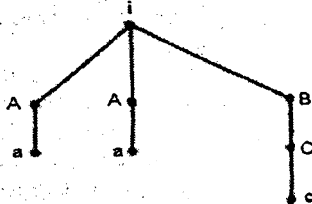


Рисунок 2.1.1 – Дерево вывода

Иногда в определение грамматики еще включают алфавит связей между буквами или словами, а также выделяют операции и формулы, которые используются в построении цепочки, 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 пишет цифры или другие знаки; эти знаки можно рассматривать как элементы, которые он порождает 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 функционирования рассматриваемой системы (или машины). Процессы вывода в порождающих грамматиках аналогичны преобразованиям слов в ассоциативных исчислениях.

2.2. Алгоритмические языки

Общие подходы к построению алгоритмических языков и трансляторов. Алгоритмический язык является средством точного формулирования вычислительных процессов для их последующей реализации на вычислительных машинах.

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

- 1) алгоритмы численного анализа (расчеты по формулам и т.п.);
- 2) процессы обработки данных в экономических расчетах (обработка таблиц);
- 3) переработка символьной информации (обработка текстов, аналитические выкладки, моделирование интеллекта и т.п.).

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

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

- 1) отбора изобразительных средств языка;
- 2) описания языка;
- 3) разработки транслятора.

Охарактеризуем кратко каждый из этапов.

Отбор изобразительных средств языка. 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 действия над ними, которые после их формализации и введения соответствующей символики объявляются элементарными объектами и базисными операциями языка.

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

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

Алгоритмы численного анализа находятся в лучшем положении, так как для них уже существовала сложившаяся веками математическая символика. Кроме того, применение 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 компонентов 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, а 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 априорного 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 или объекты могут описываться синтаксически допустимыми текстами языка (семантика).

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

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

- 1) набор основных символов (алфавит языка: буквы, цифры, разделители, знаки операций и т.д.);
- 2) набор имен или символов для обозначения «частей речи» языка;
- 3) набор математических формул, указывающих, как данная часть речи может быть сконструирована из других частей путем сочетания последних в некотором заданном порядке.

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

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

С помощью металингвистических формул можно давать определение лишь объектам, которые представляют собой конечные последовательности некоторых символов. Ошибки в программе или в элементах программ (в выражениях, операторах и пр.), обусловленные несоответствием программы или элементов программы металингвистиче-

жим формулам, называются синтаксическими ошибками. Программы или какие-либо элементы программы, не содержащие синтаксических ошибок, называются синтаксически правильными. Синтаксическая правильность программы не означает, что в программе вообще нет ошибок. Например, средствами синтаксического контроля нельзя обнаружить ошибку, допущенную автором, если синтаксически фраза не противоречит правилам ее написания. Это можно сравнить с известным примером, когда перестановка запятой на другое место решает судьбу человека: «Помиловать, нельзя повесить», хотя автору текста хотелось отдать другой приказ: «Помиловать нельзя, повесить».

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

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

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

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

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

Первый язык называется входным языком, а второй – выходным.

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

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

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

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

Процесс трансляции можно разбить на несколько составных частей: синтаксический анализ и контроль текста на входном языке (обнаружение конструкций, записанных с отклонениями от правил их записи); анализ описания данных и распределение памяти для *них и 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 сохранить возможность выполнения программ для ЭВМ старых моделей на новых ЭВМ.

2.3. Операционные 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 по 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 заданий, ведает такими ресурсами машины, как память, управляет процессами обмена с внешними устройствами.

Планирование распределения ресурсов осуществляется на основе анализа потока заданий. Информация о ресурсах для данной задачи описывается пользователем ЭВМ (примерный 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 – выводом, которая организует ввод – вывод при выполнении вычислительной системой всех возможных заданий. Диагностика сбоев и неисправностей машины сообщается оператору, и он принимает соответствующие меры по устранению неисправностей или же продолжает решение оставшейся части задач, которые не используют вышедшей из строя аппаратуры. Весь ход вычислительного процесса на ЭВМ протоколируется автоматически.

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

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

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

Операционные системы с каждым днем продолжают совершенствоваться, и степень их доступности для пользователей будет повышаться. Обобщение материалов по существующим ОС и перспективным разработкам ОС идет пока медленно. Это связано с тем, что большинство учебников по ОС тесно привязано к конкретным маркам машин, но постепенно общие закономерности начинают выкристаллизовываться и находить отражение в литературе по системному программированию. Следует отметить, что долгие годы работа по обеспечению «дружественного» интерфейса системными программами-профессионалами считалась черновой и неинтересной, так как она была направлена лишь на облегчение освоения ОС пользователем и не касалась основных функций ОС.

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

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

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

2.4. Использование формальных языков для 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 ранение. Информация часто представляется не только в количественной, но и в качественной форме. В первую очередь такого рода задачи используются при применении азличных автоматических систем обработки информации, при поиске конструкторско-ехнологической информации в системах автоматизированного проектирования, в оперативном управлении предприятиями.

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

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

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

- 1) дескрипторные системы без грамматики;
- 2) дескрипторные системы с грамматикой.

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

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

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

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

Таблица 2.4.1 – Инверсный массив

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

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

Шаги поиска:

- 1) 0031, 0034. 0038. 0045 (дескриптор 101);
- 2) 0003, 0022. 0027. 0031 (дескриптор 105);
- 3) 0031, 0168, 1342 (дескриптор 205);
- 4) 0031 – ответ.

Ответ получается как отыскание общей части для всех шагов поиска.

Иногда дескрипторы запросов делятся на категории, указывающие их важность.

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

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

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

Иногда роль дескрипторов определяется их положением в наборе или связями между отдельными дескрипторами. В этом случае дескрипторные системы приближаются по своему принципу действия к фактографическим информационным системам с грамматикой. Например, для приведенного выше запроса можно потребовать, чтобы слова «электронный банк 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 другого 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, по-

стов в различных областях деятельности. Уже современные автоматизированные хранилища позволяют в малых объемах сосредоточить колоссальные объемы информации. Например, с помощью лазерной технологии порядка тысячи страниц текста и чертёж можно записать на специальной пластинке, которая по размерам примерно равна этикетке спичечного коробка. Телевидение, спутниковые и другие средства связи позволяют обеспечить передачу информации на большие расстояния.

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

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

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

2.5. Базы данных и знаний

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 обмена данными с выполняемыми одновременно другими задачами, а при создании интеллектуальных систем и подбора методов для разрешения создавшихся ситуаций. Это привело к развитию систем, близких по духу к операционным, но 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 базы данных и знаний. Идея базы данных имеет в своей основе подход в решении ряда задач, когда исходные данные и другие промежуточные результаты хранятся отдельно от исполняемых программ и с помощью заранее описанных механизмов управления извлекаются из базы данных по стандартным схемам, используемым каждой задачей независимо от вычислительного процесса на стадии подготовки исходного материала для текущего этапа его реализации. Базу данных часто определяют как совокупность взаимосвязанной информации, организованной по определенным правилам. База данных обычно определена по схеме, не зависящей от программ, которые к ней обращаются. Чтобы легче искать данные и их обрабатывать, обычно используют системы управления базами данных (СУБД), которые ориентированы на поддержку выполнения различных запросов и ведения БД. На практике шире других используются реляционные СУБД, когда данные отображаются в табличной форме. В каждой СУБД имеется специальный язык запросов, обеспечивающий эффективный доступ к различным элементам (или их группам) в базе данных.

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

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

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

К схемам такого рода относятся и банки данных, позволяющие осуществлять в текстовых документах поиск необходимой информации на основе логических конструкций, построенных пользователем с применением ключевых слов.

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

При автоматизации проектирования и принятия решений в различных областях деятельности полезно в БД иметь особый подраздел в виде архива готовых прототипов (проект-

тов решений). Однако это связано с рядом особенностей введения этого подраздела, так как необходим экспертный отбор элементов архива, их заверения электронной цифровой подписью с выполнением аналогичных условий и при изъятии элементов архива.

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

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

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

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

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

Рассмотрим, например, фрагмент программы на алгоритмическом языке Паскаль.

$P_i := 3,14;$

$R := 20;$

$S := P_i * R * R;$

WRITELN ('Площадь круга S =', S).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Теперь на первом этапе прямого вывода появляется возможность применять либо правило 2, либо правило 3. Если сначала применить правило 2, то на следующем этапе можно будет применять правило 1 и правило 3. Если на этом этапе применить правило 1, то выполнится условие останова системы, но если прежде применить правило 3, то 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 это множество носит название *конфликтного набора*, а выбор одного из них называется *разрешением конфликта*.

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

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

Если на основании этого условия подтверждается цель «использовать джип», то для достижения первоначальной цели достаточно применить только одно правило 1, однако, чтобы подтвердить новую цель «дорога ухабистая», открывается возможность применения правила 1, нужно использовать либо правило 2, либо правило 4. Если сначала применить правило 2, то это будет самый удачный выбор, поскольку сразу же можно применить и правило 1. С другой стороны, если попытаться применить правило 2, то, поскольку образца «место отдыха – пляж», который является условием правила 4, в ра-

бочей памяти не существует и, кроме того, не существует правила, подтверждающего его, данный выбор является неудачным. И лишь со второго захода, применяя правило 2, можно подтвердить цель "дорога ухабистая".

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

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 понятия абстрактного образа. Например, слово «автомобиль» вызывает у слушающих образ устройства, 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 отдельных единиц, называемых слотами. Он имеет однородную структуру:

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

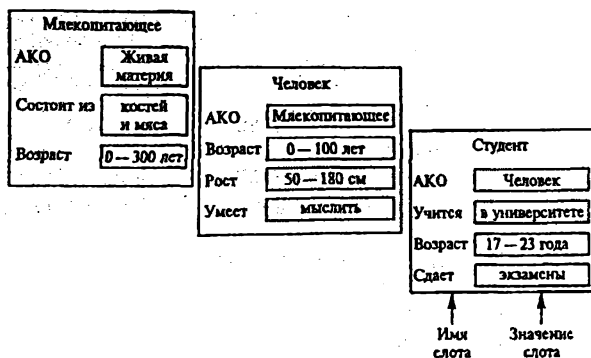


Рисунок 2.5.1 – Сеть фреймов

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 любые знания 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 (рис. 2.5.2). Сам термин «семантическая» означает «смысловая».

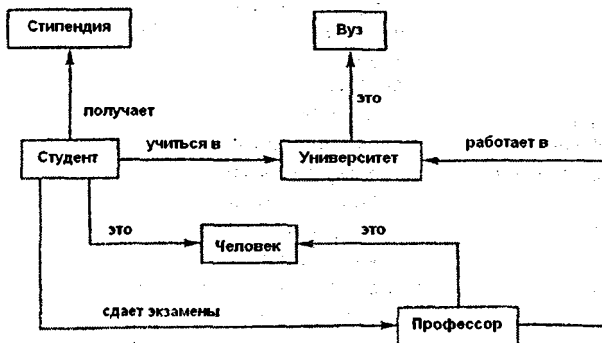


Рисунок 2.5.2 – Семантическая сеть

Основным преимуществом этой модели является наглядность представления знаний, а 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.

2.6. Языки в проектировании вычислительных систем

Языки логического управления. В системах логического управления традиционно используются *булевы функции и системы булевых функций*, задаваемые в форме таблиц истинности для полностью определенных функций и таблиц решений для неполностью определенных функций. При этом таблицы истинности, описывающие автоматы с памятью, носят название кодированных таблиц переходов, или кодированных таблиц переходов и выходов. Применение таблиц истинности ограничивается задачами небольшой размерности, а применение таблиц решений – в основном автоматами без памяти – комбинационными схемами.

Системы булевых формул могут быть изоморфно реализованы лестничными или функциональными схемами. Иногда используются также и другие аналитические формы представления булевых функций, например, пороговые, спектральные 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, сети Петри и графы операций*. На этапе перехода от автоматного описания к тексту программ целесообразно применять алгоритмические языки.

Язык ПРАЛУ параллельных алгоритмов логического управления. Язык ПРАЛУ удобно использовать для описания систем, характеризующихся сложным взаимодейст-

вием, асинхронностью и параллелизмом. Он объединяет возможности моделей «если-то» с возможностями сетей Петри и обладает средствами для представления последовательности текущих состояний вычислительной системы. С помощью языка ПРАЛУ возможно описание временной упорядоченности событий, возникающих при реализации протокола обмена сообщениями, абстрагируясь 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 « $\rightarrow A_i$ » приводит к наступлению некоторого события, представленного также конъюнкцией логических переменных, в описываемом 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 « $\mu_i: \eta_i \rightarrow v_i$ »; через η_i обозначен некоторый линейный алгоритм, составленный из операций языка; μ_i и v_i – начальная и конечная метки, которыми служат непустые 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 « $\mu_i: \eta_i \rightarrow v_i$ » выполняется условие $\mu_i \subseteq N_i$ и реализуется событие p_i , с ожидания которого начинается цепочка η_i , то она запускается. При этом N_i заменяется на $N_i \setminus \mu_i$, а после завершения цепочки N_i становится равным $(N_i \setminus \mu_i) \cup v_i$. 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 гашения в трех модификациях: « $\rightarrow *$ », « $\rightarrow * \gamma$ » и « $\rightarrow * \gamma'$ », где $\gamma \subseteq M$. Ее действие заключается в прекращении реализации всех или некоторых (из множества γ или $\gamma' = N_i \setminus \gamma$) активных цепочек алгоритма. Наряду с булевыми в ПРАЛУ допустимо использование и арифметических переменных, в частности введены операции: задержки « $- l$ » – выдержки l единиц времени; счета событий: « $\rightarrow (x = n)$ » – присвоения многозначной переменной x натурального значения n ; « $\rightarrow (x +)$ » и « $\rightarrow (x -)$ » – единичного положительного и отрицательного « $\rightarrow (x -)$ » приращения значения; « $-(x = n)$ » – ожидания наступления события: значение x равно n .

ГСА и ПРАЛУ-описания равносильны. Операторным вершинам ставится в соответствие операции действия, цепочкам условных вершин – операции ожидания. Кроме того, помечаются входы вершин ГСА, соединенные с выходами нескольких вершин, и входы условных вершин, соединенные с выходами операторных вершин, если данные условные вершины не являются ждущими. 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 « $\mu_i: -p \rightarrow a \rightarrow v_i$ », где μ_i и v_i – соответственно начальное и конечное состояния перехода, p – условие перехода, a – выходной сигнал. Фрагмент ГСА, изображенный на рис. 2.6.1, представится частью таблицы задания автомата с абстрактным состоянием в виде табл. 2.6.1.

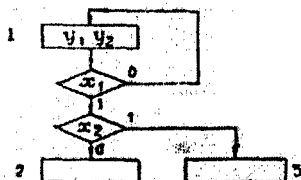


Рисунок 2.6.1 – Фрагмент ГСА

Таблица 2.6.1 – Автомат с абстрактными состояниями

Условие перехода	Начальное Состояние	Конечное Состояние	Выходной сигнал
-	1	-	y_1, y_2
$\neg x_1$	1	1	-
$x_1 \wedge x_2$	1	2	-
$x_1 \wedge \neg x_2$	1	3	-

Таблица 2.6.2 – Эквивалентное задание автомата с абстрактными состояниями

Условие перехода	Начальное Состояние	Конечное состояние	Выходной сигнал
$\neg x_1$	1	1	y_1, y_2
$x_1 \wedge x_2$	1	2	y_1, y_2
$x_1 \wedge \neg x_2$	1	3	y_1, y_2

Графовые модели электронных изделий. Для решения задач автоматизированного проектирования сложных электронных изделий и БИС используют различные математические модели: графовые эквиваленты коммутационных схем соединений отдельных частей объекта (мультиграфы, гиперграфы, ультраграфы); графы пересечений для отображения взаимного расположения соединений; неориентированный топологический граф решетки для задания метрического пространства. Их использование сохраняет наглядность и содержательность описываемых объектов и позволяет строить алгоритмы обработки для ЭВМ.

Любую функциональную или принципиальную схему объекта можно представить состоящей из множества $E = \{e_1, e_2, \dots, e_n\}$ элементов, множества $\{c_i\}$ выводов (контак-

тов) каждого элемента e_i и множества цепей (эквивипотенциальных контактов), соединяющих элементы схемы. В зависимости от целей проектирования решаемых задач можно использовать различную информацию об исследуемой схеме. Рассмотрим наиболее употребительные в практике автоматизированного проектирования графовые эквиваленты схем, использующие известные 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, и имеет большое практическое значение: мультиграфы коммутационных схем применяются при решении задач декомпозиции и оптимального размещения элементов в пространстве. На рис. 2.6.2 приведен фрагмент коммутационной схемы, состоящий из пяти элементов $E = \{e_1, e_2, \dots, e_5\}$ и шести цепей $V = \{v_1, v_2, \dots, v_6\}$. Данному фрагменту схемы соответствует математическая модель в виде мультиграфа, представленного на рис. 2.6.3.

Для целей автоматизированной обработки используют матричные эквиваленты мультиграфа: матрицу смежности $A = \|a_{ij}\|_{n \times n}$ и матрицу инцидентности $S = \|s_{ij}\|_{n \times k}$, 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 задач 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 в многом неформально.

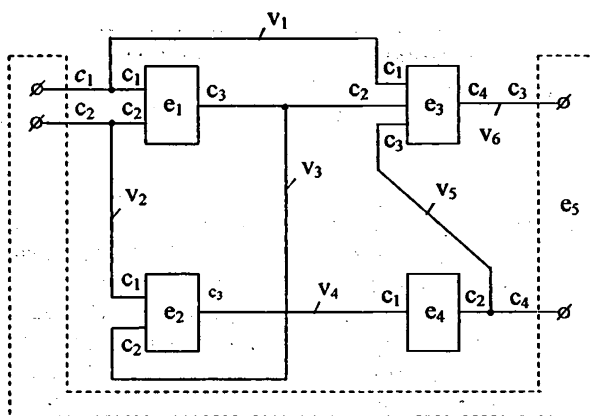


Рисунок 2.6.2 – Фрагмент коммутационной схемы

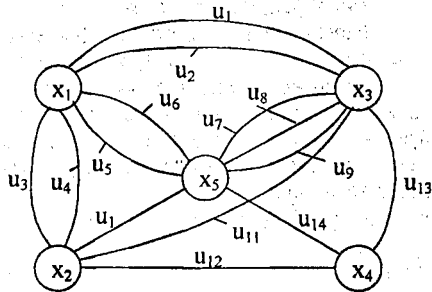


Рисунок 2.6.3 – Мультиграф коммутационной схемы

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 течении времени $t(n)$, начиная с момента времени t_n .

Например, диаграмма автомата, показанная на рис. 2.6.4, представляется в виде последовательности, которую для наглядности представим в виде таблицы 2.6.3.

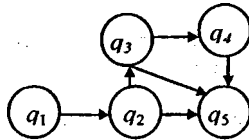


Рисунок 2.6.4 – Диаграмма автомата

Таблица 2.6.3 – Временная диаграмма

n	$t(n)$	$U(n)$	$q(n)$	$V(n)$
1	3	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 2,3,5,2), строки 6-9 – цикл (2,3,4,5,2).

Язык описания схем VHDL. Из четырех конструкций верхнего уровня языка VHS 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.

Формально конструкция Entity записывается следующим образом:

```
entity <имя_entity> is
  generic (<generic_list>);
  port (<port_list>);
  <entity_declarative_part>
end <имя_entity>;
```

Конструкция generic_list используется для задания препроекторных констант и другой статической информации, определяемой окружением. Она используется не всегда. Конструкция port_list является основной. Она содержит описание портов, т.е. входов и выходов схемы. Декларативная часть entity содержит объявления сигналов, констант, введенных пользователем новых атрибутов и типов данных, а также тексты подпрограмм и функций. Все эти объекты могут использоваться в любой архитектуре, относящейся к этому entity. Кроме того, если некоторые из этих объектов используются в различных entity, то объявления можно выделить в отдельную конструкцию, называемую пакетом, и в декларативной части entity вместо объявлений поместить ссылку на этот пакет.

Стили описания архитектур. При описании архитектуры средствами языка VHS могут использоваться три стиля, которые обеспечиваются разными конструкциями языка:

- 1) поведенческий;
- 2) потоковый;
- 3) структурный.

При поведенческом описании алгоритм записывается последовательными операциями (аналогичные которым имеются в большинстве языков программирования), при потоковом описании архитектура представляется в виде множества параллельных регистровых операций, а при структурном описании проект представляется в виде множества компонентов, 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 поступит сигнал на этот вход.

Поведенческий стиль описания является основным для пользователей САПР, так как именно он используется для создания VHDL-описания, поступающего на вход САПР. В общем случае поведенческое описание выглядит следующим образом:

```
architecture <имя_архитектуры> of <имя_entity> is
  <architecture_declarative_part>
begin
  process (sensitivity_list_1)
  <process_declarative_part>
  begin
    <sequential_statements_1>
  end process;
```

...

```

process (sensitivity_list_n)
  <process_declarative_part>
  begin
    < sequential_statements_n>
  end process;
end <имя_архитектуры>;

```

Декларативная часть архитектуры может содержать такие же конструкции, как и декларативная часть Entity. Процессы, как и другие операторы, допустимые в теле архитектуры, выполняются параллельно. В то же время в теле процесса, как и в теле функции или подпрограммы могут присутствовать только последовательные операторы. В основном эти операторы аналогичны операторам таких языков как Си и Паскаль. Кроме того в языке VHDL имеются последовательный оператор назначения сигнала и оператор ожидания (wait), которые позволяют описывать алгоритмы, работающие в реальном масштабе времени. Так как в языке VHDL нельзя смешивать последовательные и параллельные операторы, встает проблема описания алгоритмов, содержащих параллельно выполняемые действия. Эта проблема имеет следующие решения. Во-первых, так как последовательные операторы (кроме оператора wait) выполняются за нулевое время, группа расположенных рядом операторов назначения сигнала стартует одновременно, т.е. эти назначения сигналов можно считать выполняющимися параллельно. Во-вторых, в более сложных случаях можно воспользоваться оператором wait.

Рассмотрим алгоритм, представленный на рис. 1.1.6, с параллельным выполнением алгоритмов (операторов) 2 и 3. На языке VHDL его можно описать следующим образом:

```

architecture PAR of <имя_entity> 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 сигнал 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 вы-

полнения 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.

Отметим, что кроме описания параллельных алгоритмов, оператор wait может использоваться для описания генератора синхросигналов, хотя в ряде случаев это удобнее сделать при помощи оператора назначения сигнала. В качестве примера рассмотрим схему, которая должна выполнять некоторые действия с периодом в 1 с:

```
entity PERIOD_ALG is
    generic (period: time:= 1 sec);
    port (<port list>);
end PERIOD_ALG;
architecture AR of PERIOD_ALG is
    signal IND: bit:= '0';
begin
    P1: process (ind)
        begin < алгоритм > end process P1;
    P2: process
        begin
            IND <= not IND after period; end process P2;
        end AR;
```

В этом примере через время, равное периоду 1 с, будет изменяться значение сигнала IND, а, следовательно, стартовать процесс P1.

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 схемы. В частности по потоковому описанию можно построить аналогичное поведенческое описание:

```

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;

```

Могут существовать различные структурные описания одного и того же алгоритма, так как при построении схемы могут использоваться различные элементы. Структурное описание нашей схемы может иметь следующий вид:

```

architecture STRUCT of ONES_CNT is
  component MAJ3 port (X: in bit_vector (0 to 2); Z: out bit);
  component OPAR3 port (X: in bit_vector (0 to 2); Z: out bit);
begin
  COMP_1: MAJ3 port map (A,C(1));
  COMP_2: OPAR3 port map (A,C(0));
end STRUCT;

```

Согласно этому описанию, схема GNES_CNT состоит из двух подсхем: OPAR3 и MAJ3, первая из которых вычисляет нулевой бит результата, а вторая – первый. Чтобы полностью описать схему ONES.CNT, нужно описать схемы OPAR3 и MAJ3 Описание может быть любого типа, мы приведем структурное.

```

entity MAJ3 is
  port (X: in bit_vector (0 to 2)); Z: out bit);
end MAJ3;
architecture AND_OR of MAJ3 is
  component AND2 port (11,12: in bit; O: out bit);
  component OR3 port (11,12,13: in bit; O: out bit);
  signal A1,A2,A3: bit;

```

```

begin
  G1: AND2 port map (X(0), X(1), A1);
  G2: AND2 port map (X(0), X(2), A2);
  G3: AND2 port map (X(1), X(2), A3);
  G4: OR3 port map (A1, A2, A3, Z);
end AND_OR;

```

Схема MAJ3 состоит из трех экземпляров компонент типа AND2 и одного компонента типа OR3, связи между которыми задаются картой портов. Структурное описание схемы OPAR3 имеет аналогичный вид, но несколько сложнее:

```

entity OPAR3 is
  port (X: in bit_vector(0 to 2); Z: out bit);
end OPAR3;
architecture AND_OR of OPAR3 is
  component AND3 port (I1, I2, I3: in bit; O: out bit);
  component O4 port (I1, I2, I3, I4: in bit; O: out bit);
  component INV port (I1: in bit; O: out bit);
  signal A1, A2, A3, A4, B0, B1, B2 bit;
begin
  G0: INV port map(X(0), B0);
  G1: INV port map(X(1), B1);
  G2: INV port map(X(2), B2);
  G3: AND3 port map(X(2), B1, B0, A1);
  G4: AND3 port map(B2, B1, X(0), A2);
  G5: AND3 port map(X(2), X(1), X(0), A3);
  G6: AND3 port map(B2, X(1), B0, A4);
  G7: OR4 port map(A1, A2, A3, A4, Z);
end AND_OR;

```

В свою очередь для компонент AND2, AND3, OR3, OR4, INV (инвертор) также должны быть приведены описания, например, потоковые.

Описание автоматов. Описания автоматов на языке VHDL покажем на примере автомата Мили, заданного таблицей переходов и выходов.

	z1	z2	z3
x1	z2/y1	z2/y2	z1/y1
x2	z3/y2	z3/y2	z3/y1

```

entity MEALY is
  port (x: in integer range 1 to 2; y: out integer range 1 to 2);
end MEALY;
architecture AR of MEALY is
  constant period: time:=10ns; -- clock
  signal clk: bit:=0';
  signal z: integer range 1 to 3:=1;
begin
  --z -- states
  CLOCK: process begin
    clk <= not clk after period;
  end process CLOCK;

```

```

MAIN : process (clk) begin
  case z is -- transitions
    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;
    end case;
  end case;
  when 3 => case x is
    when 1 => z <= 1; y <= 2;
    when 2 => z <= 3; y <= 1;
  end case;
end process MAIN;
end AR;

```

Если входами являются вектора битов, то блок синхронизации имеет вид:

```

begin
EXTERN : process – clocking
  begin
    u(1) <= '0', '1' after 5 ns, '0' after 15 ns;
    u(2) <= '0',
    ...
  end process EXTERN;

```

Литература

[10, 15, 18, 20, 22-25, 30]

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 грамматик?
6. Какие основные этапы необходимо выполнить при разработке алгоритмического языка и его транслятора?
7. В чем заключается принципиальное отличие трансляторов компилирующего и интерпретирующего типа?
8. Какие основные функции выполняет операционная система ЭВМ?
9. В чем заключается отличие базы данных от базы знаний?
10. Каковы основные функции систем управления базой данных (СУБД)?
11. В чем отличие банка данных от базы данных?
12. Чем отличаются процедурные и декларативные знания?
13. Назовите особенности продукционных моделей вывода.
14. Чем отличаются семантические представления знаний?
15. Назовите основные языки для проектирования ВС и укажите цели их применения.

Глава 3. МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОЕКТИРОВАНИИ

3.1. Проблемы создания искусственного интеллекта

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

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

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

Развитие робототехники в последние годы 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. А. Тьюринг развивал функциональный подход к оценке деятельности автоматов. Такой подход нашел сторонников при решении вопросов о возможности машинного мышления.

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

Имеются 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 (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 приме-

рами для эксперимента могут послужить решение шахматных этюдов или задач, проектирование режущего инструмента и т.д.

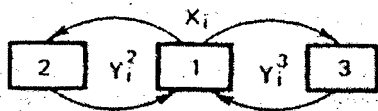


Рисунок 3.1.1 – Схема эксперимента Тьюринга

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 по типу 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 эффективно мозг с помощью случайно скоммутированной сети маловероятно. Другой путь связан с моделированием работы мозга на уровне информационных процессов, добиваясь того, 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. Клыковым, 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 процесс как для построения модели объекта, так и для управления им. Однако решение задач этим методом из-за отсутствия специального математического обеспечения достаточно трудоемко и 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 главным образом 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 построению 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 наблюдений 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 эвристике рассматриваются лишь некоторые методы решения задач, но не исследуются вопросы, 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 эффективный 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 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 предположительные решения и делать следующие попытки в более перспективном направлении.

Процедура распознавания образов позволяет не исследовать все возможности, организовать поиск, 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. По этому решается вопрос о принадлежности исследуемого объекта к одному из образцов соответствия с весом выявленных признаков. Поэтому основной задачей многих эвристических программ является отыскание хорошей системы признаков, инвариантных носительно 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 обучающаяся 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 последовательностью 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 дает 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. обеспечить 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 нет программ с развитым индуктивным мышлением.

Все пять видов процедур в той или иной степени были использованы впервые в США в 60-х годах А. Ньюэллом, Дж. Шоу и Г. Саймоном в программе «Общий решатель задач», предназначенной 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 возможных альтернатив. Авторы программы выдвинули тезис, что решение любой задачи человеком состоит в поиске пути, приводящего от начальной площадки лабиринта, соответствующей исходному описанию ситуаций, к некоторой конечной площадке, соответствующей решению этой задачи. На каждом шаге поиска пути используются локальные критерии успеха, 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 теорем исчисления высказываний и игровые программы. Но проверка универсальности процедуры поиска по лабиринту на шахматной программе показала ее авторам, что, основываясь лишь на ней, практически невозможно решить сколько-нибудь серьезную задачу.

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

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

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

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

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

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

Не останавливаясь детально на философских аспектах этой проблемы, будем считать, что процесс получения любого результата всегда расчленяется на несколько этапов, каждый из которых может полностью или частично выполняться вычислительной машиной. Если аналогичные результаты, полученные человеком на каком-то этапе или на нескольких автоматизированных этапах, квалифицируются как творческие, то будем и машинный результат относить к разряду творческих. При невозможности автоматизировать весь процесс, можно остановиться на автоматизации лишь его рутинных этапов. Однако во всех случаях машиной будет получаться либо творческий результат, либо использование машины поможет человеку прийти к этому результату. На наш взгляд, в практике не существует резкой границы при классификации задач на творческие и рутинные. Всегда разумно говорить о сложности той или иной конкретной задачи и необходимом уровне творчества для ее решения, т.е. считать, что творчество носит многоуровневый характер. Термин «машина» становится более расплывчатым, так как многие процессы могут выполняться программно и 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 и принятия различных логических решений. Использование специалистами на своих рабочих местах удаленных от ЭВМ терминалов или персональных ЭВМ позволит использовать машину в качестве основного инструмента для выполнения практически любых творческих работ. Однако эффективность использования этого инструмента будет определяться полнотой и качеством системы из отдельных модулей, ориентированных на решение частных вопросов, и наличием проблемно-ориентированного входного языка, удобного для пользователей.

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

3.2. Диалоговые методы решения задач и экспертные системы

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

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

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

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

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

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

Например, при решении аэродинамической задачи о входе космического аппарата в атмосферу Земли необходимо рассчитать силовые нагрузки, тепловой режим и т.д. Получить такие 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 строится математическая модель для выполнения вычислительного эксперимента, который позволяет сэкономить миллионы рублей.

Использование при решении задач проектирования больших банков данных, в которых хранятся типовые решения и элементы, поможет обеспечить получение новых конструкций

из стандартных 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 позволит любому 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 человека 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 ними объекты. Электронная вычислительная машина в таких случаях может стать тем звеном, которое в диалоговом режиме поможет группе специалистов, находящихся в разных местах, прийти к эффективному решению.

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

Например, проблемы загрязнения среды, развития животного и растительного мира, осушения 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 связи 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 корректировок 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 процедур обмена информацией, порождает еще массу проблем, связанных с достоверностью передачи данных и их защитой от постороннего вмешательства.

Для рядового пользователя ЭВМ – специалиста только в своей области – сгусток всех этих специфических вопросов 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 узлах 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. Персональные являются компонентами в локальных сетях ЭВМ и используются в автоматизации подготовки данных, «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, обла

дающими 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. Такими знаниями обладают экстрасенсы, контактеры, шаманы.

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

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

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

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

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

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

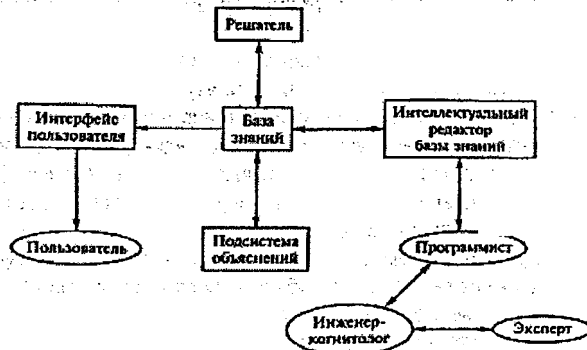


Рисунок 3.2.1 – Типичная блок-схема экспертной системы

- база знаний – ядро экспертной системы, совокупность знаний предметной области, записанная на машинном носителе в форме, понятной эксперту и пользователю;
- интеллектуальный редактор базы знаний – программа, представляющая инженеру-когнитологу и программисту возможность создавать базу знаний в диалоговом режиме (Она включает в себя системы вложенных меню, шаблонов языка представления знаний, подсказок (help-режим) и других сервисных средств, облегчающих работу с базой знаний);
- интерфейс пользователя – комплекс программ, реализующих диалог пользователя с экспертной системой на стадии как ввода информации, так и получения результатов;
- решатель (синонимы; дедуктивная машина, блок логического вывода) – программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в базе знаний;
- подсистема объяснений – программа, позволяющая пользователю получать ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?». Ответ на вопрос «Как?» – это трассировка всего процесса получения решения с указанием исполняющих фрагментов базы знаний, т.е. всех шагов цепи умозаключений. Ответ на вопрос «Почему?» – ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т.е. отход на один шаг назад.

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

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

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

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, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном экспертной системе. Из-за эвристического характера знаний их приобретение является весьма трудоемким.

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 экспертной системы.

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

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

Переконструирование выбранного ранее способа представления знаний предполагает возврат 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 идентификации. В этом случае речь идет о переформулировании понятий, используемых в системе, т.е. перепроектировании системы заново.

3.3 Распознавание образов и обработка изображений

Общая характеристика задач распознавания образов и их типы. Под образом понимается структурированное описание изучаемого объекта или явления, представленное вектором признаков, каждый элемент которого представляет числовое значение одного из признаков, характеризующих соответствующий объект. Общая структура системы распознавания и этапы в процессе ее разработки показаны на рис. 3.3.1.

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

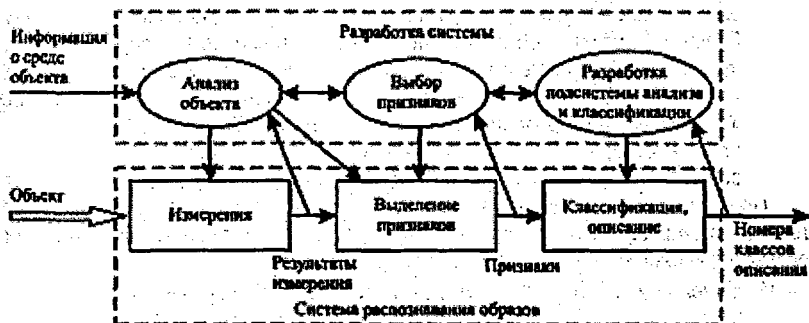


Рисунок 3.3.1 – Структура и этапы разработки системы распознавания

Задачи распознавания имеют следующие характерные черты.

1. Это информационные задачи, состоящие из двух этапов: а) приведение исходных данных к виду, удобному для распознавания; б) собственно распознавание (указание принадлежности объекта определенному классу).

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

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

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

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

Целесообразно выделить следующие типы задач распознавания.

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

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

3. Задача выбора информативного набора признаков при распознавании.

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

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

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

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 низкий процент 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 некоторой 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».

Возможен и иной подход к обнаружению и локализации объектов на изображениях. Он заключается в проведении классификации отдельных пикселей или групп пикселей, после чего может быть проведена локализация объектов, например, на основании близости на изображении пикселей одного класса.

Пусть $f(x, y)$ – дискретная функция, описывающая анализируемое изображение; X – конечное 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 точек (x, y) из каждого подмножества S_i удовлетворяет некоторому критерию однородности этого подмножества. Сегментацией изображения $f(x, y)$ по предикату L_p называется разбиение $S = \{S_1, S_2, \dots, S_k\}$, удовлетворяющее условиям:

$$а) \bigcup_{i=1}^k S_i = X;$$

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

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

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

Отметим, что k областей, полученных в результате сегментации часто различных классов, 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 гранично-ориентированные, когда определяются границы между S_i . В методах первой группы критерием однородности при задании L_p выступают общие свойства изображения: значение интенсивности (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

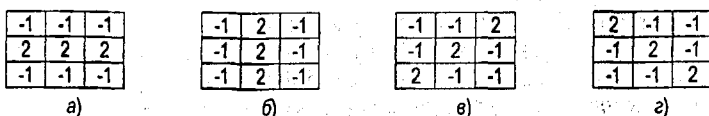
реализуется отслеживание контуров и при определении L_p используется характеристика изменения интенсивности – разность величин и направления градиента, т. е. два пикселя (x, y) и (x', y') подобны, если имеет место соотношение

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

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

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

Для вычисления градиента используются операторы локального анализа: Собеля, Робертса, Кирша и т. д. В свою очередь, операторы свертки позволяют непосредственно отобрать пиксели, которые, например, могут принадлежать прямым определенной ориентации и ширины (рисунок 3.4).



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

Рисунок 3.3.2 – Операторы свертки для выделения прямых, толщиной в один пиксель

Не существует универсального критерия эффективности названных методов – выбор того или иного метода зависит от конечной цели всего процесса обработки изображений, типа обрабатываемых слоев и имеющихся вычислительных мощностей.

Найденные контура обычно представлены в ортогональной решетке, которые затем аппроксимируются 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. Предпочтение отдается последнему преобразованию, которое обладает инвариантностью к ряду искажений:

$$x \cos \theta + y \sin \theta = \rho, \quad (3.5)$$

где θ – угол, образуемый нормалью к прямой и осью X ; ρ – расстояние от начала координат до прямой.

При распознавании объектов используются три типа моделей для представления информации о классах: детерминистская – в виде эталонов с фиксированными значениями признаков; нечеткая (квази- и псевдонечеткая, 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 $f_{i, эт}$. Для этого формируются некоторые функционалы – меры близости $\rho(f, f_{i, эт})$. Наиболее часто ис-

пользуются следующие меры:

$$\rho_1^1 = \max |f(x, y) - f_{i,sm}(x, y)|;$$

$$\rho_1^2 = \left\{ \frac{1}{S_D} \iint_{(D)} [f(x, y) - f_{i,sm}(x, y)]^2 dx dy \right\}^{\frac{1}{2}};$$

$$\rho_1^3 = \frac{\left[\iint_{(D)} f(x, y) f_{i,sm}(x, y) dx, dy \right]^2}{\iint_{(D)} f(x, y) dx dy \iint_{(D)} f_{i,sm}^2(x, y) dx, dy}.$$
(3.6)

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 меры близости естественно задавать минимальные значения ρ_1^1 и ρ_1^2 , но максимальное значение ρ_1^3 , достигается путем вариации этих параметров.

Алгоритмы распознавания реализуют принцип похожести новых объектов с заданными (т. е. 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 ошибок отказов должно быть минимальным и не превышать заданное число 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 геометрических 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 шаг 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 $y = [y_1, y_2, \dots, y_M]$. При этом $y_i = F_i(x_1, x_2, \dots, x_N) = F_i(x)$, $i = 1, \dots, M$, где F_i – функции преобразования переменных или признаковые операторы. Эти функции (операторы) могут осуществлять просто отбор части исходных измерений ($M < N$), взвешивание $y_i = \omega_i x_i$, $i = 1, \dots, M$, глобальные (покальные) преобразования и т.д.

Для получения признаков могут быть использованы коэффициенты разложения по полному ортогональным системам функций. К ним следует отнести тригонометрические многочлены; 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 бинарные 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, расстояния. Они обеспечивают как высокую скорость, так и невысокую сложность их вычисления на ЭВМ. Тем не менее, конструирование масок для этого типа признаков является довольно трудной задачей.

Методы геометрического и топологического анализа признаков могут представлять как 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 получить признаки, инвариантные к некоторым глобальным деформациям изображения, таким как сдвиг или поворот. В исследованиях используют разложения Фурье, Уолша, Хаара, разложение Карунена-Лозва, ПХ, преобразования проекций, преобразование в цепной код и так далее. Процедура выделения признаков и конструирования масок считается несложной для признаков этого типа, однако высокие требования к компьютерным ресурсам являлись существенным недостатком этих методов. Глубокие теоретические разработки в области линейных операторов, одно – и двумерных преобразований, цифровой фильтрации, а также значительный прогресс в 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.

Выделение проекционных признаков на основе преобразования Хафа имеет инвариантные свойства по отношению к масштабированию и сдвигу, а на этапе классификации возможен учет и компенсация искажений поворота. Так, для обеспечения инвариантности к изменениям масштаба изображения достаточно подвергнуть его нормализации. Для этого значения функции $g(\rho, \theta)$ заключаются в интервал $[0, 1]$. Нормализация проводится в соответствии с формулой:

$$g_{norm}(\rho, \theta) = \frac{g(\rho, \theta)}{\max |g(\rho, \theta)|}, \quad r = 0, \dots, R-1, \quad (3.7)$$

где $g(\rho, \theta)$ – значение проекционной функции в точке ρ, θ ; $g_{norm}(\rho, \theta)$ – нормализованное значение проекционной функции в точке ρ, θ ; $\max |g(\rho, \theta)|$ – абсолютное максимальное значение проекционной функции.

Для достижения инвариантности признаков к повороту исходного изображения необходимо вычислять циклическую свертку результатов преобразования Хафа, либо использовать относительные значения параметра, определяющего ориентацию признака (например, угол φ_i между соседними прямолинейными сегментами). Выделенные признаки сначала упорядочиваются по возрастанию параметра θ , а затем используется соотношение:

$$\varphi_i = \theta_i - \theta_{i-1}; \quad i = 1, \dots, N, \quad \theta_0 = 0. \quad (3.8)$$

Фурье-дескрипторы формы позволяют описать границы объекта так, что она не будет зависеть от изменений типа параллельного перемещения, расширения и сужения, а также поворота. Замкнутую кривую $\Phi^*(t)$ представляем как ряд Фурье:

$$\Phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} (a_k \times \cos kt + b_k \times \sin kt). \quad (3.9)$$

В полярной форме соотношение (1.9) можно записать в виде

$$\Phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} A_k \times \cos(kt - \alpha_k), \quad (3.10)$$

где (A_k, α_k) есть Фурье-дескрипторы, представляющие собой амплитуду и фазу k -й гармоники.

Две кривые, которые отличаются только положением, ориентацией и размером, но имеют сходную начальную точку обхода, преобразуются в одинаковую волновую форму $\Phi^*(t)$ и, следовательно, имеют идентичные Фурье-дескрипторы. Известны специальные формы их представления для описания замкнутых кривых, и показано, что они инвариантны к сдвигу, повороту, масштабированию. Показана также эффективность применения в качестве дескрипторов формы коэффициентов Уолша с использованием быстрых процедур вычисления. Отметим, что преобразование Хафа не обладает свойством инвариантности спектра к циклическим сдвигам исходного сигнала, но с помощью моментных функций можно получить инвариантность относительно вращения и масштабирования. Выделение информативных признаков изображений на основе усеченных спектральных разложений в базисах одно- и двумерного преобразования Адамара дает возможность редуцирования пространства признаков приблизительно на 2/3 без ухудшения результатов классификации.

В качестве информативных признаков могут использоваться моменты различных видов: Лежандра, Цернике, комплексные, кольцевые и секторные моментные функции, которые обеспечивают инвариантность относительно сдвига, вращения и изменения масштаба. Центральные моментные функции вычисляются в системе координат с началом в центре масс изображения по формуле:

$$\mu_{kl} = \sum_{i=1}^N \sum_{j=1}^M (i - i_0)^k (j - j_0)^l x_{ij}, \quad (3.11)$$

где (i_0, j_0) – центр масс изображения. Они более приспособлены для проведения распознавания, поскольку обладают инвариантностью к сдвигу объекта. Все они могут быть выражены через геометрические моменты. Поэтому возникает задача нахождения быстрого алгоритма вычисления геометрических моментов изображения.

Известно также применение *вейвлет анализа* для выделения признаков, инвариантных к повороту, масштабированию и сдвигу. Это так называемый набор инвариантных вейвлет-моментов:

$$\Psi_{m,n}(r) = 2^{m/2} \psi(2^m r - 0.5n) \quad (3.12)$$

с материнской функцией вейвлета:

$$\psi(r) = \frac{4a^{n+1}}{\sqrt{2\pi(n+1)}} \sigma_w \cos(2\pi f_0(2r-1)) \times \exp\left(-\frac{(2r-1)^2}{2\sigma_w^2(n+1)}\right), \quad (3.13)$$

где $n = 3$; $a = 0.697066$; $f_0 = 0.409177$; $g_w^2 = 0.561145$.

Набор вейвлет-моментов, инвариантных к повороту, имеет вид:

$$\|F_{m,n,q}^{wavelet}\| = \left\| \int S_q(r) \cdot \Psi_{m,n}(r) \cdot r dr \right\|, \quad (3.14)$$

где $m = 0, 1, 2, 3$; $n = 0, 1, \dots, 2^{m-1}$; $q = 0, 1, 2, 3$; $S_q(r) = \int f(r, \theta) e^{i q \theta} d\theta$.

В качестве материнской функции можно применить и другие известные функции, например, функции Хаара, что приводит к некоторому увеличению скорости алгоритма без значительного уменьшения разделяющей способности признаков.

Выделение информативных признаков изображений на основе локальных преобразований широко используется в рамках статистического подхода к распознаванию образов. Примером локального преобразования может служить двумерная фильтрация массива отсчетов $F(i, j)$ размером $N \times N$ на основе свертки с массивом H размером $L \times L$, $L \ll N$:

$$Q(m, n) = \sum_{i=1}^m \sum_{j=1}^n F(i, j) H(m-i+1, n-j+1). \quad (3.15)$$

Если изображения, которые нужно анализировать, представлены как N -мерные векторы, то для построения классификатора используются дискриминантные методы, например, на основе преобразования Фоли-Сэммона или синтетических дискриминантных функций. *Задача дискриминантного анализа* формулируется следующим образом: на заданном множестве признаков x_1, x_2, \dots, x_M , характеризующих две (или более) группы, необходимо найти линейные дискриминантные функции вида

$$d = a + b_1 x_1 + \dots + b_M x_M \quad (3.16)$$

и использовать их для классификации наблюдений к одной из групп. Если классы в N -пространстве компактны и линейно разделимы и, кроме того, переменные статистически независимы и имеют одинаковую дисперсию $\Sigma^{(1)} = \Sigma^{(2)} = \Sigma = \sigma^2 I$, то классификация может осуществляться по минимуму евклидова расстояния, однако при этом возможны ошибки классификации. Учет структуры распределений можно обеспечить одновременной минимизацией суммарного внутриклассового разброса, что приводит к *критерию Фишера*

$$R = \frac{|\bar{x}^{(1)} - \bar{x}^{(2)}|^2}{S_2^{(1)} + S_2^{(2)}}, \quad (3.17)$$

где $S_2^{(1)}$ и $S_2^{(2)}$ – полная матрица рассеяния внутри двух классов.

Дискриминантные методы анализа могут быть основой для построения сценария параметрической классификации дефектов на изображении.

Дискретные ортогональные преобразования. При обработке изображений дискретные ортогональные преобразования широко применяются для восстановления и улучшения качества изображений: повышения резкости и контрастности; подавления помех, сжатия и кодирования изображений по спектральным признакам; восстановления после сжатия; выделения контуров объекта; устранения искажений сдвига, поворота и масштабирования.

Пусть $f(n)$ ($n = 0, \dots, N - 1$) – периодическая с периодом N комплексная последовательность, $\{w_m(n)\}_{m=0}^{N-1}$ – семейство N -периодических комплексных функций с условием ортогональности

$$\langle w_m, w_k \rangle = \sum_{n=0}^{N-1} w_m(n) \overline{w_k(n)} = \delta_{mk}, \quad (3.18)$$

где δ_{mk} – дельта-символ Кронекера, а черта означает комплексное сопряжение.

Преобразование $F(m) = \sum_{n=0}^{N-1} f(n) w_k(n)$ ($m = 0, 1, \dots, N - 1$) называется дискретным ортогональным преобразованием с базисом $W_N = \{w_m(n)\}_{m=0}^{N-1}$.

Используя специальные базисные функции, получаем дискретные преобразования Фурье, Хаара, Адамара и Уолша, которые широко используются в обработке изображений. Эффективность их применения обусловлена тем, что они имеют быстрые алгоритмы вычислений.

Дискретные преобразования Фурье, Уолша и Хаара. Дискретным преобразованием Фурье (ДПФ) называется пара преобразований:

$$X(k) = \sum_{n=0}^{N-1} x(nT) W_N^{kn}; \quad (3.19)$$

$$x(nT) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad (3.20)$$

где $x(nT)$ ($n = 0, \dots, N - 1$) – последовательность из N временных отсчетов с периодом T ;

$X(k)$ ($k = 0, \dots, N - 1$) – последовательность из N частотных отсчетов; $W_N = e^{\frac{-2\pi i}{N}}$.

Преобразование (3.19) называется *прямым ДПФ*, а (3.20) – *обратным ДПФ*. Функции $W_N^{nk} = (W_N)^{nk} = e^{\frac{-2\pi i nk}{N}}$ являются базисными для преобразования Фурье. ДПФ используется для спектрального анализа, на котором основаны процедуры улучшения изображений и распознавания объектов, т. е. позволяет получать спектр изображения, характеризующий его на различных пространственных частотах. Кроме того, применение ДПФ дает возможность реализовать свертку сигналов и определить их корреляционную функцию, что используется при распознавании.

Множество функций Хаара $har(r, m, t)$ образует периодическую ортонормированную и полную систему функций и для них справедлива следующая рекуррентная зависимость:

$$har(0,0,t) = 1,$$

$$har(r,m,t) = \begin{cases} 2^{r/2}; \frac{m-1}{2^r} \leq t < \frac{m-1/2}{2^r}; \\ -2^{r/2}; \frac{m-1/2}{2^r} \leq t < \frac{m}{2^r}; \\ 0; \text{в остальных случаях,} \end{cases} \quad (3.21)$$

где $0 \leq t < 1$; $0 \leq r < \log_2 N$; $1 \leq m \leq 2^r$.

Рассмотрим первые восемь функций Хаара ($N = 8$; $r = 0, 1, 2$; $m = 1, 2, 3, 4$). Тогда преобразование Хаара n -периодичной последовательности $\{x(n)\} = \{x(0), x(1), \dots, x(n-1)\}$ определяется как

$$\{X(n)\} = \frac{1}{N} [H_C(n)] \cdot \{x(n)\}, \quad (3.22)$$

где $\{x(n)\}^T = \{x(0), x(1), \dots, x(n-1)\}$ – векторное представление последовательности $\{x(n)\}$;

$$\{X(n)\}^T = \{X(0), X(1), \dots, X(n-1)\} – \text{коэффициенты преобразования Хаара.}$$

Сигнал может быть однозначно восстановлен при помощи обратного преобразования Хаара:

$$\{x(n)\} = [H_C(n)] \cdot \{X(n)\}, \quad (3.23)$$

где $[H_C(n)]$ – матрица преобразования Хаара, которая получается в результате двоиной перестановки индексов входной последовательности и может быть рекурсивно сгенерирована следующим образом:

$$[H_C(0)] = [1];$$

$$[H_C(n)] = \begin{bmatrix} [H_C(n-1)] & [H_C(n-1)] \\ 2^{n-1/2} [I(n-1)] & -2^{n-1/2} [J(n-1)] \end{bmatrix} \quad (3.24)$$

где $[I(n)]$ – единичная матрица размера $2^n \times 2^n$. На основе $[H_C(n)]$ проектируется быстрое преобразование Хаара.

Дискретные функции Уолша длины $N = 2^m$ принимают только два значения $+1$ и -1 и определяются как строки матрицы W_N размерности $N \times N$, удовлетворяющей следующему рекуррентному соотношению (матрицы Адамара)

$$W_N = \begin{bmatrix} W_{N/2} & W_{N/2} \\ W_{N/2} & -W_{N/2} \end{bmatrix}, W_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, N = 2^m. \quad (3.25)$$

Для вычисления преобразования Уолша используется алгоритм, аналогичный алгоритму Кули-Тьюки для БПФ.

Двумерные дискретные ортогональные преобразования. Для изображений, представляющих собой двумерный сигнал, спектром является также двумерный сигнал. 2-ДПФ определяется следующими формулами:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] \cdot W_{N_1}^{n_1 k_1} \cdot W_{N_2}^{n_2 k_2}; \quad (3.26)$$

$$x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X[k_1, k_2] \cdot W_{N_1}^{n_1 k_1} \cdot W_{N_2}^{n_2 k_2}, \quad (3.27)$$

где $x[n_1, n_2]$ – исходный сигнал; $X[k_1, k_2]$ – спектр сигнала; $N_1 \times N_2$ – размер исходного сигнала и его спектра; k_1 и k_2 – номера базисных функций (номера коэффициентов 2-ДПФ, при которых эти функции находятся).

Поскольку размер спектра равен размеру исходного сигнала, то $k_1 = 0, \dots, N_1 - 1$; $k_2 = 0, \dots, N_2 - 1$; n_1 и n_2 – переменные-аргументы базисных функций. Поскольку область определения базисных функций совпадает с областью определения сигнала, то $n_1 = 0, \dots, N_1 - 1$; $n_2 = 0, \dots, N_2 - 1$. Таким образом, 2-ДПФ суть преобразование Фурье, произведенное над двумерным массивом данных.

Двумерные функции Уолша $wal(n, m, x, y)$ задаются на квадратах с интервалами отсчета $[0, 1]$ или $[-1/2, 1/2]$ по каждой из координат. В зависимости от вида упорядочения одномерных функций двумерные функции образуют соответствующий базис. На рисунке 3.3.3 показан базис функций Уолша, упорядоченный по Качмажу для $n = m = 4$, где плюс и минус соответствуют значениям 1 и -1 соответственно.

Двумерное преобразование Уолша задается выражением

$$f_{mn} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} s_{ij} \cdot wal_{ij}^{mn}, \quad m, n = 0, 1, \dots, N - 1, \quad (3.28)$$

где s_{ij} и wal_{ij}^{mn} – элементы матриц значений сигнала и двумерной функции $wal(m, n, x, y)$ соответственно.

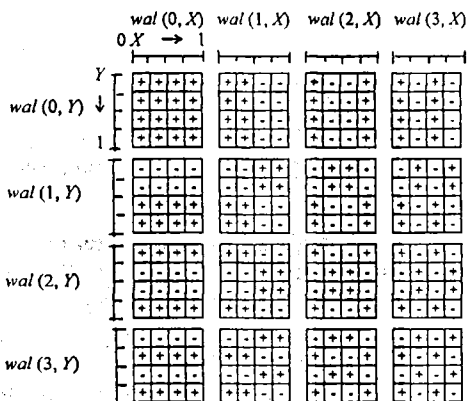


Рисунок 3.3.3 – Базис двумерных функций Уолша $wal(m, n, x, y)$ для $m, n = 0, 1, 2, 3$

Двумерные функции Хаара также образуют базис из N^2 функций . . .

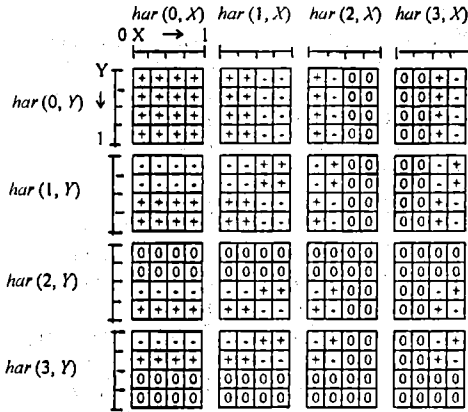


Рисунок 3.34 – Базис двумерных функций Хаара $har(m, n, x, y)$ для $m, n = 0, 1, 2, 3$

Двумерное преобразование Хаара задается выражением

$$f_{mn} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} s_{ij} \cdot har_{ij}^{mn}, \quad m, n = 0, 1, \dots, N-1, \quad (3.29)$$

где s_{ij} и har_{ij}^{mn} – элементы матриц значений сигнала и двумерной функции $har(m, n, x, y)$ соответственно.

3.4. Нейронные сети

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 развития родились на стыке наук.

Интерес к нейроинтеллекту возник еще на заре развития вычислительной техники. В 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 в этой области. Разработка искусственных разумных систем, кото-

рые соединяют преимущества биологических существ и современной вычислительной техники, создает потенциальные предпосылки для перехода к качественно новому этапу эволюции в вычислительной технике.

Первые шаги в области искусственных нейронных сетей сделали в 1943 г. В. МакКалох (W. McCulloch) и В. Питс (W. Pitts). Они показали, что при помощи пороговых нейронных элементов 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 годах Ф. Розенблатт предложил и исследовал модель нейронной сети, которую он назвал персептроном. 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. Пайперт (S. Papert) осуществили математический анализ персептрона и показали ограничения, присущие ему. В 70-е годы появился ряд работ в области ассоциативной памяти. В этом направлении продолжил исследование Т. Кохонен (T. Kohonen), который предложил модель оптимальной линейной ассоциативной памяти. В 1976 г. С. Гроссберг (S. Grossberg) разработал теорию адаптивного резонанса, которая может быть использована для построения ассоциативной памяти.

В 80-е годы происходит значительное усиление интенсивности исследований в области нейронных сетей. Дж. Хопфилд (John Hopfield) в 1982 году произвел анализ устойчивости нейронных сетей с обратными связями и предложил использовать их для решения задач оптимизации. Т. Кохонен разработал и исследовал самоорганизующиеся нейронные сети. Ряд авторов (Rumelhart, Hinton, Williams) предложил и алгоритм обратного распространения ошибки, который стал мощным средством для обучения многослойных нейронных сетей.

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

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

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

где $X = (x_1, x_2, \dots, x_n)^T$ – вектор входного сигнала; $W = (\omega_1, \omega_2, \dots, \omega_n)$ – 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.

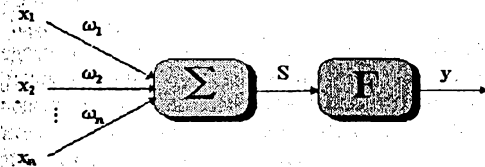


Рисунок 3.4.1 – Нейронный элемент

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

$$S = \sum_{n=1}^n \omega_n x_n = (W, X) = |W| \cdot |X| \cdot \cos \alpha,$$

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

Длины весового и входного векторов определяются через их координаты:

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

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

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 ω_j является отрицательной, то такая связь называется тормозящей. В противном случае синаптическая связь является усиливающей.

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

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

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

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные на сегодняшний день конфигурации. Если же задача не может быть сведена к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами: возможности сети возрастают с увеличением числа ячеек сети (нейронов), плотности связей между ними и числом выделенных слоев; введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети, сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тормозящих и др.), также способствует усилению мощи НС. Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

Первые успехи нейросетевого подхода связаны с решением, часто плохо формализованных, задач прогнозирования и распознавания, а также кластеризации данных. Последняя сводится к группировке входных данных по присущей им «близости». Сеть кластеризует данные на заранее неизвестное число кластеров, особенно при скачке 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, 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 *NP*-проблемы, удовлетворяющее ряду ограничений и оптимизирующее значение целевой функции (например, задача коммивояжера).

Нейросетевой подход к распознаванию. Рассмотрим НС, состоящую из n нейронных элементов распределительного слоя и m элементов выходного слоя. Для данной сети (рисунок 3.4.2) каждый нейрон распределительного слоя имеет синаптические связи w_{ij} , $i = 1, \dots, n$, $j = 1, \dots, m$, со всеми нейронами обрабатывающего слоя. В качестве нейронов выходного слоя используются элементы с некоторой функцией активации F . На вход сети подаются входные образы, т. е. векторы $\vec{x}^k = (x_1^k, \dots, x_n^k)$, $k = 1, \dots, L$.

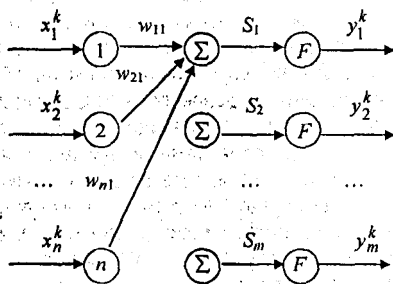


Рисунок 3.4.2 – Схема слоя НС

Выходное значение j -го нейрона сети для k -го образа определяется выражением $y_j^k = F(S_j^k)$, где $S_j^k = \sum_{i=1}^n w_{ij} x_i^k - T_j$, $j = 1, \dots, m$, $k = 1, \dots, L$. Для обучения многослойных НС применяется алгоритм обратного распространения ошибки, в основе которого лежит градиентный метод. Его недостатками для практических задач являются медленная сходимость градиентного метода с постоянным шагом обучения и сложность выбора подходящего шага для обеспечения приемлемой скорости обучения. Для ускорения работы алгоритма используют эвристические методы обучения на основе стандартного метода наискорейшего спуска, а также методы оптимизации, например, метод сопряженных градиентов и их модификации. При этом значительные усилия посвящены анализу сходимости обучения.

Существует ряд НС, позволяющих решать задачу распознавания образов с определенной степенью точности, которые отличаются друг от друга как по архитектуре, так и по принципам обучения:

- персептронные, характеризующиеся градиентными методами обучения и функционирующие аналогично персептрону: многослойный персептрон, рекуррентная и рециркуляционная НС (используются для классификации данных и аппроксимации функций);
- самоорганизующиеся НС Кохонена, характеризующиеся конкурентным методом обучения и функционирования (используется для кластеризации данных, дискретизации входного пространства и уменьшения размерности входного пространства);
- гибридные, отличающиеся применением двух подходов к обучению: конкурентного и градиентного: сети встречного распространения, сети с радиально базисной функцией активации (используются для кластеризации данных, интерполяции функций, распознавания образов, принятия решений);
- релаксационные, характеризующиеся правилом обучения Хебба и итеративным принципом функционирования. В таких сетях циркуляция информации происходит до тех пор, пока не перестанут изменяться (достигнут состояния равновесия) выходные значения НС (сети Хопфилда, Хэмминга, двунаправленная ассоциативная память).

Для повышения точности иерархического распознавания и увеличения производительности НС исследования проводятся в следующих направлениях:

- модификация правил выделения признаков за счет введения новых дополнительных инвариантов относительно искажений;
- модификация структуры и принципов послойной обработки НС;
- разработка алгоритмов обучения и самообучения НС.

Моделирование. В настоящее время все большее внимание уделяется разработке и использованию при проектировании СБИС программ моделирования процессов изготовления, так называемых виртуальных степперов. Они эмулируют перенос изображения на поверхность полупроводниковой пластины, и результаты травления анализируются с учетом таких параметров, как длина волны, коэффициент уменьшения, цифровая апертура, коэффициент когерентности, освещение и абберация.

Цель моделирования – выявить и исправить все дефекты, возникающие в процессе фотолитографии, до начала производства. Таким образом, использование виртуальных степперов сокращает стоимость разработки и цикл производства. Для разработки могут быть использованы виртуальные степперы фирм Synopsys, КБТЭМ-ОМО, Mentor Graphics,

Leica Microsystems, продукты фирмы Silvaro Data Systems (США) – Optolith, ATHENA, Virtual Wafer Fab, предназначенные для моделирования процессов фотолитографии. В данных системах используются подсистемы обработки изображений.

Литература

[1, 3, 4, 7, 10, 11, 14, 17, 19, 24-28, 30-32, 35, 38]

Контрольные вопросы

1. В чем заключается особенность схемы эксперимента Тьюринга для оценки меры интеллектуальности системы?
2. В чем заключается особенность моделей ситуационного управления при описании объекта и создании схемы управления им?
3. Каковы основные особенности построения общего решателя задач?
4. Какие перспективы имеют методы диалогового решения задач и специфика их использования?
5. Какова специфика создания и использования экспертных систем?
6. 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. Какие преимущества даёт применение процессов обучения и самообучения НС при классификации и распознавании образов?

Глава 4. ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

4.1. Основные понятия интеллектуальной информационной технологии

Понятие технологии имеет своими истоками материальное производство. Поэтому технологию 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 реализовать 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 стационарных 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 важнейшие для данного процесса сведения, но и правила вывода для выработки необходимой информации. Близкие по характеру результаты были достигнуты в ряде отраслей при использовании экспертных систем, аккумулирующих знания ведущих специалистов в конкретных отраслях с правилами получения, а иногда и объяснения рекомендуемых решений.

Технологии с такими элементами можно назвать «интеллектуальными» информационными технологиями. Роль «интеллектуальных» технологий в перспективе будет возрастать. Такой вывод напрашивается в связи с построением информационного общества во всех передовых странах. В частности, об этом говорят успехи в автоматизации проектно-конструкторских задач, автоматизации управления предприятиями, начатые разработки по созданию «электронных» правительств, управлению космическими полетами, автоматизации военных операций, прогнозированию сложных процессов, управлению транспортными средствами на Земле из космоса и др.

Особо впечатляют «интеллектуальные» информационные технологии в создании шахматных программ, которые воплотили ряд основных достижений в развитии систем «искусственного» интеллекта и суперкомпьютеров. Качество их работы росло стремительными темпами за последние примерно 40 лет. Если в 1968 году на матче между шахматными программами СССР и США демонстрировалась игра на уровне 3 разряда, то в 2008 году сильнейшие из вычислительных систем для игры в шахматы демонстрируют результаты на уровне чемпиона мира. Увлечение ученых созданием шахматных программ базируется на уверенности, что полученные здесь результаты приведут к прорыву и в других областях деятельности человека, т.е. шахматы в вычислительной технике рассматриваются как подопытный кролик в медицине. Основная причина выбора такого объекта состоит в том, что правила игры формализованы, накоплен значительный багаж сыгранных гроссмейстерами шахматных партий и окончаний, создана теория

стратегии и тактики игры, что позволило создать системы принятия решений и копирования лучших достижений в игре за счет громадной базы данных и знаний, а также быстрое действие суперкомпьютера. Это достижение явилось плодом работы многих специалистов в области шахмат, теории принятия решений, «искусственного» интеллекта, программирования, передачи и обработки данных, сетей связи, математиков, электриков и многих других.

Остановимся более подробно на дистанционных информационных технологиях в управлении 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 особенную часть (одну или несколько цифровых электронных подписей).

Названные законы открывают путь к использованию «безбумажных» систем документирования в различных отраслях деятельности и к равноправному обращению юридическом плане как обычных документов, так и «электронных». Предполагается, что при использовании сертифицированных процедур электронной цифровой подписи электронные документы будут эквивалентны обычным и будут иметь такую же доказательную силу.

Однако требуется еще пройти значительный путь, чтобы обеспечить функционирование электронных документов по описанной схеме: создать сертификационные центры и центры распространения сертифицированных средств шифрования и электронной подписи, отработать процедуры подтверждения цифровой подписи. Поэтому пока в России и Республике Беларусь полноценное функционирование описанной схемы дополнительно гарантируется договорами между хозяйствующими субъектами о взаимном признании процедур электронной подписи и границах возможного ведения хозяйственных дел на основе электронных документов.

Подписание электронных документов, кроме того, тесно связано с вопросами сохранения коммерческой тайны и часто основывается на применении аналогичных математических методов шифрования информации. Действуя в этом направлении, Национальный банк Республики Беларусь издал инструкцию о порядке передачи служебной информации ограниченного распространения в электронном виде по открытым каналам связи. Пока еще она при реализации имеет под собой договорную основу между банками РБ и Национальным банком РБ, который устанавливает в других банках свой программно-технический комплекс, который обеспечивает конфиденциальность передачи информации с клиентских рабочих мест других банков. Под управлением центрального банка проводится как установка клиентского программного обеспечения, так и генерация открытых и тайных личных ключей шифрования. Электронная цифровая подпись ос-

ществляется на клиентском рабочем месте ответственным за него работником. Чтобы избежать злоупотреблений с повторами и перехватами сообщений, пользователем указывается системное время отправки документа и выполнения наложения электронной цифровой подписи (ЭЦП) на него. Подписанное зафиксированное сообщение отсылается клиенту, от которого получается подписанное и зашифрованное сообщение (квитанция) по факту приема, проверки достоверности и целостности принятого документа.

После расшифровки квитанции проверяется достоверность ЭЦП клиента-получателя, сверяется дата и время своего отправленного сообщения с датой и временем, указанными в квитанции. Расшифрованная квитанция сохраняется в качестве ЭД получателя, подтверждающего факт и время переданной информации. Спецификой обладает процедура выполнения электронных платежей и выдачи наличных денег физическим лицам через банкоматы и кассы с использованием электронных пластиковых карт. В РБ используются пластиковые карты, имеющие применение как внутри РБ, так и в других государствах.

Абсолютное большинство составляют пластиковые карты РБ для дистанционного доступа к банковским счетам клиентов, на которые в основном вносится заработная плата, хотя допускается и внесение средств из других источников. Главными базовыми компонентами здесь являются банкоматы, инфокиоски и централизованная платежная система для обслуживания счетов клиентов, как правило, в крупных населенных пунктах РБ для любого пользователя из другого или того же населенного пункта страны. В целом по РБ находится в обращении 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 (микروпроцессора) и фотографии клиента, так как эти элементы повышают защищенность пользователя от кражи или потери карты, а также от использования ее другими лицами в магазинах и т.п.

Новая организация труда работников предприятий и государственного аппарата позволяет в ряде случаев использовать формы дистанционной работы сотрудников.

К дистанционной работе будем относить ту, которая выполняется работником предприятия с использованием информационно-вычислительной техники и средств коммуникаций всегда или время от времени вне пределов предприятия. Ведущим 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 сервисный 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 поме-

щения 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 гибкий график работы в течение дня, недели и т.п., а 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 и другого места. Этот вариант может оказаться более приемлемым для работодателя, так как можно регламентировать время контактов работника с коллегами и его руководителем.

Такого рода деятельность может иметь место и в условиях Республики Беларусь, например, выполнение переводов, программирование, сбор заказов и т.п.

Мобильные виды дистанционной работы в РБ вполне возможны для сотрудников с разъездным характером работы или сервисным обслуживанием по вызову.

Если исходить из материалов исследований Евросоюза, проведенных Фраунгоферовским институтом (ФРГ) путем опроса предприятий и органов власти, в 2000 г. имелись потребности в 875 000 дистанционных рабочих мест, из которых 500 000 относились к мобильной работе, 350 000 – к работе на альтернативной основе частично дома и на предприятии, 22 000 – к чисто дистанционной работе и 3 000 – к прочим видам работ.

Широко распространилась дистанционная работа в США 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 ряд трудностей, связанных со спецификой работы такого рода на каждом конкретном предприятии. Тем не менее, по отдельным элементам, обеспечивающим такую работу, были сделаны прикидочные расчеты, они охватывали такие аспекты, как средние стоимости программного обеспечения, компьютеров и аппаратных средств к ним, реализуемые технологии передачи данных, создание рабочей среды на дистанционном рабочем месте, обеспечение надежности информационных технологий и конфиденциальности данных и документов, борьбу с вирусами, возможные налоги и издержки производства. В зависимости от полноты набора этих элементов и загрузки коммуникаций и оборудования стоимость одного дистанционного рабочего места можно 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 жилого

площади. Проблемными вопросами являются те, которые касаются разграничения затрат работника на нужды предприятия и собственные, а также связанные с ними налоговые платежи.

Интерес к дистанционной работе обычно проявляют высококвалифицированные специалисты (программисты, инженеры, конструкторы, экономисты, переводчики), которые склонны к самостоятельной работе, требующей их знаний и способностей, инициативы. По оценкам министерства экономики ФРГ, реальное число потенциальных дистанционных работников в Евросоюзе 8 миллионов, из которых только в Германии около 2,5 миллионов.

Особая роль в дистанционной экономической работе отводится маркетинговым исследованиям и рекламе. Современные маркетинговые исследования существенно опираются на средства коммуникации, Интернет и различные региональные и отраслевые базы данных, содержащие большой объем информации практически по всем областям знаний и крупнейшим фирмам, на огромное количество рекламных материалов.

Учитывая, что маркетинговые исследования часто можно оформить как специальный заказ, они могут быть источником дистанционной работы разового плана. Аналогичные работы могут выполняться по поиску возможных путей приобретения или продажи различной «интеллектуальной» собственности (патенты, промышленные образцы, товарные знаки, произведения литературы, музыки и т.п.).

Следует обратить внимание на сложность выполнения такого рода работ из-за трудностей поиска необходимой информации. Более того, все чаще заходит речь о патентовании средств, позволяющих эффективно решать поисковые задачи в различных сетях и Интернете. В частности, фирма Amazon.com предложила однощелчковый поиск электронной мышью для выполнения закупочных операций.

Рекламные заказы в Интернете становятся сложными и дорогостоящими из-за трудностей вывода потребителя на потенциального изготовителя товара или предоставляющую услуги фирму. В связи с этим только мощные фирмы могут найти средства на рекламу, носящую глобальный характер. Специфика рекламы на базе сетевых операций содержит такой элемент, как использование медиасредств: в частности, возможность показывать короткометражные фильмы и ролики о рекламируемом объекте в действии.

Кроме того, работу многих программных и информационных продуктов можно показывать, пользуясь их демонстрационными версиями, передавая их во временное или даже постоянное пользование, чтобы потребитель смог оценить потенциальные возможности продукта для своих условий. Первые попытки таких рекламных действий уже имеются и в Республике Беларусь. По аналогии можно демонстрировать и отдельные элементы учебного процесса при рекламе дистанционных форм обучения, включая схемы проведения отдельных опытов, выполнения и оценки заданий и т.д. Такого рода деятельность тоже может осуществляться как дистанционная работа.

Начинает распространяться такой вид рекламы, когда пользователю на бесплатной основе предоставляется сам продукт, но с ограничением на количество раз или времени использования. По такого рода продуктам облегчаются и коммерческие операции по их установке у заказчика с безналичными платежами по сетям связи. Обычно продавец в таких случаях имеет программы для анализа особенностей оборудования заказчика и сообщает ему особый код. Перенести же объект в обход продавца на другое оборудование невозможно или крайне сложно.

Из перечисленных направлений развития исследований и прикладных работ следует выделить особо важные ввиду их всеобщей применимости: защита и безопасность

данных; общие аспекты создания и функционирования дистанционных рабочих мест (экономические, правовые и ИТ-вопросы); использование автоматизированных средств идентификации личности с учетом уникальных физических особенностей человека; создание средств для современной рекламы в сетях с использованием медиа-средств и демонстрационных версий; автоматизация дистанционной оплаты коммунальных услуг, получения различных справок и документов, оплаты покупок, предварительных заказов на различные услуги, управления банковскими счетами и т.п.

Эти вопросы должны стать предметом общегосударственных и отраслевых программ, так как они должны решаться системно и требуют затрат, которые не дают мгновенной окупаемости для отдельного предприятия.

4.2. Нейронные системы и нейрокомпьютеры

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

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

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

Отметим основные особенности в их работе.

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

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

В качестве важных направлений развития нейрокомпьютеров выделяют следующие четыре:

1. Решение традиционных задач для ИИ: распознавание образов (зрение, слух, обоняние); классификация (извлечение знаний из данных и т.п.). На этой основе создаются модели искусственных органов человека: искусственный глаз, ухо, нос и др.
2. Решение сложных вычислительных задач (систем линейных уравнений, молекулярное конструирование лекарств, с получением легкого распараллеливания работы алгоритмов на основе НС)

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

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

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

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

Опишем общую структурную схему абстрактного нейрокомпьютера (рис. 4.2.1).

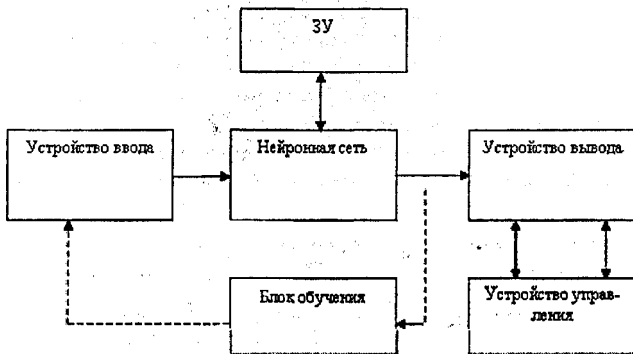


Рисунок 4.2.1 – Структурная схема абстрактного НК

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 ЗУ хранится не программа решения задачи, а программа изменений коэффициентов связи между нейронами. Устройства ввода и вывода информации выполняют обычные функции. Устройство управления служит для синхронизации работы всех структурных блоков НК при решении конкретной задачи.

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

Настройка и решаемая задача идут по итерационной схеме: при подаче очередного эталонного образа на НС выходной сигнал может отличаться от желаемого. Блок обуче-

ния 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. 4.2.2.

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

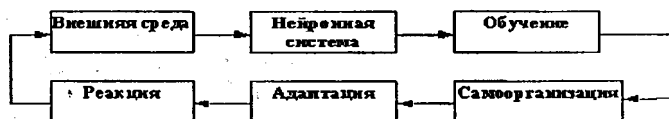


Рисунок 4.2.2 – Взаимодействие нейронной системы с внешней средой

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 нейронная сеть с прямыми связями (рис. 4.2.3).

Входной слой содержит 30х32 нейронных элемента, на которые подается преобразованное от видеокамеры изображение пути. Скрытый слой состоит из пяти, а выходной слой из 30-ти 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 на определенное число градусов. Нейронная сеть обучается при управлении автомобилем оператором. При этом оператор управляет автомобилем при движении со скоростью 9.5 км/ч, моделируя различные ситуации. Изо-

бражение от видекамеры использовалось как вход, а текущее направление руля – как желаемый выход. С целью упрощения получения обучающей выборки используется программное вращение изображения от видекамеры (рис. 4.2.4), и соответствующим образом меняется реакция нейронной системы.

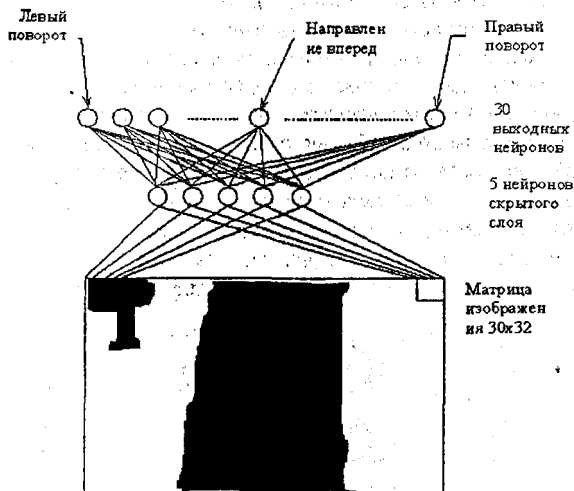


Рисунок 4.2.3 – Отображение изображения дороги на нейронную сеть

В результате была создана обучающая выборка, объем которой составил 1200 тренировочных наборов. Обучение нейронной сети проводилось с использованием трех станций «Sun-4». Время обучения методом обратного распространения ошибки составило пять минут. После обучения, как показали эксперименты, нейронная сеть может автономно управлять автомобилем.

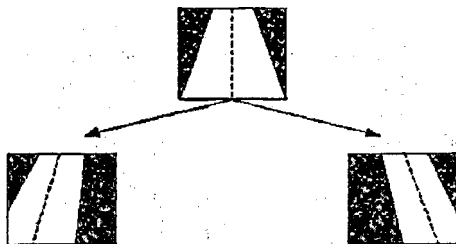


Рисунок 4.2.4 – Вращение исходного изображения от видекамеры

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 ИС. Рассмотрим решение задачи идентификации топологических слоев с помощью многослойного персептрона и неокогнитрона, которые апробированы и рекомендуются для использования.

Многослойный персептрон (рисунок 4.2.5) 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 ИС эффективно используется также для решения задач прогнозирования.

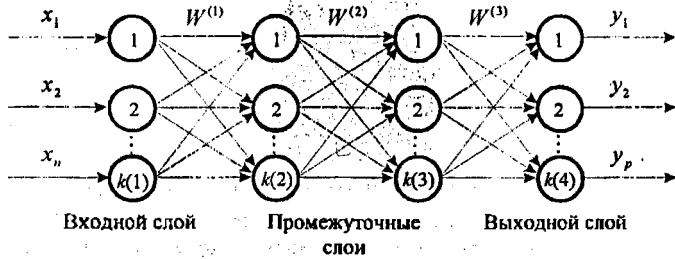


Рисунок 4.2.5 – Архитектура многослойного персептрона

Неокогнитрон (рисунок 4.2.6) позволяет выполнять 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, встречающихся у всех элементов ИС, таких, как линейные границы перепадов яркостей различной ориентации. Все подслои этого слоя состоят из нейронов с одинаковыми размерами подгрупп рецепторного поля 4×4 нейрона. Обучающие образцы для этих подслоев изображены на рисунке 4.2.7.

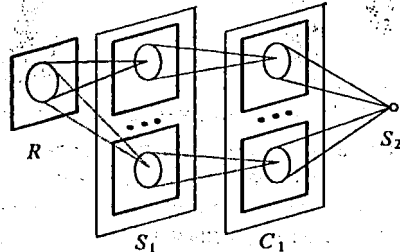


Рисунок 4.2.6 – Используемая архитектура сети



Рисунок 4.2.7 – 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 рисунке 4.2.6 горизонтальной прямоугольной скобкой). В результате получаем четыре подслоя в S_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 S_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 ИС (структурная схема представлена на рисунке 4.2.8).

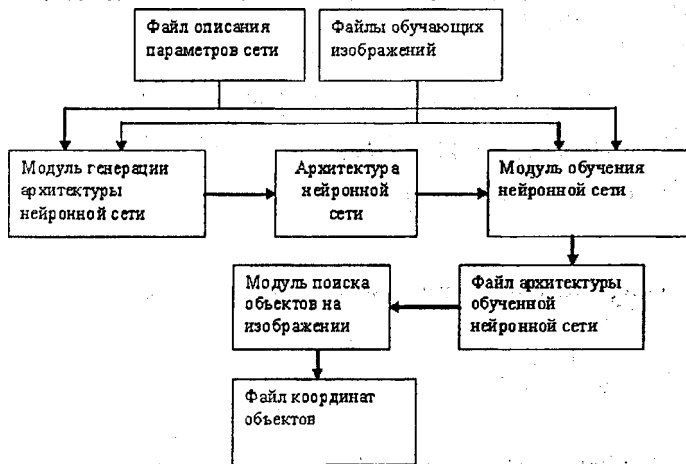


Рисунок 4.2.8 – Схема выделения топологических объектов

Обобщенная структура неокогнитрона для распознавания. Рассмотрим две матрицы

$$W = \begin{pmatrix} w_{1,1} & \dots & w_{1,Y} \\ \vdots & \ddots & \vdots \\ w_{X,1} & \dots & w_{X,Y} \end{pmatrix} \text{ и } A = \begin{pmatrix} a_{1,1} & \dots & a_{1,Y1} \\ \vdots & \ddots & \vdots \\ a_{X1,1} & \dots & a_{X1,Y1} \end{pmatrix}, \quad (4.1)$$

где W – матрица пикселей эталонного изображения; A – 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 яркостных искажений изображения, определяющий максимально допустимое расстояние между двумя пикселями изображения, которые считаются эквивалентными. Параметр R — радиус 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

$$D = \begin{pmatrix} d_{1,1} & \dots & d_{1,y} \\ \vdots & \ddots & \vdots \\ d_{x,1} & & d_{x,y} \end{pmatrix}$$

$$d_{x,y} = \begin{cases} 0, & \text{если } \min_{i,j=0,\dots,2R} (a_{x+i,y+j}) \leq w_{x,y} \leq \max_{i,j=0,\dots,2R} (a_{x+i,y+j}); \\ 0, & \text{если } |w_{x,y} - a_{x+R,y+R}| < B; \\ 1 & \text{в остальных случаях.} \end{cases} \quad (4.2)$$

Теперь функция нечеткого различия матриц A и W определяется как

$$fd(A, W) = \frac{100}{X \cdot Y} \sum_{x,y} d_{x,y} \quad (4.3)$$

Структура нейрона, реализующего описанное выше правило активации, приведена на рисунке 4.2.9, где значения $[\min_k, \max_k]$ определяются так же, как в формуле (4.2) $\min_k = \min_i (a_{k+i}), i = 0, \dots, 2R; \max_k = \max_i (a_{k+i}), i = 0, \dots, 2R, k = 1, \dots, N$, 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.

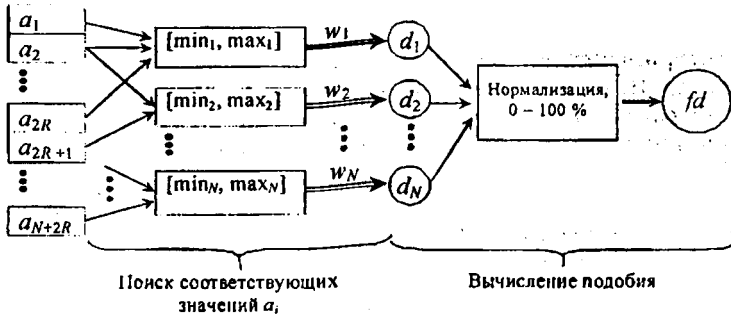


Рисунок 4.2.9 – Общая структура нечеткого нейрона

Определим функцию нечеткого динамического среднего:

$$w_{x,y}(t+1) = \begin{cases} w_{x,y}(t), & \text{если } \min_{i,j=0..2R} (a_{x+i,y+j}(t)) \leq w_{x,y}(t) \leq \max_{i,j=0..2R} (a_{x+i,y+j}(t)), \\ w_{x,y}(t), & \text{если } |w_{x,y}(t) - a_{x+R,y+R}(t)| < B, \\ \frac{w_{x,y}(t) \cdot t + a'_{x,y}}{t+1}, & \text{в остальных случаях} \end{cases} \quad (4.4)$$

где t – порядковый номер итерации усреднения; $a'_{x,y} \in \{a_{x+i,y+j}(t) | i, j = 0, \dots, 2R\}$, 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 связей неоконитрона с тем лишь отличием, что из-за упрощения структуры нейрона в ней отсутствуют обобщающие С-слои (рис. 4.2.10).

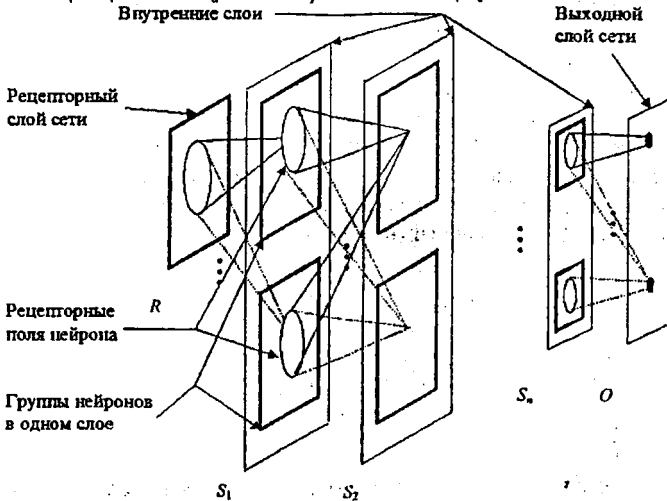


Рисунок 4.2.10 – Структура НС

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 рисунке 4.2.11 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.

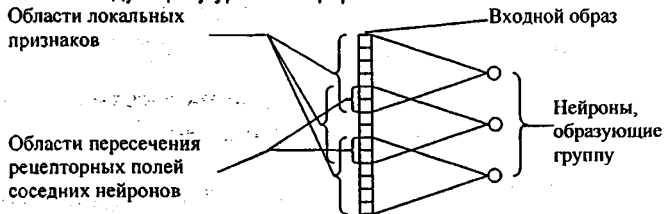


Рисунок 4.2.11 – Схема объединения нейронов в группу

Функционирование нейрона определяется соотношениями (4.2)–(4.4), в которых переменные R и B являются параметрами. Значение B зададим соотношением

$$B = \frac{1}{Y \cdot X} \cdot \sum_{x,y} \left[\frac{1}{L} \cdot \sum_l a_{x,y} - \frac{1}{L} \cdot \sum_l a_{x,y} \right], \quad (4.5)$$

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 B . Теперь R зададим как

$$R = [0,3 \times 1/F], \quad (4.6)$$

где $[\bullet]$ – операция взятия ближайшего целого.

Результаты экспериментальных исследований предложенной НС показали, что функции активации нейрона сильно зависят от выбора параметра B , а анализ результатов обучения – их зависимость от информативности обучающей выборки. Для устранения этого недостатка предлагается следующая функция активации нейрона, которая соответствует модифицированной функции корреляции

$$c = 1 + \frac{n \sum_{x,y} (w_{x,y} a'_{x,y}) - \sum_{x,y} w_{x,y} \sum_{x,y} a'_{x,y}}{\sqrt{\left(n \sum_{x,y} (w_{x,y} a'_{x,y}) - \left(\sum_{x,y} w_{x,y} \right)^2 \right) \left(n \sum_{x,y} (w_{x,y} a'_{x,y}) - \left(\sum_{x,y} a'_{x,y} \right)^2 \right)}} \quad (4.7)$$

где n – количество точек в эталонной матрице; $w_{x,y}$ – значения точек в эталонной матрице; $a'_{x,y}$ – значения соответствующих точек на анализируемом изображении,

$$a'_{x,y} = \begin{cases} 0, & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}) \right] \cap \\ & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}) \right] \neq \emptyset; \\ a_{x,y}, & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (a_{x+rx, y+ry}) \right] \cap \\ & \left[\min_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}); \max_{rx=-R, \dots, R; ry=-R, \dots, R} (w_{x+rx, y+ry}) \right] = \emptyset. \end{cases}$$

В результате применения функции (4.7) 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 перепадов, вычисляемое по формуле

$$F = \frac{F_h + F_v}{2}, \quad (4.8)$$

$$\text{где } F_h = \frac{1}{X} \left(1 + \frac{1}{Y} \sum_{x=1, y=1}^{X-1, Y} \delta(g_{x,y}^h, g_{x+1,y}^h) \right), \quad F_v = \frac{1}{Y} \left(1 + \frac{1}{X} \sum_{x=1, y=1}^{X, Y-1} \delta(g_{x,y}^v, g_{x,y+1}^v) \right).$$

$g_{x,y}^h, g_{x,y}^v$ – 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 (4.6).

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. рисунок 4.2.12 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: $F_n/F_k \leq 0,8$ или $F_k/F_n \leq 1,25$.

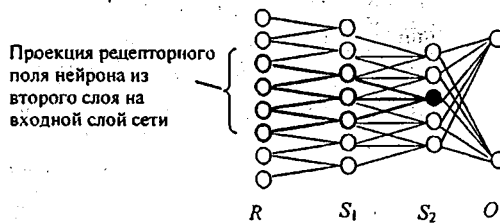


Рисунок 4.2.12 – Метод построения проекции рецепторного поля

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 рисунке 4.2.13.

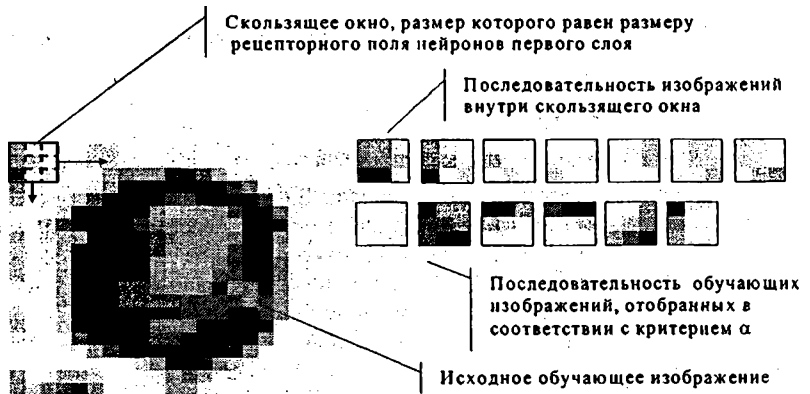


Рисунок 4.2.13 – Генерация обучающего множества слоя

Эксперименты показали, что использование функции (4.8) в качестве функции активации нейрона позволяет пропустить этап повышения информативности без ущерба качеству обучения. Такой подход сокращает время обучения на время, затрачиваемое для вычисления F_m и F_k .

4.3 Системы технического зрения

Методы оптического контроля. Важной проблемой при производстве интегральных схем (ИС) 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 состоит в обнаружении дефектов на ИС либо ФШ, возникших в ходе изготовления или подготовки данных. Она заключается в проверке качества металлизации соединений и контактных окон, элементов топологии: поликремниевых и диффузионных областей.

Один из шагов на пути к высокому уровню качества – контроль дефектов и обнаружение их источников с применением автоматического оптического контроля (АОК). Он позволяет определить до 90 % дефектов, что делает стратегию управления качеством производственного процесса эффективной. По сравнению с визуальным контролем, использование АОК 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.

АОК базируется на методах морфологических описаний, сравнения с эталоном и контроля проектных норм (КТО).

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 ИС, полученного САПР, формируется список эталонных признаков. Далее списки признаков анализируемого образца сравниваются 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 ФШ или ИС и фиксируются различия. При сравнении с эталоном возможно как непосредственное попиксельное 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 сравнения проста и эффективна с точки зрения вычислений, ограничена необходимостью иметь образец, а также требует точного совмещения изображений. Морфологический подход работает с большим числом форм, и требует постоянной 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 АОК для достижения лучшего результата (рисунок 4.3.1).

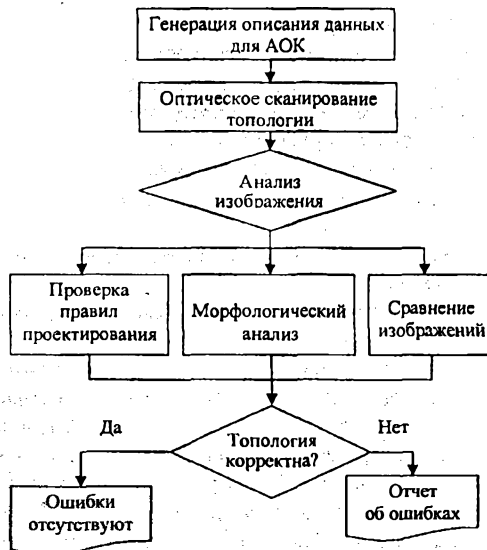


Рисунок 4.3.1 – Схема комбинирования методов АОК

Однако ни один из методов не даст ответа о наличии дефекта, когда на одном изображении одни контактные площадки (КП) соединены с другими, а третьи являются изолированными. В этом случае только АОК, дополненный электрическим тестированием, может дать ответ о наличии дефекта. Такой вид контроля получил название автоматического оптического тестирования.

Отличие автоматического оптического тестирования от АОК в том, что используется не анализ формы или другое графическое описание, а связность.

Общая функциональная схема системы оптического контроля. В настоящее время успехи в машинном зрении позволяют обеспечить автоматизированный процесс производства как электронных компонент, так и ИС в целом. Типовая СТЗ контроля выполняет следующие три основные функции (рисунок 4.3.2):

- сбор изображений (оптическое сканирование);
- анализ изображений (детектирование дефектов);
- локализация дефектов (генерация отчета об ошибках).

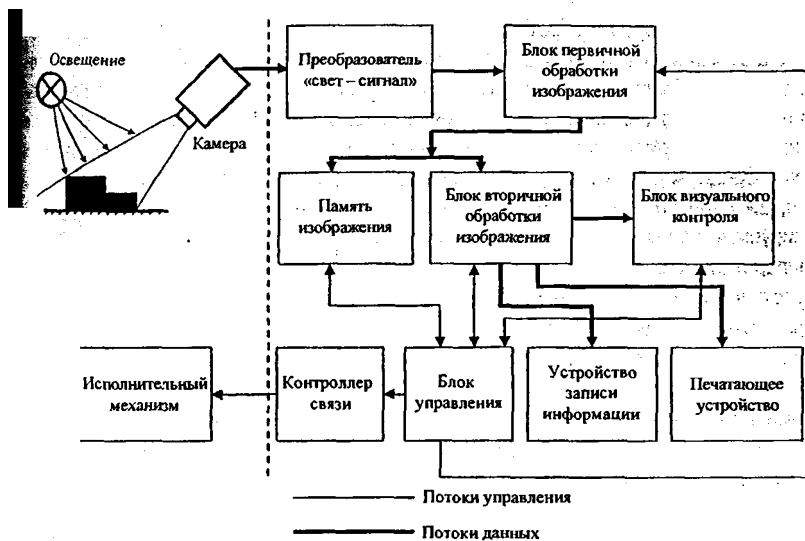


Рисунок 4.3.2 – Функциональная схема СТЗ контроля

Цель устройства первичной обработки состоит в подготовке (улучшении) изображения и частичном сжатии информации для уменьшения общего времени обработки изображения. При этом изображение представляется в виде его характерных точек и параметров (центра объекта, размера, позиции, углов, расстояний от центра до края, размера контура, текстурных мер областей и т. д.), т. е. извлекаются признаки, которые формируют описание входного изображения.

Устройство анализа изображения (вторичной обработки) служит для выделения и распознавания объекта, определения его координат и положения, т. е. формирует функционалы (меры близости, коэффициенты разложения, выборки и др.), с помощью которых по характерным признакам изображений определяет, к какому классу принадлежит объект. В состав данного устройства входит также блок обучения, обеспечивающий необходимые регулировки в процессе обучения в соответствии с выбранной программой. Анализирующее устройство и классификатор могут иметь отдельные блоки памяти. При иерархическом построении система может содержать несколько блоков анализа и классификации, используемых на различных этапах распознавания.

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, успешности выбора первичных параметров и класса решающих правил. Кроме того, возможно выполнение одними и теми же блоками нескольких функций.

Для увеличения скорости обработки используется специальное аппаратное обеспечение. Для эффективной реализации цифровой обработки сигналов хорошо подходят системные структуры благодаря параллелизму и регулярности потока данных. Для них характерно наличие однотипных модульных локально-связанных процессорных элементов, совмещенные в своей архитектуре двух принципов параллельной обработки: конвейерного и матричного. Среди матричных структур, реализующих логические преобразователи информации, широкое применение находят программируемые логические матрицы (ПЛМ), 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 актуальной разработка методов и алгоритмов декомпозиции систем булевых функций (СБФ), ПЛМ и покрытия логических схем библиотечными элементами полужаказных 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 также робастной, т. е. адаптировать себя автоматически и достигать постоянно высокой производительности, несмотря на нерегулярность в освещении и условиях фона, приспосабливаться к неопределенностям 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 слабая 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 рисунке 4.3.3 (использованием системы обозначений, предложенной Г. Бучем (она соответствует схеме, приведенной на рисунке 4.3.2). 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.)

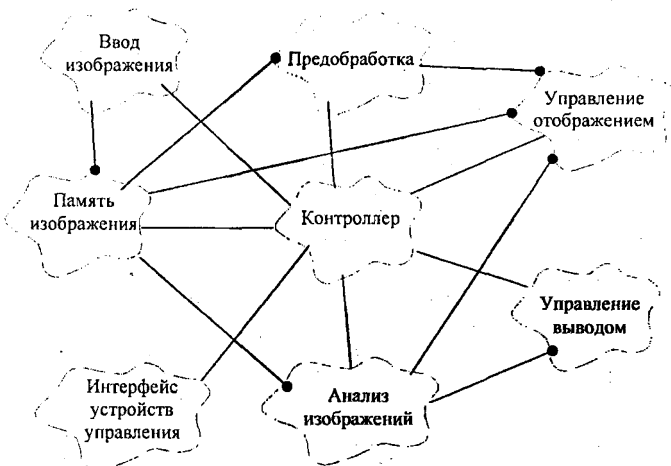


Рисунок 4.3.3 – Архитектура ПО СТЗ

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 рисунке 4.3.4. Конкретные схемы зависят от технологий, реализованных в СТЗ.

Методы проектирования программных систем. СТЗ проектирования и изготовления ИС представляет собой сложную систему. Основной путь преодоления сложности системы – это ее декомпозиция. Методы проектирования программных систем по способу декомпозиции можно разделить на три основные группы:

- метод структурного проектирования;
- метод потоков данных;
- объектно-ориентированное проектирование (ООП).

Наиболее распространенной методологией проектирования программного обеспечения (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, как правило, неэффективен в объектных и объектно-ориентированных языках программирования.

Метод потоков данных предполагает рассмотрение программной системы в качестве преобразователя входных потоков в выходные. Как и структурный метод, он 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 быстродействию.

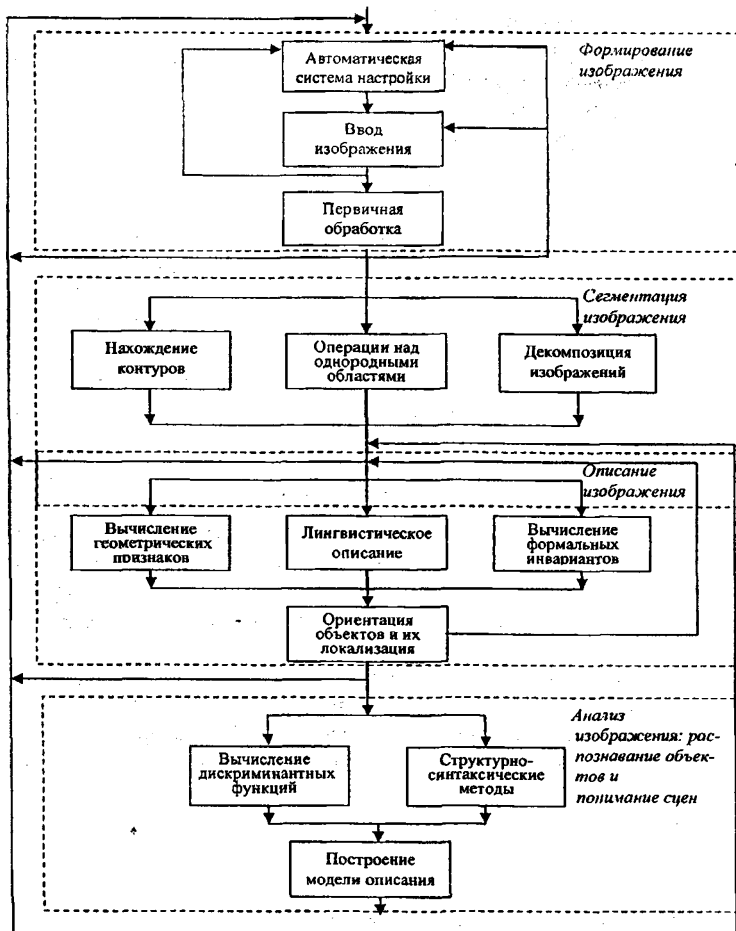


Рисунок 4.3.6 – Структурная схема обработки изображений

В основе ООП лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причем классы образуют иерархию. Объектно-ориентированный подход отражает топологию новейших языков высокого уровня, таких как Smalltalk, Object Pascal, C++, CLOS и Ada. Основными являются следующие методологии: OMT (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 нотацию UML.

Принимая во внимание сложность СТЗ обработки изображений топологии, целесообразно использовать комплексный подход для построения ПО СТЗ: приняв за основу ООП, использовать метод потоков данных для организации параллельной обработки в рамках многоагентного подхода.

4.4 Мультиагентные системы

Организация вычислений на базе многоагентной архитектуры. Общая архитектура вычислительной платформы показана на рисунке 4.4.1. 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.

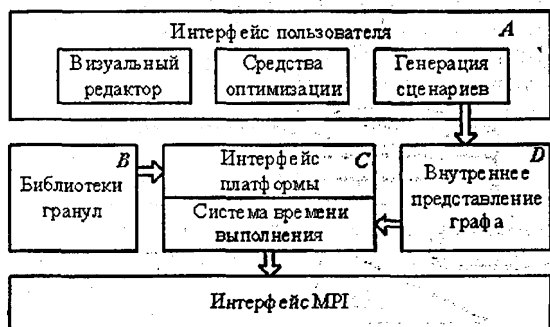


Рисунок 4.4.1 – Архитектура платформы организации параллельных вычислений

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 их 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 вычислительной системы. 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 вычислительной системы плюс один агент-координатор. Взаимодействие между агентами осуществляется с помощью передачи сообщений, реализующих простой протокол взаимодействия (рисунок 4.4.2).

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.

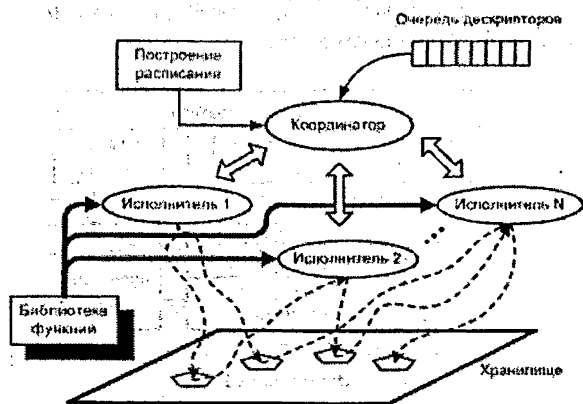


Рисунок 4.4.2 – Архитектура системы времени выполнения

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 (рисунок 6.6) 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 ИС и средства удаленного вызова процедур параллельного сервера.

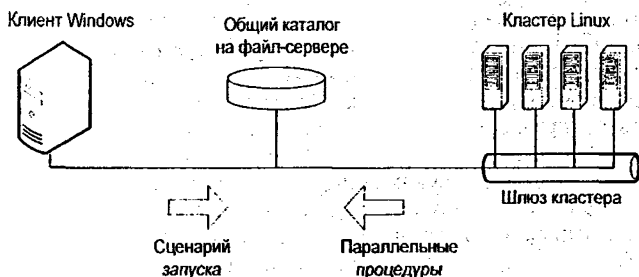


Рисунок 4.4.3 – Архитектура вычислительного комплекса

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.

4.5. Защита информации

С каждым годом повышается роль средств, обеспечивающих информационную безопасность различных технологий, с помощью компьютера. Ранее информационная безопасность ограничивалась сферами разведки, дипломатии, иногда коммерческих операций. Сейчас приходится обеспечивать информационную безопасность практически для всего взрослого населения страны. Это объясняется, прежде всего, выполнением дистанционных коммерческих операций от простейших платежей за квартиру через электронное оборудование до сложнейших расчетов через систему международных банков, а также защитой прав человека на охрану информации о нем как о конкретной личности в связи с появлением различных электронных картотек (медицинских, налоговых, правоохранительных и т.п.). Последнее привело к необходимости использовать сложнейшие информационные технологии в защите самих компьютеров и информации, хранящейся в них и передаваемой по сетям связи.

Успехи в защите информации достигаются там, где используется система, опирающаяся на широкий комплекс с взаимодействием различных средств: привычных всем замков и сейфов, охрану объекта людьми, хитроумные системы на основе лазерной техники и различных камер наблюдения и, наконец, программные методы. Во многих случаях программные методы являются основными, так как с их помощью решается главная задача – защита от несанкционированного доступа даже украденной (скопиро-

ванной) информации, а также авторизация (подпись) документов, существующих только на машинных носителях, и дистанционное управление банковским счетом.

Математической основой этих методов является теория криптографии (тайнописи). Ее истоки уходят в очень древние времена. На первых порах она часто использовала довольно простые шифры замены знаков (каждого в тексте или группы) путем подстановки другого знака методом, который был известен получателю и допускал обратную операцию. Широко известен среди них метод Цезаря, который задавал в каждом сообщении шаг сдвига для буквы в тексте: вместо истинной буквы записывалась другая, отстоящая от нее в алфавите на заданный шаг в закольцованном алфавите. Например, алфавит {A, B, C, D, E, F, R}, истинное сообщение ARFA, закодированное с шагом 3, будет – D C B D.

Секрет такого шифра в состоянии раскрыть даже школьник. Поэтому теория кодирования пошла по пути усложнения трудностей в прочтении закодированных текстов таким образом, чтобы, даже зная метод кодирования, но, не зная ключа (в нашем примере шаг сдвига), с помощью современной техники за практически разумное время (жизнь человека) нельзя было найти ключ к чтению текстов.

Компьютерный этап криптографии начался с трудов американского математика Клода Шеннона. Он выделил два общих принципа, использующихся в практическом шифровании: рассеивание (влияние одного знака на шифротекст) и перемешивание (создание трудностей в использовании статистических свойств из-за повтора знаков или характера сообщений, например, стандартные донесения с датами, наименованиями типов вооружений и их количества и т.п.). Практические трудности здесь стали заключаться в том, что надо не только затруднить раскрытие истинного сообщения, но и само шифрование сделать достаточно легким.

Таким образом, для решения этих задач пришли к идее использовать сами ЭВМ в шифровании и дешифровании текстов под управлением человека, который остался лишь носителем небольшой секретной информации (ключей).

Постепенно созрели идеи и о необходимости использования государственных стандартов шифрования, чтобы уверенно и безопасно пользоваться программами и алгоритмами, прошедшими экспертизу в специальных государственных органах. В мире первым таким стандартом стал DES (Data Encryption Standart, США). Он опирался на открытый для всех шифроалгоритм, основанный на реализации принципов рассеивания и перемешивания. В нем открытый текст, криптограмма и ключ представлялись в виде двоичных последовательностей длиной соответственно 64, 64 и 56 битов. Спустя более 30 лет после его опубликования, с помощью специального компьютера «Большой взлом» (Deep Crack) за 56 часов при переборе $18 \cdot 10^{16}$ комбинаций удалось осуществить взлом контрольной шифровки. В связи с этим на смену DES пришел AES (Advanced Encryption Standart – улучшенный стандарт шифрования), в котором для перебора всевозможных ключей (2^{256}) потребуется столько операций, которые практически нельзя осуществить даже на специальном комплексе из многих суперкомпьютеров за приемлемое время.

Аналогичного типа стандарты стали использоваться в качестве ГОСТов и в других странах (Россия и др.). Они относятся к классу систем с симметричным алгоритмом шифрования, т.е. один и тот же ключ используется для шифрования и дешифрования текста. В этом случае возникает другая сложность: как передать адресату ключ по открытым каналам связи без риска его перехвата. Поэтому дальнейшее усовершенствование криптосистем пошло в направлении создания двухключевых систем: их первый ключ

публикуется для применения всеми пользователями для шифрования данных, а для их расшифровывания имеется секретный второй ключ, который нельзя получить из первого. Суть математических преобразований для этих целей опирается на теорию функций с «лазейкой», когда $y=f(x)$ легко вычисляется, а обратная функция $x=f(y)$ невычислима без знания дополнительной информации (лазейки). Простейший пример такого типа представляет вычисление целочисленной функции $y=x \bmod n$, когда ее результатом является остаток от деления целого числа на целочисленный модуль n . Например, $y=12 \bmod 7=5$, но обратная операция отыскания x как функции от остатка 5 не выполняется однозначно (один и тот же остаток при делении на 7 могут дать 5, 12, 19, ...) Неизвестное x можно вычислить лишь при знании «лазейки», например, номера числа в ряду 5, 12, 19, ...

Покажем реализацию одноключевой системы такого типа с открытой передачей ключей на основе функций с лазейкой. Для открытого распространения ключей Диффи и Хелман предложили использовать функцию $f(x)=a^x \pmod{p}$, где p – очень большое простое число, x – целое от 1 до $(p-1)$ число, a – целое число из полуинтервала $(1 < a \leq p)$, степени которого в некотором порядке равняются 1, 2, 3, ... $(p-1)$ по модулю p . Предполагается, что всем пользователям закрытой системы известны a и p (сообщает администратор системы при регистрации). Пользователь i случайным образом выбирает целое число x_i ($1 < x_i < (p-1)$) и держит его в секрете. Потом он вычисляет $y_i = a^{x_i} \pmod{p}$ 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, где содержатся функции для шифрования текстов. (Администратор может выдавать такой справочник и время от времени его менять; важно, что их ключ (6) остается секретным и для администратора сети, так как «6» не передается по открытому каналу).

Информационная безопасность вычислительных сетей различного ранга от локальных (предприятия) до общегосударственных и корпоративных может совершенствоваться всегда вместе с развитием науки и техники. Поэтому встает вопрос о стоимости систем защиты информации. На практике используется критерий, опирающийся на оценку стоимости «взлома» системы. Если «взлом» системы обходится дороже украденной информации, то логично защиту считать достаточной, так как получение таких данных законными путями будет дешевле.

Чтобы затруднить раскрытие сообщений, иногда применяют не только методы шифрования, но и методы стеганографии, задачи которых – скрыть сам факт пересылки секретного сообщения. Для этой цели используются два типа файлов: само сообщение (первый закодированный файл) и контейнер (второй файл), задача которого скрыть содержание 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.)

Сетевые компьютерные технологии все шире начинают использоваться в построении информационного общества: развитие электронных коммерческих операций, создание электронного правительства, борьба с контрафактной продукцией, системы электронного голосования и т.д.

Проблема доверия к результатам работы различных автоматизированных систем и средств связи становится одной из ключевых. Кроме того работа различных систем становится уязвимой, если они опираются лишь на процедуру идентификации взаимодействующих сторон, так как практика борьбы со злоумышленниками в различных компьютерных сетях и злоупотреблениями в различных операциях через банки и особенно при использовании банкоматов и мобильных телефонов показали необходимость не только идентификации пользователей (пароль, PIN-код и т.п.), но и их авторизацию.

Поэтому владение современными элементами защиты оказалось для специалистов по автоматизированной обработке информации важным с двух точек зрения: изучение принципов современной защиты передаваемой и хранимой информации в сетях на основе сертифицированных систем и получения навыков работы и применения отдельных элементов защиты, как в своей практической работе, так и пропаганде широкого применения защищенных сетей среди пользователей для пояснения принципов, которые обеспечивают нераскрываемость и целостность информации благодаря шифрованию и электронной цифровой подписи при использовании сертифицированных систем.

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 аппарат теории сравнений ($x \equiv a \pmod{p}$, 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 $y = (\text{остаток})$ 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 время решения прямой и обратной задач требуется решать за время t , оцениваемое как функция 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 послал поддельное поручение. Задача конфиденциальности информации решается ее шифрованием, а ее целостность и аутентификация участников при обмене сообщениями обеспечивается протоколами (совокупностью алгоритмов, которые scrupulously должны выполнять участники обмена информацией).

Общую 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 (случайного набора битов) на передаваемый текст.

Поэтому в первую группу задач вошли алгоритмы шифрования и обеспечения выполнения защиты информации и гарантий безопасности коммерческих операций. Особенность таких алгоритмов состоит в том, что принципы обработки сообщений и схемы вычислений известны всем пользователям, а гарантом стойкости зашифрованного сообщения и электронных подписей служат секретные ключи (известны только автору) и взаимосвязанные с ними открытые ключи (известные всем пользователям системы).

Математические методы защиты информации проиллюстрируем на небольших примерах для понимания механизмов защиты информации в сетях и их пропаганды среди пользователей сетей ЭВМ.

Решим задачу построения системы шифрования Диффи и Хеллмана самостоятельно для a =(количество согласных букв в фамилии студента), p больше или равно количеству всех букв в фамилии. Подобрать a и p методом проб и ошибок, выбрать два секретных числа X_i и X_j и для связи пользователей сети i и j вычислить числа Z_i и Z_j .

Система шифрования Диффи и Хеллмана в сети с открытым распределением ключей основана на использовании функции $F(x) = a^x \bmod p$, 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 $a = 3$

$$f(1) = 3^1 \bmod 7 = 3, f(2) = 3^2 \bmod 7 = 2, f(3) = 3^3 \bmod 7 = 6, f(4) = 3^4 \bmod 7 = 4,$$

$$f(5) = 3^5 \bmod 7 = 5, f(6) = 3^6 \bmod 7 = 1$$

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 $y_i = a^{x_i} \bmod p$ 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 $Z_{ij} = (y_j)^{x_i} \bmod p$. 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, $Z_{ij} = y_j^{x_i} \bmod p = (a^{x_j})^{x_i} \bmod p = a^{x_j x_i} \bmod p$ и $Z_{ji} = a^{x_i x_j} \bmod p$.

Пример ($p = 7, a = 3, x = \{1, 2, 3, 4, 5, 6\}$)

$$x_i = 3 \text{ (секрет } i) \quad y_i = 3^3 \bmod 7 = 6$$

$$x_j = 4 \text{ (секрет } j) \quad y_j = 3^4 \bmod 7 = 4$$

$$Z_{ij} = 4^3 \bmod 7 = 1$$

$$Z_{ji} = 6^4 \bmod 7 = 1296 \bmod 7 = 1$$

Цифра 1 может означать некоторую функцию, которая используется при кодировании; страницу 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.

Построим лишённую этого недостатка двухключевую систему с использованием алгоритма RSA и выполним 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 такая ближайшая пара при подборе 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 и дешифрования. Ключи шифрования известны всем, а дешифрующий 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 простейшем случае однобуквенные). 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 $\{A,O,Y\}=\{1,2,3\}$ 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:

$$C1 = (2^7) \bmod 33 = 29$$

$$C1 = (M1^D) \bmod N : C2 = (1^7) \bmod 33 = 1$$

$$C3 = (3^7) \bmod 33 = 9$$

Зашифрованный текст получается (29, 1, 9)

Расшифровка:

$$M1^* = 29^3 \bmod 33 = 2$$

$$M1^* = (C1^D) \bmod N : M2^* = 1^3 \bmod 33 = 1$$

$$M3^* = 9^3 \bmod 33 = 3$$

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 не-секретных 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. Поэтому в него добавляется еще одно число, которое является результатом вычисления хэш-функции $h(T)$, зависящей от текста T .

Для подбора хэш-функции $h(T)$ используем секретный ключ E из задания № 2 и по зашифрованному сообщению (три буквы) вычислим $m=h(T)$ и $S=(m^E) \bmod N$. Далее, пользуясь открытым ключом D вычислим m из соотношения $(S^D)=m \bmod N$ и убедимся в его совпадении с m владельца секретного ключа (в конечном виде обязательно передаваемое $m < N$).

Описание теоретических основ метода. 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 (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. Оно очень велико по сравнению со временем полезного действия сообщения.

Покажем в упрощенном варианте проверку подписи сообщения $T=(\text{ОЛЯ})$, с выполнением его параметрами m и s . В качестве функции хеширования $h(T)$ возьмем произведение Π сумм из двух элементов каждого шифруемого знака: его номера позиции в тексте с его числовым кодированием. В нашем случае их позиции $O=1$, $\Lambda=2$, $Я=3$, а коды $\Lambda=1$, $O=2$, $Я=3$, т.е. $\Pi=(1+2)(2+1)(3+3)=54$, чтобы получить m вычисляем его так: $m = \Pi \bmod N = 54 \bmod (33) = 21$. Можно убедиться, что эта функция удовлетворяет требованиям к ней по крайней мере для любого сообщения из 3-х букв. Это обнаруживается при вычислении всех возможных 6 разнобуквенных сообщений T : (ОЛЯ)–21, (ОЯЛ)–14, (ЛОЯ)–15, ЛЯО (20), ЯОЛ (31), ЯЛО (27), т.е. нет равных $h(T)$.

Если взять любые сообщения из трех букв типа ООО, ЯОО и т.п., тоже совпадающих $h(T)$ не будет. 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 ($s = m^E \bmod N = 21^2 \bmod 33 = 21$; $m = 21 < 33$). 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 ПЗ4, 101.25-2008 для электронной подписи, в котором алгоритм RSA один из трех рекомендуемых для применения.

Проверка штрих-кода со скрытой частью представляет интерес для многих предприятий по описанному далее алгоритму. Отметим его полезные функции в автоматизированных системах управления, логистике, создании баз данных. Скрытая часть (пломбируемая или защищенная краской), например, какая-то функция от серийного но-

мера изделия $m = h(T)$, RSA – кодирование №, передаваемое в двоичной форме для серийного номера изделия.

Особенности использования штрих-кодов в борьбе с контрафактной продукцией. 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 следующим 4-м код продукта и ещё по следующим 4-м код предприятия и последняя цифра является контрольной) и дополнительно эту же информацию в буквенно-цифровой форме на разных языках и фирменные знаки, а также электронный адрес фирмы.

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 прочесть 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 EAN-13, если они не соответствуют коду, присваиваемому предприятием. Существует простой алгоритм проверки кода EAN-13.

Алгоритм проверки кода:

1. Сложить цифры, стоящие на четных местах S_r .
2. Умножить $S_r \cdot 3 = S_1$
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 (t)
6. Найти разность $P = 10 - t$

При правильном коде P должно совпасть с контрольной цифрой.

Пример. 8590721001209

1. $S_r = 5 + 0 + 2 + 0 + 1 + 0 = 8$
2. $S_1 = 8 \cdot 3 = 24$
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 не только «водяных» знаков на продукции для борьбы с контрафактом, но и создании сложных многоступенчатых систем защиты информации.

Многоступенчатость защиты информации выражается как в смене 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_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_i^+ одинаковой длины (чаще всего 64 бита). Каждый открытый блок D_i^+ путем сложения с гаммой преобразуется в гаммированный блок D_i^+ аналогичной длины готовый для шифрования. Уравнение гаммирования для каждого блока i из набора k блоков, тогда записывается так $D_i^+ = D_i^+ \oplus D_i^+$. Обратная процедура на приемном конце сводится к повторной генерации гаммы по известному для принимающего закону. Тогда расшифрованный текст легко получается по формуле: $D_i^+ = D_i^+ \oplus D_i^+$. Такой метод позволяет изменять гамму для каждого шифруемого блока случайным образом за счет генерации псевдослучайных чисел.

Защита интеллектуальной собственности также может опираться на шифры.

Для специальности АСОИ характерна защита интеллектуальной собственности в форме ноу-хау, так как пока это единственно разумный путь для защиты баз данных и программных продуктов из-за того, что эти объекты пока не патентуются.

Разновидностью нераскрытой информации являются *секреты производства (ноу-хау)*, под которыми в международной правовой практике понимаются преимущественно знания и опыт конфиденциального технического характера, имеющие коммерческую ценность. Для охраны секретов производства (ноу-хау) необходимо соблюдение всех условий, установленных в отношении охраны коммерческой тайны.

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.

Перечислим основные формы недобросовестной конкуренции:

– действия, способные вызвать смещение в отношении хозяйствующих субъектов, товаров или предпринимательской деятельности конкурентов (незаконное использование фирменного наименования, товарного знака (знака обслуживания), наименования места происхождения товара на товарах, их упаковке, на вывесках, при демонстрации экспонатов на выставках и ярмарках, в рекламных материалах, печатных изданиях и иной документации);

– копирование внешнего вида товара другого хозяйствующего субъекта, за исключением случаев, когда копирование товара или его частей (узлов, деталей) обусловлено исключительно их техническим применением;

– введение в гражданский оборот товаров другого хозяйствующего субъекта с использованием собственных средств индивидуализации товара, если иное не предусмотрено договором, заключенным между хозяйствующими субъектами.

– неправомерные утверждения при осуществлении предпринимательской деятельности, способные дискредитировать хозяйствующий субъект, товары или предпринимательскую деятельность конкурента;

– указания или утверждения при осуществлении предпринимательской деятельности, которые могут ввести в заблуждение относительно характера, свойств, способа и места изготовления, пригодности к применению или количества товаров конкурента;

– распространение хозяйствующим субъектом в любой форме и любыми способами ложных заявлений и сведений о собственном товаре в целях сокрытия несоответствия его своему назначению или предъявляемым к нему требованиям в отношении качества, потребительских и иных свойств;

– действия хозяйствующего субъекта непосредственно или через других лиц, направленные на внутреннюю дезорганизацию предпринимательской деятельности конкурента и т.п.

Анализ такого рода деятельности в сетях ЭВМ может вестись по реестрам.

Литература

[3-5, 7, 10, 12, 14, 17-19, 24-28, 30, 34, 35, 38, 40, 41]

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?

ЗАКЛЮЧЕНИЕ

Перспективы внедрения в повседневную жизнь человека различных систем «искусственного» интеллекта грандиозны:

- стремительное нарастание количества бытовой техники с программным управлением и возможностями принятия отдельных решений без непосредственных контактов с хозяевами;

- от года к году увеличивается количество услуг, связанных с повседневной жизнью человека (оплата бытовых расходов, заказ билетов, гостиниц и других услуг, выполнение действий по отношению к отсутствующему или спящему человеку, обеспечение его безопасности, компоновка и запись кинофильмов, телерепортажей, идущих одновременно в заданном порядке, для последующего просмотра с автоматическим созданием их краткого дайджеста и т.д.);

- развитие систем «электронных» правительств, обеспечивающих контакты избирателей по получению информации о постановлениях и готовящихся проектах;

- обеспечение систем безопасности в массовых мероприятиях, мгновенный анализ результатов видеонаблюдения и передвижения людей в различных видах транспорта;

- создание гибких автоматизированных производств с возможностями выполнять индивидуализацию типовых изделий под требования заказчика;

- различные формы информационного обеспечения по индивидуальным заказам и т.д.

Можно специально выделить процессы обучения, так как их совершенствование коренным образом должно повлиять на индивидуальную подготовку отдельного специалиста:

- внедрение дистанционных систем обучения с 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).

ЛИТЕРАТУРА

1. Абламейко, С.В. Обработка изображений: технология, методы, применение / С.В. Абламейко, Д.М. Лагуновский – Минск: Ин-т техн. кибернетики НАН Беларуси, 2000. – 304 с.
2. Айзерман, М.А. Логика. Автоматы. Алгоритмы / М.А. Айзерман [и др.]. – М.: Физматгиз, 1963. – 556 с.
3. Алексеенко, А.Г. Проектирование радиоэлектронной аппаратуры на микропроцессорах / А.Г. Алексеенко [и др.]. – М.: Радио и связь, 1984. – 272 с.
4. Алгоритмы и подсистемы автоматизированного логического проектирования цифровых СБИС / Авт.-сост.: А.А. Дудкин, В.А. Головки, Г.Л. Муравьев, И.В. Качков, А.Г. Мачнев, В.П. Махнист, А.В. Гладышук; под науч. ред. Р.Х. Садыхова. – Минск: Ин-т техн. кибернетики АН Беларуси, 1994. – 126 с.
5. Баричев, С.Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. – М.: Горячая линия-Телеком, 2001. – 120 с.
6. Бибило, П.Н. Основы языка VHDL / П.Н. Бибило. – М.: «Солон-Р», 2002. – 183 с.
7. Братко, И. Программирование на языке ПРОЛОГ для искусственного интеллекта (пер. с англ.) / И. Братко. – М.: Мир, 1990. – 550 с.
8. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.: БХВ-Петербург, 2002. – 608 с.
9. Глушков, В.М. Основы безбумажной информатики / В.М. Глушков. – М.: Наука Ф.М., 1982. – 552 с.
10. Голенков, В.В. Графодинамические методы и средства параллельной асинхронной переработки информации в интеллектуальных системах / В.В. Голенков. – Мн.: БГУИР, 1996. – 295 с.
11. Головки, В.А. Нейронные сети: обучение, организация и применение. Кн.4. Учебное пособие для вузов / В.А. Головки – М.: ИПРЖР, 2001. – 256 с.
12. Головки, В.А., Матюшков Л.П. Основы защиты информации и управления интеллектуальной собственностью: учебно-методический комплекс / В.А. Головки, Л.П. Матюшков. – Брест: Изд-во УО «БрГТУ», 2011. – 76 с.
13. Гэри, М. Вычислительные машины и труднорешаемые задачи: Пер. с англ. / М. Гэри, Д. Джонсон. – М.: Мир, 1982. – 416 с.
14. Дудкин, А.А. Обработка изображений в проектировании и производстве интегральных схем / А.А. Дудкин, Р.Х. Садыхов. – Минск: ОИПИ НАН Беларуси, 2008. – 270 с.
15. Закревский, А.Д. Основы логического проектирования. Кн. 1. Комбинаторные алгоритмы дискретной математики / А.Д. Закревский, Ю.В. Поттосин, Л.Д. Черемисинова. // Минск: ОИПИ НАН Беларуси, 2004. – 226 с.
16. Закревский, А.Д. Параллельные алгоритмы логического управления / А.Д. Закревский. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1999. – 202 с.
17. Искусственный интеллект. Кн.1: Системы общения и экспертные системы / Под ред. В. Попова. – М.: Радио и связь, 1990. – 420 с.
18. Кемени, Дж. Кибернетическое моделирование / Дж. Кемени, Дж. Снелл // М.: «Советское радио», 1972. – 192 с.
19. Комарцова, Л.Г. Нейрокомпьютеры: учебное пособие для вузов / Л.Г. Комарцова, А.В. Максимов – Москва: Изд. МГТУ им. Н.Э. Баумана, 2004. – 400 с.

20. Король И.А. Информационный менеджмент: учебное пособие / И.А.Король, А.С.Гринберг // Мн.: Академия управления при Президенте Республики Беларусь, 2002. – 479 с.
21. Криницкий, Н.А. Алгоритмы и роботы / Н.А. Криницкий. – М.: Радио и связь, 1983. – 168 с.
22. Криницкий, Н.А. Программирование и алгоритмические языки / Н.А. Криницкий [и др.]. – М.: Наука, 1979. – 509 с.
23. Ларионов, А.М. Вычислительные комплексы, системы и сети: учеб. пособие для вузов / А.М. Ларионов, С.А. Майоров, Г.И. Новиков. – Л.: Энергоатомиздат, 1987. – 285 с.
24. Лорьер, Ж.-Л. Системы искусственного интеллекта: Пер. с франц. / Ж.-Л. Лорьер. – М.: Мир, 1991. – 568 с.
25. Матюшков, Л.П. Основы машинной математики / Л.П. Матюшков, А.А. Лихтарович – Минск: Нар. Асвета, 1988. – 240 с.
26. Матюшков, Л.П. Перспективы информационных технологий в развитии дистанционных рабочих мест и коммерческих операций. «Вучоныя запіскі БрДУ». – Брест: БрДУ, 2006. – Т.2. – Ч.1. – С. 107-115.
27. Пархоменко, П.П. Основы технической диагностики / П.П. Пархоменко, В.С. Сагомонян. – Москва, Энергоиздат, 1981. – 349 с.
28. Пасхин, Е.Н. Автоматизированная система обучения ЭКСТЕРН / Е.Н. Пасхин, А.И. Митин. – М.: Изд-во МГУ, 1985. – 144 с.
29. Попов, Е.П. Роботы и человек / Е.П. Попов, А.С. Ющенко. – М.: Наука, 1984. – 112 с.
30. Попов, Э.В. Общение с ЭВМ на естественном языке / Е.П. Попов. – М.: Наука, 1982. – 360 с.
31. Садыхов, Р.Х. Технический сервис однородных вычислительных устройств / Р.Х. Садыхов, М.М. Татур. – Минск: Университетское, 2001. – 280 с.
32. Садыхов, Р.Х. Методы обработки сигналов в дискретных базисах / Р.Х. Садыхов, П.М. Чеголин, В.П. Шмерко. – Минск: Наука и Техника, 1987. – 296 с.
33. Столингс, У. Структурная организация и архитектура компьютерных систем. Пер. с англ. 5-е изд. / У. Столингс. – М.: «Вильямс», 2001. – 892 с.
34. Тарасов, В.Б. От многоагентных систем к интеллектуальным организациям: Философия, психология, информатика / В.Б. Тарасов. – М.: Едиториал, 2002. – 352 с.
35. Тихонов, А.Н. Вводные лекции по прикладной математике / А.Н. Тихонов, Д.П. Костомаров. – М.: Наука, 1984. – 192 с.
36. Трахтенброт, Б.А. Алгоритмы и вычислительные автоматы / Б.А. Трахтенброт. – М.: Сов. радио, 1974. – 199 с.
37. Успенский, В.А. Машина Поста / В.А. Успенский. – М.: Наука, 1979. – 95 с.
38. Форсайт, Д. Компьютерное зрение. Современный подход / Д. Форсайт, Ж. Понс; пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 928 с.
39. Шпаковский, Г.И. Организация параллельных ЭВМ и суперскалярных процессоров: учеб. пособие для вузов / Г.И. Шпаковский. – М.: Университетское, 1996. – 287 с.
40. Якимахо, А.П. Управление объектами интеллектуальной собственности в РБ / А.П. Якимахо. – Минск: Амалфея, 2005. – 472 с.
41. Янсен, Ф. Эпоха инноваций / Ф. Янсен. – М.: ИНФРА-М, 2002. – 308 с.

ГЛОССАРИЙ

- Бейсик** – популярный алгоритмический язык программирования высокого уровня, первоначально созданный для целей обучения программированию. Язык имеет относительно простой синтаксис.
- Схема алгоритма** – функционально-ориентированное графическое изображение, с помощью которого, используя текст и специальные символы, описывают последовательность шагов процесса во времени, связи рассматриваемой системы с внешней средой и ветвление процесса.
- Графический дисплей** – дисплей, обеспечивающий представление графической информации.
- Графопостроитель** – устройство вывода, предназначенное для представления данных в виде графического изображения на бумаге.
- Джойстик** – устройство для ручного управления движением курсора на экране дисплея.
- Дисплей** – устройство, обеспечивающее визуальное представление цифровой, алфавитно-цифровой и (или) графической информации на экране электронно-лучевой трубки, в плазменных панелях, на жидких кристаллах, светодиодах и т.п.
- 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.
- КОБОЛ** – язык программирования высокого уровня, ориентированный на решение экономических и коммерческих задач.
- Команда машинная** – элементарное предписание для ЭВМ, определяющее действия машины в течение некоторого отрезка времени (содержит указание операции, адреса операндов и другие служебные признаки).
- Компилятор** – обслуживающая программа, выполняющая трансляцию текста с исходного языка программирования на язык команд ЭВМ.

Локальная сеть микро – ЭВМ	– сеть микро-ЭВМ, сосредоточенная на ограниченной территории (в пределах одного или нескольких зданий) и не использующая средств связи общего назначения.
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.
Оператор	– допустимая в языке программирования синтаксическая конструкция, отражающая определенное действие в программе (присвоение значения, передачу управления и т.п.).
Операционная система	– комплекс взаимосвязанных управляющих и обслуживающих программ, обеспечивающих автоматическое управление вычислительными процессами и ресурсами ЭВМ при решении задач.
Отладка программ	– выявление ошибок в программе и доказательство ее правильности (по крайней мере, путем решения ряда контрольных задач с известными заранее результатами).
Параметр	– величина, входящая в формулы и выражения, значение которой является постоянным в пределах одной задачи.
Персональный компьютер	– небольшая по размерам и стоимости универсальная микро-ЭВМ, предназначенная для индивидуального пользователя.
Подпрограмма	– часть программы, допускающая многократное обращение к ней из различных точек программы.
Пользователь	– лицо, использующее данное вычислительное устройство для выполнения необходимых ему работ.
Программа	– последовательность инструкций, реализующих алгоритм. Программы обычно могут быть написаны: а) в машинном коде, который непосредственно выполняется процессором; б) на языке типа ассемблер; в) на языке высокого уровня.
Промышленный робот	– программируемый многофункциональный манипулятор, предназначенный для перемещения материалов, деталей, инструмента или специализированных устройств по переменным программируемым траекториям для выполнения широкого круга задач.
Процедура	– порядок выполнения ряда последовательных действий.
Процедура рекурсивная	– процедура, в описании которой содержится обращение к ней самой.
Распределенная система	– система обработки данных, в которой отдельные функции обработки выполняются независимыми устройствами.

Расшифрование	– процесс, обратный шифрованию. На основе ключа зашифрованный текст преобразуется в исходный.
Речевой ввод	– ввод информации в ЭВМ с помощью голоса. В этом случае ЭВМ реализует процесс распознавания речи (либо отдельных слов, либо фраз).
Робот	– способная действовать целенаправленная система управления и переработки информации, оборудованная датчиками восприятия информации о внешней среде и исполнительными механизмами.
Световое перо	– устройство, используемое для указания элемента изображения на экране графического дисплея.
Семантика алгоритмического языка	– толкование содержательного смысла единиц алгоритмического языка.
Сеть ЭВМ	– система соединенных между собой и обменивающихся информацией ЭВМ.
Синтаксис алгоритмического языка	– совокупность правил однозначного описания содержания алгоритмов в алгоритмическом языке.
Система	– совокупность частей, связанных общей функцией.
Система команд	– полный набор всех инструкций, допустимых в машинном языке данной ЭВМ.
Системное программное обеспечение	– набор обслуживающих программ, предназначенных для трансляции, редактирования, отладки и загрузки прикладных программ пользователя.
Структура данных	– упорядоченное множество элементов данных (вектор, матрица и др.
Технология	– наука о способах воздействия на сырье, материалы, полуфабрикаты и информацию соответствующими орудиями производства.
Транслятор	– программа, предназначенная для перевода описаний алгоритмов с одного формального языка на другой (обычно с алгоритмического на машинный).
Файл	– последовательность размещаемых на внешних ЗУ записей, рассматриваемая в процессе пересылки и обработки как единое целое.
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.
Эмуляция	– имитация функционирования одной системы средствами другой системы без потери функциональных возможностей или искажения получаемых результатов.
Язык алгоритмический	– средство точного формулирования вычислительных процессов для их последующей реализации на автоматических вычислительных машинах.
Язык ассемблера	– символический язык программирования, структура операторов которого определяется форматами команд и данными машинного языка.
Язык высокого уровня	– язык программирования, средства которого допускают описание проблемы в наглядном, легко воспринимаемом виде.

УЧЕБНОЕ ИЗДАНИЕ

Головко Владимир Адамович
Дудкин Александр Арсентьевич
Матюшков Леонид Петрович
Садыхов Рауф Хосровович

ОСНОВЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Методическое пособие по дисциплинам
«Основы искусственного интеллекта»
«Автоматизация проектирования вычислительных машин и систем»
«Архитектура вычислительных машин и систем»

Рекомендовано Советом университета для студентов специальностей:

1–40 03 01 «Искусственный интеллект»

1–40 02 01 «Вычислительные машины, системы и сети»

1–53 01 02 «Автоматизированные системы обработки информации»

Ответственный за выпуск: Матюшков Л.П.

Редактор: Боровикова Е.А.

Компьютерная вёрстка: Кармаш Е.Л.

Корректор: Никитчик Е.В.

ISBN 978-985-493-241-5



9 789854 932415

Издательство БрГТУ.

Лицензия № 02330/0549435 от 08.04.2009 г.

Подписано к печати 25.01.2013 г. Формат 60×84 1/16.

Бумага «Снегурочка». Гарнитура «Arial Narrow».

Усл. п. л. 8,6. Уч.-изд. л. 9,25. Тираж 140 экз. Заказ № 199.

Отпечатано на ризографе учреждения образования

«Брестский государственный технический университет»

224017, Брест, ул. Московская, 267.