МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Институт повышения квалификации и переподготовки руководящих работников и специалистов по информационным технологиям и радиоэлектронике
Кафедра информационных систем и технологий

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

ОСНОВЫ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Учебно-методическое пособие

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

Брест 2015

УДК 004.8 (07)
ББК 32 813
М 33

Рецензенты:

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

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

Головко В.А., Дудкин А.А., Матюшков Л.П.
М 33 Основы компьютерных технологий: учебно-методическое пособие. – Брест: Издательство УО «БрГТУ», 2015. – 180 с.

ISBN 978-985-493-324-5

В данном учебно-методическом пособии рассмотрены понятия теории алгоритмов, формальных языков, грамматик и автоматов, формальные модели алгоритмов, а также содержится информация по основам знаний в области искусственного интеллекта. Особое внимание уделяется рассмотрению базовых понятий и прикладным аспектам использования теоретических положений в современной технике и обществе.

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

УДК 004.8 (07)
ББК 32 813

ISBN 978-985-493-324-5

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

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

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

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

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

Материал книги подготовлен группой авторов: Головко В.А. написал параграфы 3.4, 4.1, 4.2 (совместно с А.А. Дудкиным), параграф 3.1 (совместно с Матюшковым Л.П.), заключение; Дудкин А.А. – главу 5, параграфы 1.2, 1,7, 5.3; 1.3 и 1.5 (совместно с Л.П. Матюшковым); Матюшков Л.П. – введение, главы 7 и 8 (совместно с Головко В.А.), параграфы 1.1, 1.4, 1.5, 2.1-2.5, 3.2, 4.5, параграфы 4.4 и 2.3 (совместно с А.А. Дудкиным).

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

Глава 1. АЛГОРИТМЫ И АВТОМАТЫ

1.1. Общие понятия

Предшественниками вычислительных машин являются автоматы – устройства, выполняющие некоторый процесс без участия человека. Появление автоматов относится к глубокой древности. Это были различные механические игрушки, автоматы для открывания дверей в храмах и т.п. Одно из отличий вычислительных устройств наших дней от традиционных средств состоит в том, что они стали машинами, пригодными для автоматизации умственной деятельности человека.

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

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

Естественно, что понятие алгоритма в теории алгоритмизации занимает центральное место. Над уточнением понятия «алгоритм» работали многие ученые. На основе этих работ развилась теория вычислимых функций, конечных и бесконечных автоматов, являющихся математическими моделями вычислительных машин.

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

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

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

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

Интуитивное определение алгоритма. Существует несколько определений термина «алгоритм». Вообще под алгоритмом часто понимается некоторое формальное предписание, действуя согласно которому, можно получить нужное решение задачи. Эта формулировка 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 искомый результат и процесс вычислений останавливается, иначе ПКУ4.
4. Если первое обозреваемое число меньше второго, то переставить их местами. ПКУ5.
5. Вычитай второе число из первого и обозревай два числа: вычитаемое и остаток. ПКУ2 (процесс повторяется по вышеназванной схеме).

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

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

По определению акад. А.Н. Колмогорова, алгоритм или алгорифм – 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.

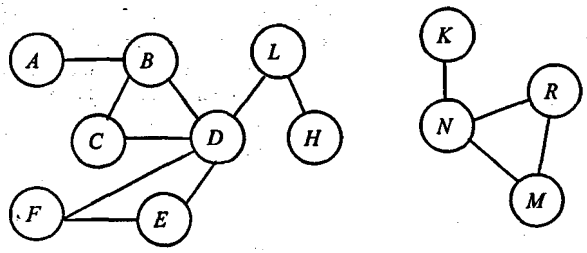


Рисунок 1.1.1 – Модель лабиринта

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

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

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

1) это площадка *Y* (цель достигнута);

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

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

4) это исходная площадка *A*;

5) отсутствие всех предыдущих признаков.

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

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

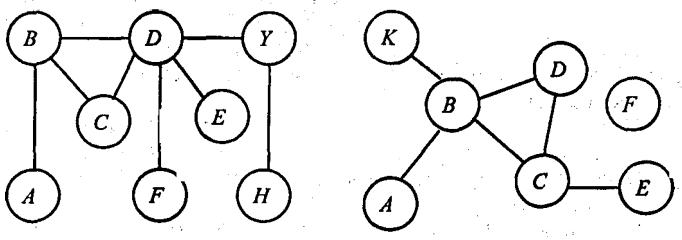
Признак	Ход
Площадка *Y*	Остановка
Петля	Возврат по последнему пройденному коридору
Имеется коридор без пометок	Движение по одному из коридоров без пометок
Площадка *A*	Остановка
Отсутствие 1-4	Возврат по пройденному коридору

Относительно рассматриваемого метода справедливы следующие утверждения:

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

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

Рассмотрим два небольших примера (рис. 1.1.2).



а) *б)*

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

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

Таблица 1.1.2 – Поиск достижимой площадки *F*

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

Таблица 1.1.3 – Недостижимость площадки *F*

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

Этот метод поиска содержит элемент произвола, которого не было в ранее рассмотренных примерах. Например, в алгоритме Евклида операции, сделанные разными вы-

числителями, 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. $a, b \in \{1, 2, ...\}$.

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

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

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

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

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

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

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

Второй тип основан на представлении алгоритма как некоторого детерминированного 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 выполняются параллельно. Так, алгоритм сложения векторов может быть представлен как последовательный, в виде: $C_i = A_i + B_i$ для $i = 1, 2, ..., n$, 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 (ВС) с малым числом процессоров может плохо работать (не давать ожидаемого ускорения) на системе с большим числом процессоров

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

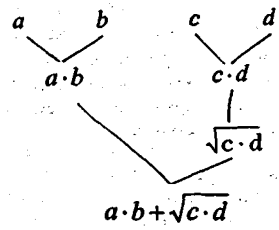


Рисунок 1.1.3 – Схема параллельного алгоритма

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 ВС с неограниченным числом вычислителей.

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

1.2. Описание алгоритмов

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

Естественно, что при описании алгоритмов нужно задавать операторы и связи между ними. Как правило, оператор представляет собой довольно простую конструкцию, и поэтому наибольшие трудности при разработке алгоритма приходятся 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 следующих за ним операторов. Естественно, что в частных случаях перестановка некоторых операторов местами допускается.

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

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

Необходимо определить входную и выходную информацию, задать области изменения 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 $X_1, X_2, ..., X_i, ..., X_n$ 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, ..., n. Отдельные операторы алгоритма будем обозначать числами 1, 2, 3, ... Они будут играть роль меток (номеров) операторов. Тогда решение задачи можно описать в виде следующего алгоритма:

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

9. ПКУ3.

10. Конец.

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

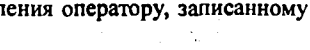
2. Если оператор зависит от параметра, то этот параметр изображается верхним индексом при его буквенном обозначении.

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.

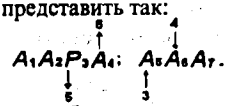
Например, операторная схема может выглядеть так:



Для удобства записи горизонтальную часть стрелки можно опускать. При этом начало стрелки отмечается малой вертикальной стрелкой, направленной от оператора

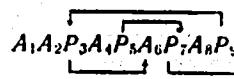
и снабженной номером оператора, которому передается управление с указанием номера, передающего управление оператора.

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 приведенный в этом параграфе алгоритм 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

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

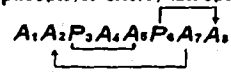
для всех дискретных значений x_i с шагом 0,1 из сегмента [−7; 5].

Так как y представлен через аналитические выражения x^2 и $x^2 + \sqrt{x}$, по которы могут быть произведены вычисления в результате подстановки в них значений x_i, т 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. 1.2.4).

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

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

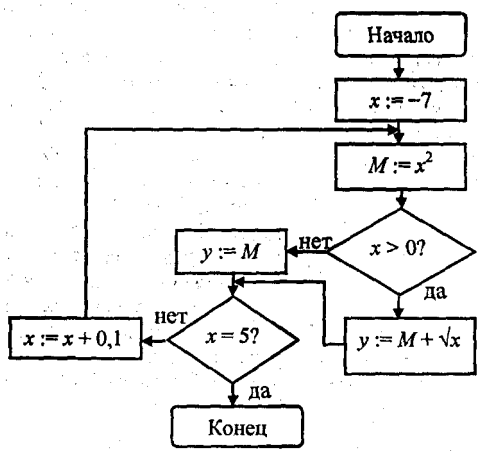


Рисунок 1.2.1 – Схема алгоритма

Основные свойства граф-схем алгоритмов следующие:

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

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

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

4. Из каждого узла P_i отходят, как правило, две стрелки, а с узла A_j – только одна стрелка.

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

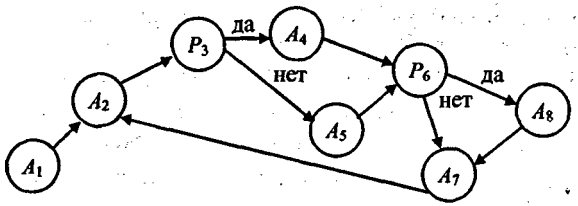


Рисунок 1.2.2 – Граф-схема алгоритма

При всем многообразии алгоритмов решения задач в них можно выделить три основных вида вычислительных процессов: линейный; ветвящийся; циклический. Линейным называется такой вычислительный процесс, при котором все этапы решения задачи выполняются в естественном порядке следования записи этих этапов. Ветвящимся называется такой вычислительный процесс, в котором выбор направления обработки информации зависит от 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 $P=\{p_1, p_2, ..., p_m\}$, 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 $J=\{j_1,...,j_n\}$. 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 $O=\{o_1, ..., o_k\}$, $k>m$. Сценарий обработки отдельного типа кадров содержит подмножество операций $O_j=\{o_{j_1}, ..., o_{j_k}\}$, $\bigcup_{j=1}^{n} O_j = O$. 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 $w_{o_i}^{j}$, представленной количеством единиц вычисления данной

операции для определенного типа кадров j. Две операции над различными кадрами, назначенные на один и тот же процессор, не могут выполняться одновременно. Два экземпляра одной и той же операции над различными кадрами также не могут выполняться в один и тот же промежуток времени.

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

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

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

W – матрица вычислительной стоимости операций $W=\bigcup W_{o_i}^{j}$; 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 включает: создание сценариев, описывающих логическую структуру программы; назначение операций обработки вершинам графа и определение параметров вызова для каждого типа данных; отображение графа сценариев на топологию вычислительного кластера. Параллельная программа представляется преобразованием $((O),(P)) \to \bigcup_{p \in P}(O_p)$, где $\forall O_k, O_p: \; O_k \neq O_p \supset O_k \bigcap O_p = \varnothing$. Каждое подмножество операций O_p размещается на выбранном узле кластера.

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

$$T_{opt} = \min\{\max T_N\}, \qquad (1.2.1)$$

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.

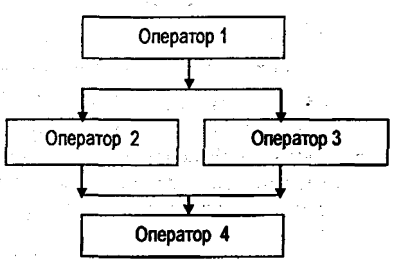


Рисунок 1.2.3 – Фрагмент параллельной ГСА

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 ВС и ее устройств.

Структура сети представляется ориентированным двудольным графом. Множество V вершин графа разбивается на два подмножества T и P, $V = T \cup P$, $T \cap P = \varnothing$. 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

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

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

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

$M_0 : P \to N_0$ – 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 $\forall p_i\in Pre(y_j)|[m(p_i)-O(p_i,t_j)\geq 0]$, 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.

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

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

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

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

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

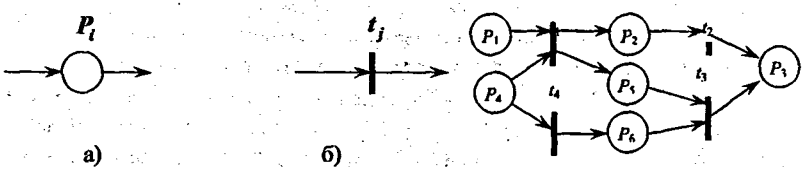


Рисунок 1.2.4 – Графическое изображение сети Петри: а) позиции; б) перехода

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

Правила срабатывания переходов 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. 1.2.5. Процесс срабатывания перходов и смены разметки называют работой сети Петри.

Рисунок 1.2.5 – 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 ВС (рис. 1.2.6), содержащую процессор (П) и устройство связи с объектом (УСО), которые могут работать параллельно, и запоминающее устройство (ЗУ).

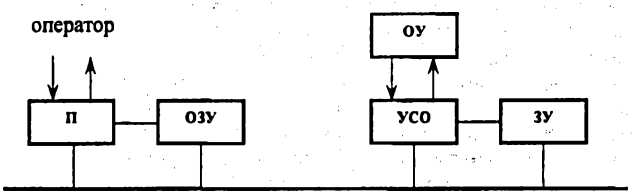


Рисунок 1.2.6 – Однопроцессорная ВС

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

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

Позиции: P_1 – П свободен; P_2, P_4 – ввод данных с ЗУ в 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.

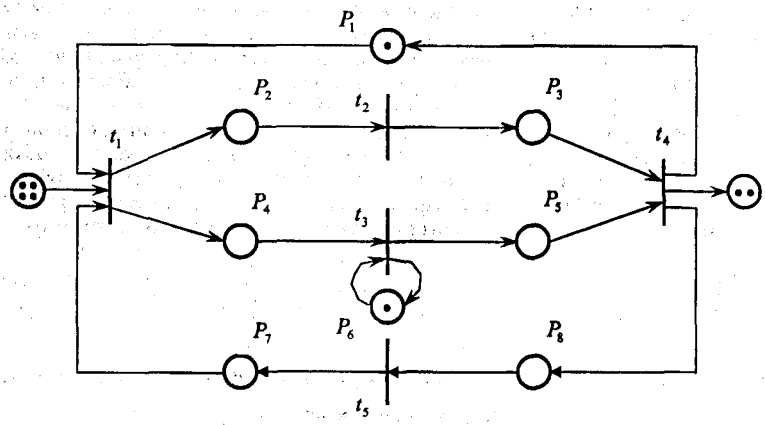


Рисунок 1.2.7 –. Модель процесса в виде сети Петри

Переход t_1 срабатывает, если есть запрос, свободный процессор и готовое к ра боте ЗУ. Возможность параллельной работы процессора и УСО отражена параллель ными ветвями с t_2 и t_3. Операция 2 (переход t_4) выполняется по завершению опера ции 1 и работы УСО. После срабатывания перехода t_4 освобождается процессор, ЗУ фиксируется метка о завершении процесса. После подготовки ЗУ (переход t_5) може отрабатываться следующий запрос оператора.

Представление алгоритма в виде диаграммы расписания. Для удобств представления параллельных алгоритмов наряду с другими способами использую также временные диаграммы выполнения операторов при заданных значениях време ни начала (окончания) их выполнения. Операторы обозначаются прямоугольниками длиной, равной времени выполнения соответствующего оператора. Для указания свя зей между прямоугольниками-операторами используются стрелки, которые соответ ствуют дугам в графе алгоритма. Когда связей становится много, из-за множеств стрелок проследить эти связи становится трудно.

Для того чтобы диаграмма оставалась наглядной при любом количестве связей однозначно отражала расписание параллельного алгоритма введем следующие до полнительные правила. Ось ординат разобьем на интервалы, каждый из которых со ответствует одному из параллельно работающих процессоров. В каждом интервал будем размещать только те прямоугольники-операторы, которые закреплены за соот ветствующим этому интервалу процессором. Операторы на диаграмме будем обозна чать в центре прямоугольников цифрой в кружке, а связи между операторами – циф рами слева и справа от этого кружка. В левой части прямоугольника-оператора будем записывать номера предшествующих по информационной связи операторов, а в пра вой части – номера операторов, следующих за данным оператором. Поскольку каж дый прямоугольник диаграммы размещается в интервале «своего» процессора, опи санный 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 напи сания параллельной программы. При этом информация о номерах связанных опера торов 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 и контроле структуры ПО. 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_i графа взвешивается числом p_i, 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:

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

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

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

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

Ниже представлена матрица достижимости графовой модели некоторой программы.

Вершины x_s, x_f соответствуют формальному началу и концу программы. Пр
ределении структурных ошибок по матрице R проверяется следующие соотношени

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

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

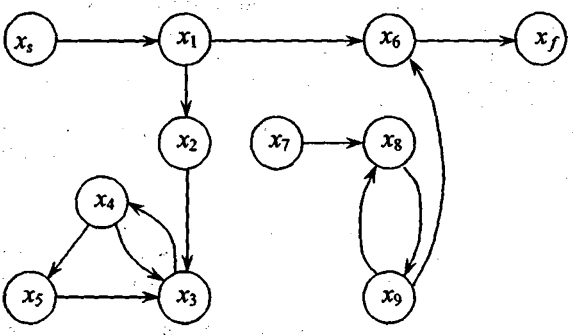


Рисунок 1.2.8 – Графовая модель программы

Анализ матрицы достижимости графа программы (рис. 1.2.8) показывает на
личие "лишних" x_7, x_8, x_9 ($r_{s7} = r_{s8} = r_{s9} = 0$) и "тупиковых" компонент x_2, x_3, x_4,
($r_{2f} = r_{3f} = r_{4f} = r_{5f} = 0$). Графовая модель программы может использоваться для оп
деления ее статистических характеристик и показателей надежности.

1.3. Методы реализации алгоритмов

Выбор численных методов реализации алгоритмов. Массовое использован
цифровых вычислительных машин в решении ряда прикладных задач привело к бу
ному развитию теории численных методов вычислений. Численный метод – это мет
приближенного или точного решения математических задач, основанный на постро
нии конечной последовательности действий над конечным множеством чисел. Ди
кретная форма представления информации в ЭВМ потребовала и создания адекват
ных методов ее обработки. Если по своей природе задача носит дискретный характе
и имеются соответствующие модели вычислений, то в этом случае метод может пра
тически сразу использоваться для машинной реализации, например, метод нахожде
ния наибольшего общего делителя двух целых положительных чисел. Проблемы мо
гут лишь возникнуть в связи с задачей представления в памяти очень больших чисе
или же из-за ограничений на время ожидания ответа. Дополнительных затрат потре
бовала переориентация методов, использующих непрерывные математические моде
ли вычислений. Непрерывные модели вычислений стали заменяться дискретными
Например, вычисление интеграла методом подстановок свелось к суммированию. Ес
ли раньше задача взятия одномерного интеграла на отрезке решалась для произволь
ных границ отрезка, то теперь она стала решаться для фиксированных границ отрезка
В первом случае точность результата определялась на базе погрешностей вычислени
по итоговой формуле, а во втором случае она еще стала зависеть и от количества то
чек деления отрезка, определяющих число элементарных трапеций при суммирова
нии. Таким образом, появилась специфическая задача автоматического выбора числ
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 $(i + 1)$ раз, пока не выполнится неравенство $|y_{i+1} - y_i| \le \epsilon$, 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) $y_4 \approx 0{,}5\,(1{,}43 + 2 : 1{,}43) \approx 1{,}41$ $|1{,}41 - 1{,}43| > 0{,}01$.

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

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:

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

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 $y_0, y_1 \ldots, y_n$, 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: $y = y_{n+1}$.

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

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

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

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

2. Функция задается конечной системой многочленов.

3. *Функция задается таблицами, графиками.*

При численном решении задач значение функции вычисляется всегда с заданной точностью. Поэтому многие способы задания функций, теоретически требующие выполнения 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 функции 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 $|f(x) - P_i(x)| < E$.

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 ВС и их компонентов на сверхбольших интегральных схемах (СБИС) является большая область поиска решений. Многие задачи являются *NP*-полными (число шагов в процессе поиска растет по экспоненте с возрастанием размерности задачи). Большинство же задач являются *NP*-сложными, то есть более сложные, чем *NP*-полные.

Все задачи решаются методами комбинаторного поиска, основанными на построении дерева поиска. Эти методы порождают ветвящиеся процессы последова-

тельного конструирования искомых решений, реализуя многошаговые переходы
начальной ситуации, отражающей лишь исходные данные решаемой задачи, чер
некоторое множество промежуточных ситуаций (в каждой из которых решается од
и та же задача – изменяются лишь данные) к заключительным ситуациям, в которь
будут представлены найденные решения. Решением является либо путь из корня д
рева в концевую вершину, либо множество дуг или множество концевых вершин.

Разработка методов решения конкретных задач связывается с конкретизацие
понятий "ситуация" и "элементарный шаг", с определением способов последовател
ного конструирования решений, с введением некоторых оптимизирующих прави
Общая схема процесса решения задач может быть представлена деревом поиска сл
дующим образом. Корню дерева поиска ставится в соответствие исходная (начальна
ситуация, а остальным вершинам – ситуации, которые могут быть достигнуты в пр
цессе поиска. Дуги соответствуют шагам решающего процесса и связывают вершин
соответствующие таким ситуациям, одна из которых преобразуется в другую посре
ством одного шага. Если из вершины выходит несколько дуг, то говорят, что ситу
ция расщепляется, а если выходит лишь одна дуга, то имеет место редуцирование с
туации. Некоторые из концевых вершин дерева поиска представляют решения.
процессе поиска требуется найти все ведущие к ним пути или один из них (любой ил
оптимальный в каком-либо смысле).

Разработка алгоритмов осуществляется по такой схеме: формулируются поняти
текущей ситуации и элементарного шага, правила редуцирования, возврата и прекра
щения поиска. Полный перебор в порожденном этими правилами дерева поиска дае
оптимальное решение. Затем формируются некоторые усеченные деревья поиска
точнее описываются правила их построения, обход которых дает некоторое прибли
женное решение.

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

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

Текущая ситуация характеризуется множеством Q допустимых миноров, образо
ванных к данному моменту, и множеством R строк матрицы, еще не включенных в ми
норы (в начальной ситуации $Q = \varnothing$ 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 отно
шение совместимости $\circ$: если минор, образованный множеством $R' \cup \{r\}$ допустимый
то $R' \circ r$. Текущую ситуацию опишем булевой матрицей $S = [Q \circ R]$, задающей отно
шение $\circ$ миноров из Q со строками из R. Для начальной ситуации матрица S не содер
жит 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 и этой строки.

Алгоритм решения этой задачи определяется следующими правилами поиска.

Правило редуцирования. Если в матрице S существует нулевой столбец, то обра
зуется новый минор и в него включается строка из R, соответствующая найденному
столбцу. Если столбцов несколько, то берется первый. В начальной ситуации выбира
ется 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 $R = \varnothing$) $|Q| < q$, 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_j.

Повышение размерности решаемых задач такого рода достигается разработкой приближенных алгоритмов, которые не гарантируют получение 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 искомый алгоритм невозможен, говорят, что данная алгоритмическая проблема неразрешима, т.е. невозможно указать единый способ решения всех задач данной серии (для отдельных задач решение возможно). Классические неразрешимые задачи на построение при помощи циркуля и линейки: удвоение куба, трисекция угла, квадратура круга. Не разрешимой является проблема выводимости для исчисления предикатов, 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. 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 алгоритма. Сложность для худшего случая это пессимистическая оценка сложности алгоритма. В алгоритмическом анализе сложность по времени и по объему памяти выражается как функция от размерности входного набора, умноженная на постоянный коэффициент. $O(f(N))$ обозначается множество всех функций $g(N)$ таких, что существуют положительные константы C и N_0 и $g(N) \leq C \cdot f(N)$ для всех $N \geq N_0$. $\Omega(f(N))$ обозначается множество всех функций $g(N)$ таких, что существуют положительные константы C и N_0 и $g(N) \geq C \cdot f(N)$ для всех $N \geq N_0$. $\Theta(f(N))$ обозначается множество всех функций $g(N)$ таких, что существуют положительные константы C_1, C_2 и N_0 и $C_1 \cdot f(N) \leq g(N) \leq C_2 \cdot f(N)$ для всех $N \geq N_0$.

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

Алгоритмы, которые имеют полиноминальную сложность по времени, являются эффективными. Полиноминально 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 ним 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 $O(2^N)$.

Класс задач, который не допускает поиска эффективных алгоритмов, известен как класс NP-полных задач. Но, может быть, возможно, найти приближенное к оптимальному решение за полиноминальное время. На практике поиск решения, прибли-

женного к оптимальному решению, может быть вполне достаточно. Алгоритм, который позволяет получить приближенное к оптимальному решение, 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 $H - M, ..., C - T$, где *H, M, ..., 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 = abcbcbab.* Заменив в слове *K* слово *H* на *M*, можно получить такие слова: *bcbcbcbab* или *abcbcbbcb* и, наоборот, заменив *M* на *H*, получим *aabcbab* или *abcabab.*

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

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

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

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

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

Пример.

$\{a, b, c, d, e\}$ – алфавит,

$ac - ca$; $abac - abacc$;
$ad - da$; $eca - ae$;
$bc - cb$; $eda - be$;
$bd - db$; $edb - be$.
} – подстановки

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

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

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

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

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

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

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

Пример.

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

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

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

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

babaac→*bbcaaac*→остановка

bcacabc→*bcacbcac*→*bcacccac*→*bcacabc*→ процесс бесконечен, т. е. остановка не наступит, так как мы получили исходное слово.

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

Задается алфавит и схема подстановок B. Для произвольного слова P просматри-ваются формулы подстановок в том порядке, в каком они заданы в схеме B, и разы-скивается формула с левой частью, входящей в P. Если такой формулы нет, то про-цесс обрывается. В противном случае берется первая из таких формул и делается под-становка ее правой части вместо первого вхождения ее левой части в P, что дает сло-во P_1, с которым поступают аналогично. Обрывается процесс в двух случаях: во-первых, когда получим такое слово P_i, что ни одна из левых частей подстановок не будет входить в него; во-вторых, когда при получении P_i нам придется применять по-следнюю формулу.

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

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

Слово P: 11 + 11 + 111.

Переработаем слово P с помощью алгоритма Маркова, отмечая знаком (------) 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 реали-зации алгоритма. Естественно, что представления любого алгоритма в такой форме достаточно, чтобы поручить его реализацию некоторому автомату. Нормальный ал-горитм Маркова можно рассматривать как стандартную форму для задания любого алгоритма. Такая форма на практике используется лишь в теоретических построени-ях. В процессе построения теории численных алгоритмов выяснилось, что любой ло-гический 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 А в пронумерованную неотрицательными целыми числами последовательность

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

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

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 Ф (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.

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

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

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

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

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

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. Абстрактные модели автоматов

Понятие автомата. Типы автоматов. Автомат – это алгоритм, определяющий :которое множество и, возможно, преобразующий его в другое множество. Нефор-альное описание автоматов выглядит следующим образом: автомат имеет входную :нту, управляющее устройство с конечной памятью для хранения номера состояния, также может иметь вспомогательную (рабочую) и выходную ленты.

Существует два типа автоматов:

1) распознаватели-автоматы без выхода, которые распознают, принадлежит ли ходная цепочка заданному множеству L;

2) преобразователи-автоматы с выходом, которые преобразуют входную цепоч-у x в цепочку y при условии, что $x \in L$.

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

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

Рабочая лента – это вспомогательное хранилище информации. Данные с нее мо-ут читаться автоматом, могут и записываться на нее.

Управляющее устройство – это программа, управляющая поведением автомата. но представляет собой конечное множество состояний вместе с отображением, опи-ывающим, как меняются состояния в соответствии с текущим входным символом, итаемым входной головкой, и текущей информацией, извлеченной с рабочей ленты. правляющее устройство также определяет направление сдвига рабочей головки и о, какую информацию записать на рабочую ленту.

Автомат работает, выполняя некоторую последовательность рабочих тактов. В ачале такта читается входной символ и исследуется информация на рабочей ленте. атем, в зависимости от прочитанной информации и текущего состояния, определя-тся 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, ко-орая 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 ля каждой конфигурации существует конечное множество возможных следующих актов, любой из которых автомат может сделать, исходя из этой конфигурации. правляющее устройство будет детерминированным, если для каждой конфигурации удет возможен только один следующий такт.

Существуют следующие типы автоматов:

1) машины Тьюринга и Поста;

2) линейно-ограниченный автомат;

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

4) конечный автомат.

Сложность рабочей ленты определяет сложность автомата. Так, например:

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

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

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

4) у конечного автомата рабочая лента отсутствует.

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

Q – множество состояний (алфавит состояний);

X – входной алфавит;

Y – выходной алфавит;

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

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

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

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

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

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

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

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

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

Закон функционирования автомата Мили задается уравнениями:

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

Закон функционирования автомата Мура задается уравнениями:

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

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 и обратно. Автомат Мура можно рассматривать как частный случай автомата Мили, имея в виду, что последовательность состояний выходов автомата Мили опережает на один такт последовательность состояний выходов автомата Мура, т.е различие между автоматами Мили и Мура состоит в том, что в автоматах Мили состояние выхода возникает одновременно с вызывающим его состоянием входа, а в автоматах Мура – с задержкой на один такт, так как. в автоматах Мура входные сигналы изменяют только состояние автомата.

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

ито автомат Мура может быть задан как входонезависимый автомат Мили. Однако это не соответствует способу функционирования автоматов Мура в соответствии с изведенным определением. В автомате Мура реализована иная временная связь между переходами из одного состояния в другое и выходом, по сравнению с автоматом Ми-ли, у которого выход, соответствующий некоторому входу и определенному состоя-нию, 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 воз-действий в предшествующие моменты времени, называется комбинационным или ав-томатом без памяти. В структурных моделях автоматы этого класса называются ком-бинационными схемами. Их функционирование описывается математической моде-лью вида $Y = f(X)$. Функция, описывающая это соотношение для одного выхода, на-зывается булевой или логической функцией. Такие функции задаются с помощью полностью или не полностью определенных таблиц, которые называются таблицами истинности и таблицами решений соответственно. Более компактной формой пред-ставления является задание булевых функций в виде булевых формул, которые в от-личие от таблиц всегда определены на всех 2^n наборах (n – число входных перемен-ных). В табл. 1.5.1 представлены примеры функций двух переменных x_1 и x_2.

Таблица 1.5.1 – Логические функции

Комбинация аргументов $x_2\,x_1$	Функция И $y = x_2 \,\&\, x_1$	Функция ИЛИ $y = x_2 + x_1$	Функция И-НЕ $y = \wedge(x_2 \,\&\, x_1)$	Функция ИЛИ-НЕ $y = \wedge(x_2 + x_1)$	Функция исключение ИЛИ $y = x_2 \oplus x_1$
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 исполь-зуют два вида ее представления.

Совершенной дизъюнктивной нормальной формой (СДНФ) называется логиче-ская сумма элементарных логических произведений, в каждое из которых аргумент или его отрицание входят 1 раз. СДНФ может быть получена из таблицы истинности следующим образом: для каждого набора аргументов, на котором функция равна «1», записывают элементарные произведения переменных, причем переменные, значение которых равно нулю, записывают с инверсией ($\wedge x$). Полученные произведения, кото-рые носят название конституент единицы, суммируют. Далее по законам алгебры ло-гики это выражение минимизируется с целью сокращения в нём как количества эле-ментарных произведений, так и некоторых переменных в них, что упрощает техниче-скую реализацию комбинационных автоматов.

Совершенной конъюнктивной нормальной формой (СКНФ) называется логиче-ское произведение элементарных сумм, в каждую из которых аргумент или его отри-цание входят один раз. СКНФ может быть получена из таблицы истинности: для каж-дого набора аргументов, на котором функция равна 0, составляют элементарную сумму, причем переменные, значение которых равно 1, записываются с отрицанием. Полученные суммы, которые носят название конституент нуля, объединяют операци-ей логического умножения, СКНФ далее по законам алгебры логики минимизируется.

Задание булевых функций 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 эффек-

тивное 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.

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

полнения нужных технологических операций в неопределенной или меняющейся гуации. Еще более интеллектуальным должен быть робот для ведения домашнего зяйства и имитации сложных механических действий человека.

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

Применительно к промышленным роботам мы рассмотрим лишь вопросы равления их рабочими органами. Абстрагируясь от конструктивных особенностей омышленных роботов, мы остановимся на математических аспектах управления и и рассмотрим вопросы программирования их действий на основе конечных авто- тов. Работа ЭВМ, робота и всякого другого реального устройства рассматривается времени. Время обычно считается непрерывным и предполагается изменяющимся настоящего к будущему. Однако при рассмотрении ряда устройств удобно вводить ображаемое дискретное время.

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

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

Будем рассматривать функционирование в дискретном времени динамических стем, описываемых координатами, каждая из которых изменяется на конечном ножестве. Всякая динамическая система получает ряд внешних возмущений: звуко- ые, электрические, световые и т. п. сигналы. Будем считать, что их число также ко- чно и каждое из возмущений задано на конечном множестве.

Динамические системы, удовлетворяющие вышеуказанным требованиям, назо- м конечными динамическими системами.

Рассмотрим множества $\{C\} = \{C_1, C_2, ..., C_k\}$ и $\{P\} = \{P_1, P_2, ..., P_r\}$, состоящие соответственно k и r символов. Набор C назовем состояниями динамической сис- мы, а набор P – входами динамической системы.

Для полного описания динамической системы необходимо задать закон ее «дви- ения», т.е. указать, как определяется состояние системы в каждый такт. Ограничим- рассмотрением одной из простейших систем.

Конечная динамическая система называется конечным автоматом, если состоя- ие системы в каждый такт однозначно определяется состоянием в предыдущий и ходом в рассматриваемый такт.

В соответствии с определением конечного автомата состояние C^i на любом так- i однозначно определяется состоянием C^{i-1} в предыдущий такт $(i-1)$ и входом P^i рассматриваемый такт i, т. е. $C^i = = F(C^{i-1}, P^i)$, где моменты времени, к которым тносятся символы C и P, обозначены верхним индексом. Очевидно, эта формула оп- еделяет собой рекуррентное соотношение, так как при известных P^1 и C^0 можно ыскать C^1, приняв $i = 1$. Далее по C^1 и P^2 можно отыскать C^2 и т. д., для всех эле- ентов множеств $\{C\}$ и $\{P\}$.

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

Фиксируя разные символы из множества $\{P\}$, получим r автономных автоматов. ля задания r автономных автоматов достаточно 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 используются таблицы.

Пусть множество входов $\{P\}$ состоит из r символов. Тогда таблица функций пе переходов состояний строится следующим образом: вычерчивается прямоугольник, со держащий $(r+1)\cdot(\kappa+1)$ клеток. В первую клетку в верхнем левом углу прямоуголь ника соответственно записываются в верхней и нижней частях ее наименования ко ординат P_i, C^{i-1}. Вправо от нее по строке поклеточно вписываются все возможны значения P_i, а по столбцу вниз – все возможные значения C. На пересечении столбца P и строки C вписываются соответствующие значения $F(C^{i-1} P^i)$ при $P^i = P$ и $C^{i-1} = C_m$. В итоге получим таблицу, которую обычно называют основной таблицей конечного автомата (табл. 1.5.2).

Таблица 1.5.2 – Основная таблица конечного автомата

C^{i-1} \ 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 $F(C_r, P_2)$ отыскивается на пересечении строки C_k и столбца P_2: $F = C_3$.

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

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

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

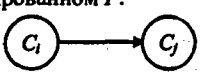


Рисунок 1.5.1 – Элемент задания конечного автомата автономными переходами

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

Таблица 1.5.3 – Основная таблица автомата A

C^{i-1} \ 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 рис. 1.5.2.

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

ия большей компактности таблицу 1.5.2. можно представить одним объединенным афом, который называется диаграммой состояний конечного автомата. Как видно рис. 1.5.3, на объединенном графе над каждой стрелкой записывается тот вход P, торый вызывает переход из исходного состояния в то, к которому направлена релка. Диаграмма состояний автомата весьма наглядна. Допустим, необходимо ус-новить, каким будет внутреннее состояние автомата A на третьем такте при входной следовательности $\{P\} = \{P_1, P_1, P_2\}$ и начальном состоянии $C^0 = C_3$. Решение зада-отыскивается простым обходом от начального состояния $C^0 = C_3$ по стрелке P_1 к едующему состоянию C_4, от него снова по стрелке P_1 к состоянию C_2, а от C_2 по релке P_2 опять возвращаемся в состояние C_2, оно и будет искомым.

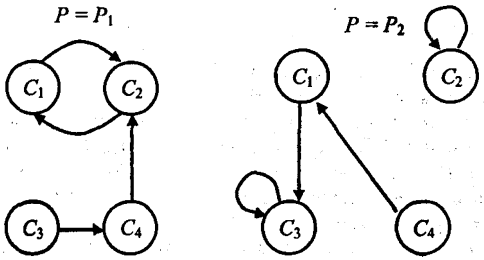


Рисунок 1.5.2 – Задание автомата диаграммой

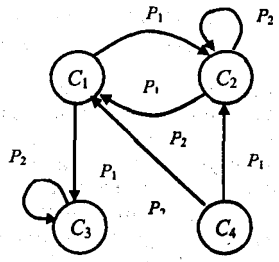


Рисунок 1.5.3 – Объединенный граф автомата

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

Графу P_c автономного автомата можно поставить во взаимно однозначное соот-тствие квадратную матрицу, состоящую из нулей и единиц. Пронумеруем все воз-жные состояния C цифрами от 1 до $к$. Составим матрицу $к \times к$, такую, что в строке i месте j будем записывать 1, если существует стрелка от i к j на графе A, а в про-вном 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 детермини-ванного автомата от каждого C должна отходить одна и только одна стрелка, и, ес-ственно, в строке m матрицы $A\,(P_c)$ найдется один элемент $a_{mi}(P_c) = 1$, а остальные дут равны нулю.

При матричном задании конечного автомата начальное состояние C^0 также зада-ся матрицей $A\,(P_0)$, состоящей из одной строки, в которой записываются $(к-1)$ ну-й, а на месте j с номером, соответствующим номеру C^0, – единица.

Пусть заданы входная последовательность $\{P\} = \{P^1, P^2, \ldots P^l\}$ и матрицей $A\,(P_0)$ чальное состояние C^0. Тогда состояние автомата на такте i устанавливается как ре-льтат вычисления произведения:

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

где $A_l(P^l)$ – 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

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

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

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

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

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

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

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

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

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

Команда на исполнительные механизмы формируется благодаря анализу входа и внутреннего состояния конечного автомата. Один и тот же вход может определять

зные команды для исполнительного механизма из-за разных состояний конечного гомата. Покажем, что конечный автомат A (задается конструктором), приведенный ижеследующей таблице, достаточен для решения поставленной задачи.

Таблица 1.5.3 – Управляющий автомат A

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 бу- г соответствовать состоянию покоя робота, C_4 – прямолинейному движению, C_3 – вороту, C_4 – захвату детали, C_5 – выгрузке детали.

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

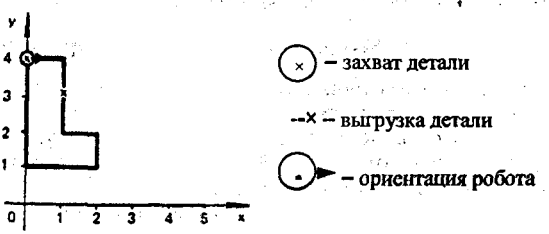


Рисунок 1.5.4 – Схема движения робота

В нашем случае $(X_0; Y_0) = (0; 4)$ и $(X_1; Y_1) = (1; 3)$. Если робот, сориентированный направлении OX, находится в состоянии покоя (C_1) в точке (0; 4) и готов к движе- ю по прямой от точки (0; 4) к точке (1; 4), то нижеследующая программа обеспечит *полнение* задачи (для наглядности будем писать в скобках соответствующие сим- лы состояния, на которое шло входное воздействие символов P_i в виде цепочки ены состояний):

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

Чтение записи происходит на базе таблицы автомата A и рисунка контура. В табли- мы находим, что при входном воздействии P_3 на состояние C_1 автомат переходит в стояние C_4 (захват детали), затем по воздействию P_2 на C_4 он начинает движение на ин шаг по контуру (состояние C_2) и приходит в точку (1; 4), по воздействию P_3 на C_4 еспечивается переход в состояние C_3 (поворот на 90° по часовой стрелке), что соответ- вует ориентации робота для движения вниз по прямой и т. д. В итоге подачи всех сиг- лов входной последовательности робот, двигаясь по контуру, снова возвратится в точ- (0; 4). Его состояние покоя C_1 во второй последовательности выделено особо (над мволом C_1 нет никакого входного символа P). Это объясняется тем, что после подачи следнего входного сигнала робот будет находиться в состоянии покоя.

Естественно, что можно начать обход контура и в другую сторону (сначала идти оси Y), но тогда изменится лишь цепочка входных воздействий. Можно пойти еще ньше и убедиться, что задача решается и для контуров другой конфигурации из из-

бранного 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.

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

1.6. Машина Поста и программирование на ней

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

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

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

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

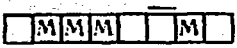
Выдающийся американский математик Э. Пост одновременно с английским математиком А. Тьюрингом в 1936 г. предложил уточненную трактовку понятия алгоритма на основе своей машины, которую позднее стали называть машиной Поста (в отличие от Тьюринга, он термин «машина» не использовал). Машина Поста менее популярна,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

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

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

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

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



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

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

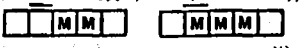
Работа машины Поста заключается в том, что головка передвигается вдоль ленты и ечатает или стирает метки в соответствии с заданной программой (инструкцией), кото-ая состоит из отдельных команд, а также распознает, имеется ли метка в секции.

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

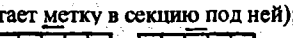
Таблица 1.5.1 – Система команд машины Поста

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

Поясним, как происходит реализация каждой команды с отображением соответ-вующих ситуаций на ленте. Состояние головки и ленты будем показывать схемати-ески рядом с кодом команды до начала ее выполнения и правее после выполнения. огда таблица команд может быть прочитана так:

1. $x \rightarrow y$ 

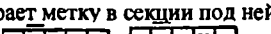
(головка сдвигается на одну секцию вправо);

2. $x \leftarrow y$ 

(головка сдвигается на одну секцию влево);

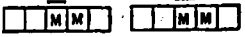
3. $x\ \mathrm{M}\ y$ 

(головка печатает метку в секцию под ней);

4. $x\ \mathrm{C}\ y$ 

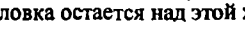
(головка стирает метку в секции под ней);

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

y_2: 

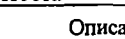
Если головка находится над пустой секцией, то управление передается команде о адресу $y1$ и головка остается над этой же секцией.

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

6. x стоп x 

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

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

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

1 М 2 

2 → 3 

3 М 4 

4 стоп 4 

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

1 С 2 

2 → 3 

3 С 4 

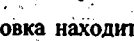
4 стоп 4 

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

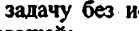
Если решать задачу без использования команды условного перехода, то программа будет следующей:

1 ← 2 

2 ← 3 

3 ← 4 

4 ← 5 

5 стоп 5 

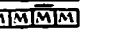
Недостаток этого решения в его конкретности, так как, зная количество меток, можно решить по данной программе только эту задачу. Используя же команду условного перехода, можно решать более общую задачу для любого количества меток: переместить головку до первой пустой секции. Программа будет иметь следующий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

1 ← 2

2 < 3, 1

3 стоп 3

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

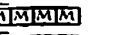
1 ← 2 

2 < 1 

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

— «» —

1 ← 2 

2 < 1 

1 ← 2 

2 < 1 

— «» —

1 ← 2 

2 > 3 

3 стоп 3 

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

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

ахождения слева или справа от головки пустой секции, если она расположена над екцией с меткой и т. д. Этим средством мы будем широко пользоваться при решении азличных задач на машине Поста, поскольку таким образом получают на ленте запись произвольного количества меток. Программой машины Поста называется конечный непустой список ее команд, обладающий следующими свойствами:

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

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

Примеры.

1) $1 \rightarrow 2$

2 М 3 – программа машины Поста 3 СТОП 3 (в соответствии с определением);

2) $2 \rightarrow 3$

3 М 1 – не является программой для 1 СТОП 1 машины Поста, так как нарушено первое условие;

3) $1 \rightarrow 2$ – не является программой для 2 М 5 машины Поста, так как нарушено торое условие (5 отсутствует среди левых номеров программы).

3 СТОП 3

Можно условиться, что в начальном состоянии головка всегда находится против ервой пустой секции левее последней левой метки на ленте, если не оговаривается особо, где она находится. Таким образом, начальное состояние машины полностью пределено состоянием ленты. Программа является той инструкцией, на основании оторой работает машина.

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

Причиной остановки машины может служить команда «стоп» или выполнение ействия, которое приводит к записи метки в секцию с уже имеющейся меткой или гиранию записи в пустой секции.

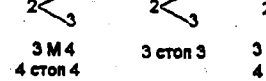
При выполнении программы могут встретиться три случая:

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

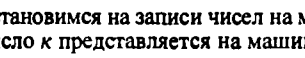
б) машина при выполнении программы дойдет до команды «стоп», и произойдет езультативная остановка, при которой программа считается выполненной;

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

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



а) 1⟶2 б) 1⟶2 в) 1⟶2

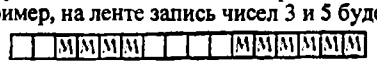


3 М 4 3 стоп 3 3 ⟵ 1

4 стоп 4 4 стоп 4

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

Число k представляется на машине Поста путем записи подряд $k+1$ метки. Между двумя числами делается интервал как минимум из одной пустой секции на ленте. апример, на ленте запись чисел 3 и 5 будет выглядеть так:



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

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

Общими для реальной и абстрактной машин являются следующие моменты:

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

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

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

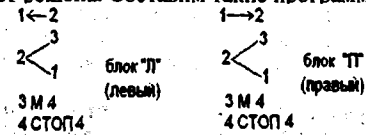
4) при составлении программы надо учитывать структуру исходных данных и расположение в памяти машины;

5) минимизация программ по числу команд.

Приведем несколько программ для сложения чисел на машине Поста.

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

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



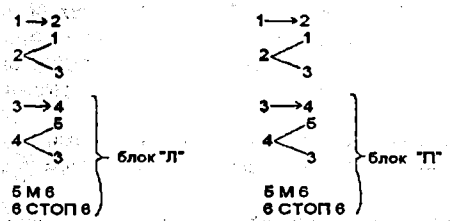
В целях проверки правильности составленных программ для машины Поста можно, выбрав какую-то общую ситуацию для постановки задачи, промоделировать, начиная с первой команды, действие программы. Например, работу блока «Л» для случая, когда головка находится над второй меткой числа, можно записать так: 1Л2 1Л2 3М4. Такая запись позволяет легко смоделировать действия машины и проверить конечный результат, глядя на исходную позицию головки и запись числа. В самом деле, запись 1Л2 отражает, что головка подвинулась влево на одну секцию и произошла передача управления команде 2; по команде 2 читается метка М под головкой, и поэтому передается управление команде 1 (запись 21); по команде 1 снова выполняется движение влево и передается управление команде 2 (вторая запись 1Л2); после второго смещения влево головка уже окажется над пустой секцией, и поэтому команда 2 передаст управление команде 3 (запись 23); по команде 3 произойдет запись метки в пустую секцию и передача управления команде 4 (запись 3М4), и наконец по команде 4 машина остановится. Такое детальное рассмотрение работы программы на достаточно общих (или в крайнем случае типичных) ситуациях называется процессом отладки программы, так как он позволяет обнаруживать ошибки как программирования, так и алгоритма, положенного в основу программы.

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

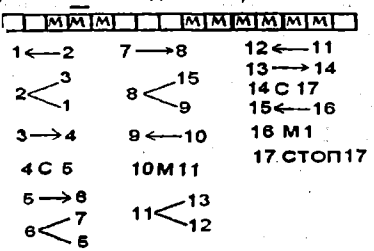
Несколько усложним задачу по прибавлению единицы к числу, расположив головку слева от числа на любое количество пустых секций. Тогда программа будет выглядеть так: 

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

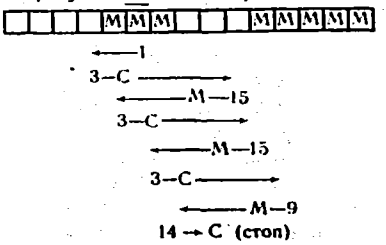
Дело в том, что написанный нами текст программы из двух команд позволяет ре-ить задачу нахождения первой метки числа на ленте, а дальше мы можем использовать товый результат, механически увеличив все номера в предыдущей программе на два:



Этот пример имеет важное значение для иллюстрации принципов «автоматиче-ого» написания программ на ЭВМ из готовых подпрограмм. Суть процесса в том, о чисто механически, без анализа предыдущих программ, получается программа с выми свойствами. Естественно, что в такой ситуации возможен перенос ошибок, пущенных другими авторами. Поэтому обычно используют программы, которые ошли многократную проверку на практике. Приведем программу для сложения це-их неотрицательных чисел a и *b* на машине Поста, когда головка находится над чис-м *a*, а число *b* находится на расстоянии *k* секций от *a* справа.



Проведем отладку этой программы для сложения чисел 2 и 4, находящихся друг от уга на расстоянии 3 секций, и пусть головка находится над второй секцией числа 2. Ре-мендованная нами выше схема записи текста будет выглядеть так: 1Л21Л23П4С5П6. евидно, что в этом случае трудно ориентироваться в движении головки вправо, влево, ранении и написании меток. Поэтому более практична будет такая схема, когда движение ловки в одну сторону отмечается стрелкой, включая записи или стирания (они помеча-ся буквой С или М под ячейкой ленты), пишется в отдельной строке слева направо (или рава налево) и в соответствии с координатой ее движения, а также пишется номер ко-нды начала строки. Убедимся в этом, записывая строчки сверху вниз с переходом на вую строку после смены направления движения головки:



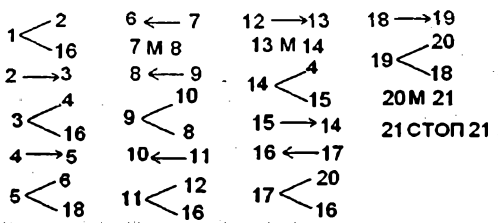
Начало стрелки содержит номер команды и находится под секцией, над которой ходится головка.

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

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

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

Принцип решения задачи сводится к следующему: проверяется, находится ли головка над одной из меток числа и, если да, то такая программа у нас уже имеется, иначе проверяется, пуста ли секция справа от головки и следующая за ней. Если обе пусты, то делается возврат головки на один шаг и ставится метка, а затем такая же операция выполняется слева, и по отмеченной дорожке головка возвращается вправо, и снова проверяются две соседние секции и т. д. до тех пор, пока головка не натолкнется на число, и тогда можно будет применить программу по прибавлению единицы, когда метка находится над числом 1. Текст этой программы (без стирания вспомогательных меток) допускает различные варианты исполнения, один из которых приводится ниже:



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

1) головка в исходном положении находится над числом;

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

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

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

Проведем некоторый сравнительный анализ машины Поста и ЭВМ.

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

Остановимся на некоторых отличиях машины Поста и ЭВМ. В машине Поста информация располагается линейно и читается подряд, а в ЭВМ можно читать информацию по адресу, что сокращает число шагов при работе ЭВМ. Для машины Поста не оговаривалось, как идет выполнение команд и перемещение головки. В частности, это может делать человек. В ЭВМ все команды выполняются автоматически,

ограмма и исходные данные вводятся до начала ее работы. ЭВМ обладает еще одм важным свойством – возможностью изменить команды в процессе работы.

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

1.7. Архитектура вычислительных систем

Классификация вычислительных систем. В узком смысле под ВС понимают вокупность технических средств, в которую входит не менее двух процессоров, язанных общей системой управления и использования общесистемных ресурсов амять, периферийные устройства, ПО и т. п.). В более широком смысле – это взаисвязанная совокупность аппаратных средств вычислительной техники и программго обеспечения, предназначенная для обработки информации.

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

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

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

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

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

Принципы построения параллельных ВС. Под параллельными вычислениями онимаются процессы обработки данных, в которых одновременно могут выполнять несколько операций компьютерной системы. Достижение параллелизма возможно лько при выполнимости следующих требований к архитектурным принципам пороения вычислительной среды:

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

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

Разновидности вычислительного параллелизма. Различают параллелизм овня инструкций, данных и задач.

Основная идея подхода, основанного на параллелизме данных, заключается в том, о одна операция выполняется сразу над всеми элементами массива данных. Различ-

ные фрагменты такого массива обрабатываются на векторном процессоре или на р
ных процессорах параллельной машины. Распределением данных между процессора
занимается ОС. Векторизация или распараллеливание в этом случае чаще всего вып
няется уже на этапе *компиляции* – перевода исходного текста программы в машинн
команды. *Роль программиста* в этом случае обычно сводится к заданию опций векто
ной или параллельной оптимизации компилятору, директив параллельной компиляц
использованию специализированных языков для параллельных вычислений.

Стиль программирования, основанный на параллелизме задач, подразумева
что вычислительная задача разбивается на несколько относительно самостоятельн
подзадач, и каждый процессор загружается своей собственной подзадачей. Компь
тер при этом представляет собой MIMD-машину, т. е., это компьютер, выполняющ
одновременно множество различных операций над множеством, вообще говоря, ра
личных и разнотипных данных.

Распределенные вычисления. Этот термин обычно используют для указан
параллельной обработки данных, при которой используется несколько обрабатыва
щих устройств, достаточно удаленных друг от друга, и в которых передача данных
линиям связи приводит к существенным временным задержкам и как результат, эф
фективная обработка данных при данном способе организации вычислений возмож
только для параллельных алгоритмов с низкой интенсивностью потоков межпроце
сорных передач данных.

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

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

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

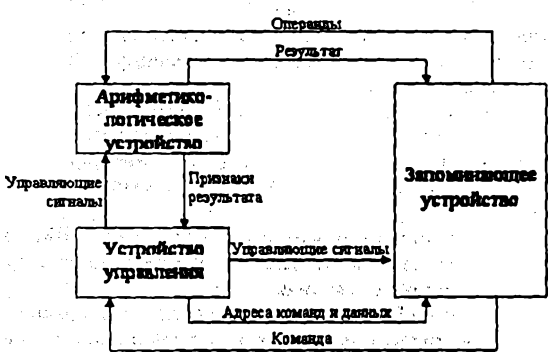


Рисунок 1.6.1 – Структура классической ЭВМ

Любое действие, выполняемое в операционном блоке, описывается некоторой
микропрограммой и реализуется за один или несколько тактов. Элементарная функ
циональная операция, выполняемая за один тактовый интервал и приводимая в дейст

ие управляющим сигналом, называется *микрооперацией*. Совокупность микроопера-ий, выполняемых в одном такте, называется *микрокомандой*. Если все такты должны меть одну и ту же длину, а именно это имеет место при работе компьютера, то она станавливается по самой продолжительной микрооперации. Микрокоманды, предна-аченные для выполнения некоторой функционально законченной последовательно-ти действий, образуют *микропрограмму*. Например, микропрограмму образует набор икрокоманд для выполнения команды умножения. Устройство управления предна-аначено для выработки управляющих сигналов, под воздействием которых происхо-ит преобразование информации в арифметико-логическом устройстве, а также опе-ации по записи и чтению информации в/из запоминающего устройства. Они реали-уются на жесткой логике, когда для каждой команды, задаваемой кодом операции, троится набор комбинационных схем, которые в нужных тактах вырабатывают не-бходимые управляющие сигналы, или микропрограммно, когда каждой команде ста-тся в соответствие микропрограмма.

Литература

[1, 9,14, 20, 21, 27, 28, 32, 35, 36, 38, 39, 40]

Контрольные вопросы

1. Чем отличается автомат от других типов устройств для выполнения процессов?
2. Назовите отличительные особенности арифметических и логических операторов.
3. Зачем разработано и используется много средств для описания алгоритмов?
4. В чем заключается особенность численных методов реализации алгоритмов?
4. В чем заключается особенность комбинаторных методов реализации алгоритмов?
5. Укажите на особенности использования ассоциативных исчислений в процес-х доказательств и общего описания функционирования ЭВМ.
6. Чем принципиально отличается нормальный алгоритм Маркова от других ал-ритмов?
7. Почему модель конечного автомата может использоваться для управления скретными процессами?
8. По каким причинам машина Поста может рассматриваться как модель ЭВМ и жет использоваться для отработки навыков по программированию?
9. Назовите основные типы архитектур вычислительных систем и их особенности.

Глава 2. ФОРМАЛЬНЫЕ ЯЗЫКИ И ОПИСАНИЕ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

2.1. Синтаксис и семантика формальных языков

Формальный и естественный языки. Интерес к применению обычного разговорного языка в диалоговом общении человека с ЭВМ и к подготовке программ объясняется целым рядом причин. Исходя из практики общения между людьми, пользователи ЭВМ, конечно, 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, и 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 смысло-

ом отношении не зависят ни от каких внешних для выбранного подмножества языка ловий. Смысл каждой фразы такого подмножества определяется только ее формой.

Описание формального языка. Для описания формального языка нужен дру- й язык. Описываемый язык называют языком-объектом в отличие от так называе- ого метаязыка (мета – вне, за пределом), применяемого для описания языка-объекта. оворя о языке, в дальнейшем, чтобы избегать путаницы, мы будем четко различать етаязыки (внешние языки) и язык-объект.

Завершенную конструкцию естественного языка обычно называют предложением. охраним этот термин и для формальных языков. Внешний язык должен обладать воз- ожностями для описания, как структуры, так и смысла предложений языка-объекта. асто его разделяют на два языка, один из которых предназначен для описания структу- предложений языка-объекта, а другой – для описания смысла. Первый язык при этом ызывают метасинтаксическим, а второй – метасемантическим. Систему правил, опреде- ющих структуру предложений языка-объекта, называют его синтаксисом; соответст- ие между предложениями языка-объекта и их значениями – его семантикой.

При описании структуры предложений формального языка обычно указывают о алфавиты букв и связей и приводят правила для построения предложений (в про- ейшем случае связей следования алфавит связей опускается). Для определенности удем считать, что каждое правило называет некоторую операцию, которую можно рименить в процессе конструирования предложений формального языка, и указыва- допустимые исходные данные для этой операции.

Следовательно, под синтаксисом формального языка понимается некоторый пе- ечень операций, для каждой из которых известна область ее определения. Кроме то- , предполагается, что синтаксис содержит формулировку некоторого условия, кото- ое выполняется для законченных конструкций формального языка и не выполняется ля конструкций, не являющихся его предложениями.

В качестве наиболее известного примера формальных языков можно назвать ко- ы. Внешний язык такого формального языка состоит из двух подъязыков, первый из торых является множеством осмысленных предложений, относящихся к интере- ующей нас области, а второй содержит описания двух правил, называемых соответ- твенно правилом кодирования и декодирования. Синтаксис формального языка со- тоит из одного правила кодирования, которое при применении к предложению пер- ого подъязыка дает предложение формального языка.

Правило декодирования определяет обратную операцию по отношению к опера- ии кодирования (которая должна допускать обратную операцию). Вообще говоря, равило декодирования задает семантику кода. Чтобы код был формальным языком, о внешний язык тоже должен быть формальным языком. Например, число семь арабскими цифрами можно следующим образом закодировать в различных системах исления: 111_2; 21_3; 13_4; 12_5; 11_6; 10_7; 7_8; 7_9 (нижний индекс указывает систему счис- ения: двоичную, троичную и т. д.). Правило декодирования тогда сведется к алго- итму 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 зработок по алгоритмическим языкам и языкам описаний различных объектов ши- око используется язык, предложенный Бэкусом. Этот метасинтаксический язык со- тоит из конечного числа предложений, называемых металингвистическими форму- ами Бэкуса. При их построении используются два универсальных метасимвола: «::= и «|», первый можно читать как «по определению есть», второй – как «или». Ос- льные метасимволы выбираются произвольно разработчиком формального языка, оторый является также и разработчиком внешнего языка.

Метасимволы являются произвольными фразами естественного языка, заклю- нными в угловые скобки (<...>). Будем называть такие метасимволы составными.

Также в формулах Бэкуса используются символы формального языка, которые легк узнать по тому, что они отличны от «:: =» и «|» и стоят вне угловых скобок. В лево части формулы Бэкуса должен стоять составной метасимвол, за ним следует зна «::=», после которого записывается правая часть формулы. Она может быть строко либо пустой, либо состоящей из конечного числа составных метасимволов и симво лов формального языка, либо конечной последовательностью таких строк, разделен ных знаками «|». По определению две формулы Бэкуса, имеющие одинаковые левы 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 фор мулу удобно, отступив от начала, а кончать, не доходя до конца строки. При перенос со строки на строку никаких дополнительных знаков ставить не следует. Смысл фор мулы 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 по мощью 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 кусственных (в частности, алгоритмических) языков. Мы остановимся более де-льно на изучении порождающих грамматик, исходя из следующих соображений:

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

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

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

Порождающей грамматикой называется упорядоченная четверка $\Gamma = \langle V, W, I, R \rangle$ оторая включает систему правил для построения последовательностей символов), е V и W – непересекающиеся конечные множества, называемые соответственно ос-вным и вспомогательным алфавитом (словарем). V обычно называют словарем рминальных, а W – словарем вспомогательных (нетерминальных) символов; I явля-ся элементом W и называется начальным символом (или выделенным вспомога-льным символом); R – набор правил вывода, или синтаксических правил, каждое из горых имеет вид $\varphi \rightarrow \Phi$. где φ и Φ – цепочки, состоящие из основных и вспомога-льных символов.

Основные символы обычно интерпретируются как слова языка, вспомогатель-е – как названия классов слов и словосочетаний, начальный символ – как символ едложения.

Синтаксические правила описывают связи между частями предложения. Приме-ние правила $\varphi \rightarrow \Phi$ к цепочке, имеющей вид $\alpha\varphi\beta$, означает преобразование ее в це-чку $\alpha\Phi\beta$ (здесь α и β – цепочки, одна из которых или даже обе могут быть пустыми).

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

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

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 $abcb$ – едложение. Другой пример вывода: A, Bc, bc.

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

В бесконтекстной или контекстно-свободной грамматике (КС – грамматике) ис-льзуется правило вывода $A \rightarrow \Phi$, где A – вспомогательный символ и Φ – непустая лочка. Языки, порождаемые такими грамматиками, называются бесконтекстными ыками.

В грамматике непосредственно составляющих (НС-грамматике) или контекстной ммматике синтаксические правила имеют вид $vAw \rightarrow v\Phi w$). В НС-грамматике каждый г вывода состоит в замене одного вхождения символа A вхождением цепочки Φ, ичем замена обусловлена наличием контекстов V и W. На каждом шаге вывода заме-ется только один символ, поэтому с каждым выводом предложения ассоциируется 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: $I \to AAB$; $A \to a$; $B \to C$; $C \to c$ (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 1 будет корнем; A, A, B – тремя вершинами второго уровня; a, a, C – тремя вершинами третьего уровня; c – вершиной четвертого уровня.

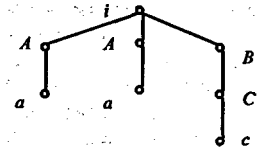


Рисунок 2.1.1 – Дерево вывода

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 и операционные системы

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 лизом классов задач, подлежащих решению. Основная цель анализа – выделить в смотриваемых задачах устойчивые и наиболее часто встречающиеся структуры ьектов языка и действия над ними, которые после их формализации и введения со-ветствующей символики объявляются элементарными объектами и базисными опе-циями языка.

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

В языке должны быть правила, позволяющие из элементарных операций и объ-ов закономерно строить более сложные. Для операций таковыми являются в ос-новном правила образования линейных структур с использованием разного рода ско-к и введением функциональных обозначений.

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

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

Основными средствами описания сложных структур в языках являются: объеди-ние объектов в линейные последовательности, образование скобочных структур и нкциональные обозначения. При этом большое место занимают операции поиска и работки данных, а также описания формата и структуры входной информации.

Проблема обработки символьной информации возникла только с появлением числительных машин. Поэтому авторы первых алгоритмических языков из рас-атриваемого класса вводили тот или иной набор операций и объектов как продукт которого умозрительного и априорного представления о закономерностях обработ-символьной информации.

Отбор изобразительных средств языка – противоречивый и сложный процесс, торый требует учета последующих стадий разработки системы программирования. став и грамматический строй алгоритмического языка всегда являются компромис-м между стремлением к богатству и разнообразию операций и объектов и стремле-ем к его простоте и лаконичности, которые позволяют обеспечить создание транс-тора и упростить его работу.

Описание языка. На этом этапе предполагается, что любой язык представляет бой некоторое множество осмысленных текстов. Описать язык – значит задать пра-ло, которое определяет, какие тексты могут быть текстами языка (синтаксис), и ус-новить, какие процессы или объекты могут описываться синтаксически допусти-ми текстами языка (семантика).

В настоящее время общепринятым методом описания синтаксиса алгоритмиче-ых языков стал метод порождающих грамматик для языков с фразовой структурой. овым языком программирования, для описания которого был применен этот метод, яется АЛГОЛ-60.

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

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

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

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

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

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

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

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 ше, не могут непосредственно выполняться на цифровой электронной вычисли-ьной машине, так как они, как правило, не представлены на языке ее команд. Язы- описания алгоритмов обычно называют языками более высокого уровня по отно-нию к машинному описанию алгоритмов на языке команд, хотя тот и другой языки яются формальными. Запись алгоритма на машинном языке сразу очень трудоемка за слишком элементарных шагов его описания, предназначенного для выполнения ЭМ. На языке высокого уровня программа, как правило, не привязана к конкретной рке ЭВМ, занимает меньше места при записи и легче обозрима человеком из-за уктурированности программы на уровне крупных блоков и операторов, а на про-емно-ориентированных языках она легко воспринимается специалистом, который знает в деталях особенности ее машинной реализации. Поэтому существуют про-аммы (трансляторы), которые выполняют функции посредника (переводчика) для еспечения выполнения алгоритма на ЭВМ.

Строго говоря, транслятор – программа, предназначенная для перевода (транс-ции) 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 шинной программы. Часто удобно использовать ступенчатую схему трансляции.

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

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

В трансляторах интерпретирующего типа процесс перевода программы на ма-инный язык совмещается с выполнением получаемой выходной программы. Эти анслятоpы более удобны для работы в диалоговом режиме. Поэтому они получили рокое распространение на современных персональных микро-ЭВМ. Пошаговое полнение алгоритма позволяет быстрее убедиться в его работоспособности и помо-т сразу скорректировать обнаруженные ошибки. Пошаговый принцип работы ино-а используется и в компилирующих трансляторах.

Процесс трансляции можно разбить на несколько составных частей: синтаксиче-ий анализ и контроль текста на входном языке (обнаружение конструкций, запи-анных с отклонениями от правил их записи); анализ описания данных и распределе-е памяти для них и для программы; получение и оптимизация текста программы, дача текста оттранслированной программы. Часть из этих функций в некоторых анслятоpах может отсутствовать, например оптимизация.

По характеру работы близки к трансляторам и различные конверторы и эмуля-ры. Их можно рассматривать как трансляторы с более узким выполнением функ-й. Программы эмуляции используются для интерпретации команд одной машины другой. К их услугам прибегают при моделировании работы проектируемых ЭВМ и же при необходимости сохранить возможность выполнения программ для ЭВМ рых моделей на новых ЭВМ. Получение текстов программ для их реализации на нкретной ЭВМ выполняется с помощью её операционной системы.

Определение операционной системы. Операционная система (ОС) – комплекс ограмм, осуществляющих управление вычислительным процессом и реализующих иболее общие алгоритмы обработки информации на данной ЭВМ. Из приведенного ределения трудно выделить функциональные границы этого комплекса программ. 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 ВС 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 рациональной

рианте с точки зрения затрат различных ресурсов на обслуживание пользователей. аким образом, очевидно, что ОС является фактически довольно сложной автоматированной системой специального назначения с рядом программ, имитирующих ятельность человека по эксплуатации и разработке программ. Создание таких ОС с ементами искусственного интеллекта за относительно короткий срок с момента пления машин, которые обслуживают одновременно нескольких пользователей, объняется тем, что их создатели-программисты должны были алгоритмизировать свой бственный труд и им не требовалось привлекать знания из других областей деяльности человека.

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

Операционные системы с каждым днем продолжают совершенствоваться, и стень их доступности для пользователей будет упрощаться. Обобщение материалов по ществующим ОС и перспективным разработкам ОС идет пока медленно. Это связа- с тем, что большинство учебников по ОС тесно привязано к конкретным маркам шин, но постепенно общие закономерности начинают выкристаллизовываться и ходить отражение в литературе по системному программированию. Следует отметь, что долгие годы работа по обеспечению «дружественного» интерфейса системми программистами-профессионалами считалась черновой и неинтересной, так как а была направлена лишь на облегчение освоения *ОС* пользователем и не касалась новных функций ОС.

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

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

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

2.3. Механизмы поиска информации

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

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

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

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

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

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

1) дескрипторные системы без грамматики;

2) дескрипторные системы с грамматикой.

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

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

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

обранными считаются те документы, номера которых оказались общими для всех скрипторов, указанных в запросе.

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

Таблица 2.4.1 – Инверсный массив

Дескрипторы / Наименование	Коды	Номера документов
ычислительная машина	101	0031. 0034. 0038. 0045
гическая обработка	102	0012. 0026. 0031. 0046
горитмический язык	105	0003. 0022. 0027. 0031
кономика	205	0031. 0168. 1342
тимизация	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 другой предмет; предмет является субъектом процесса; предмет является объектом процесса и т. п.);

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

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

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

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

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

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

$\Gamma=<V, W, I, R>$, где V – основной словарь, W – вспомогательный словарь, I ($I \in W$) – начальный символ, R – множество правил подстановки, которые представим в форме Бэкуса с использованием знаков Λ (и), $\vee$ (или), $\neg$ (нет):

<описание объекта на информационном языке> ::= <логическое произведение сообщений>

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

<сообщение> ::= <логическая сумма фраз>

<логическая сумма фраз> :: = <фраза> | <фраза> $\vee$ <логическая сумма фраз>

<фраза> ::= <понятие> $\neg$ <понятие> | <характеристика> | $\neg$ <характеристика>

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

<знак отношения> ::= > | < | >= | <= | =

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

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

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

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

Развитие поисковых систем такого типа в будущем окажет значительное влияние на развитие научно-технического прогресса и поднятие культурного уровня специалистов в различных областях деятельности. Уже современные автоматизированные хранилища позволяют в малых объемах сосредоточить колоссальные объемы информации. Например, с помощью лазерной технологии порядка тысячи страниц текста и чертежей можно записать на специальной пластинке, которая по размерам

имерно равна этикетке спичечного коробка. Телевидение, спутниковые и другие едства связи позволяют обеспечить передачу информации на большие расстояния.

В настоящее время страны выработали международные коммуникативные фор-ты, позволяющие передавать информацию по каналам связи. Например, информа-ей на машинных носителях в области изобретательской деятельности сейчас обме-ивается ряд стран. Аналогичные обмены ведутся и в области реферативной инфор-ции по публикуемым источникам.

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

В технических областях знаний запись информации на практически однозначно спринимаемых машиной подмножествах естественного языка позволяет уже сего-я получать машинные переводы, пригодные для понимания затрагиваемого вопроса ециалистом. Поэтому информационная система, снабженная соответствующими ограммами для перевода с наиболее употребительных языков, позволяет специали-ам, минуя языковой барьер, знакомиться с последней информацией в мире в кон-етных областях деятельности.

2.4. Базы данных и знаний. СУБД и языки запросов

Назначение баз данных и знаний. Большинство сложных производственных томатизированных систем различного назначения требует не только использования ецифических процессов для подготовки исходных данных для решения отдельных ач, но и обмена данными с выполняемыми одновременно другими задачами, а при дании интеллектуальных систем и подбора методов для разрешения создавшихся туаций. Это привело к развитию систем, близких по духу к операционным, но ющим целевое назначение – автоматизацию подготовки, обработки, хранения, щиты и актуализации данных, включая такие интеллектуальные процессы, как вы-р методов, и оценку путей решения текущей задачи.

Эта ситуация в своем разрешении начала свой путь с формирования понятий ба-данных и знаний. Идея базы данных имеет в своей основе подход в решении ряда ач, когда исходные данные и другие промежуточные результаты хранятся отдель-от исполняемых программ и с помощью заранее описанных механизмов управле-я извлекаются из базы данных по стандартным схемам, используемым каждой за-ей независимо от вычислительного процесса на стадии подготовки исходного ма-иала для текущего этапа его реализации. Базу данных часто определяют как сово-ность взаимосвязанной информации, организованной по определенным правилам. а данных обычно определена по схеме, не зависящей от программ, которые к ней ращаются. Чтобы легче искать данные и их обрабатывать, обычно используют сис-мы управления базами данных (СУБД), которые ориентированы на поддержку вы-нения различных запросов и ведения БД. На практике шире других используются яционные СУБД, когда данные отображаются в табличной форме. В каждой СУБД ется специальный язык запросов, обеспечивающий эффективный доступ к раз-ным элементам (или их группам) в базе данных. Для баз данных реляционного ти-массовое применение получил язык запросов *SQL*.

В отличие от базы данных, в базе знаний располагаются проверенные и накоп-ные человечеством истины, факты, принципы и другие аналогичные объекты. ычно их источником являются документы, книги, статьи и т. п. В базе знаний (БЗ) полагаются в соответствии с принятой системой классификации объекты позна-, 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, т. к они позволяют получать знания, которые непосредственно не вносятся в БД, а определяются с помощью правил вывода.

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

Банк данных. На основе баз данных и знаний в различных автоматизированных системах используются банки данных. Вместе с принятой к использованию СУБД они являются его ядром. Автоматизированные системы управления, спроектированные на 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 искусственного интеллекта.

При автоматизации проектирования и принятия решений в различных областях деятельности полезно в БД иметь особый подраздел в виде архива готовых прототипов (проектов решений). Однако это связано с рядом особенностей введения этого подраздела, так как необходим экспертный отбор элементов архива, их заверение 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

анных в виде диаграмм, графиков, функций; в компьютере на языках описания дан-ых и т.д. Более высокой формой организации данных является их представление в иде баз знаний.

Знания связаны с данными, основываются на них, но представляют собой ре-ультат мыслительной деятельности человека, обобщают его опыт, полученный в хо-е практической деятельности. Знания – это выявленные закономерности предметной бласти. При обработке на ЭВМ знания также трансформируются: от существующих орм в памяти человека как результат обучения, воспитания, мышления к знаниям, омещенным на материальных носителях – учебниках, инструкциях, методических особиях, книгах и далее к знаниям, описанным на языках их представления для за-есения в компьютерные базы знаний.

Знания могут быть классифицированы на *первичные* о видимых взаимосвязях ме-ду отдельными событиями и фактами в предметной области и *глубинные* – абстрак-ии, аналогии, схемы, отображающие структуру и процессы в предметной области.

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

Исторически первыми были процедурные знания, т. е. знания, представленные в игоритмах.

Рассмотрим, например, фрагмент программы на алгоритмическом языке Пас-аль.

Pi := 3,14;
R := 20;
S := *Pi* * *R* * *R*;
WRITELN ('Площадь круга *S* =', *S*).

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

Существуют десятки способов представления декларативных знаний для раз-ичных предметных областей. Большинство из них может получаться на основе сле-ующих классов моделей: продукционных; фреймовых; семантических сетей.

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

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

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

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

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

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

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

В данном примере при рассмотрении правила 1 оказывается, что образец «наме-ение – отдых» имеется в рабочей памяти, а образец «дорога ухабистая» отсутствует, этому условная часть правила 1 считается ложной. При рассмотрении правила 2 ясняется, что его условная часть истинна. Механизм вывода выполняет заключи-льную часть этого правила, и образец «дорога ухабистая» заносится в рабочую па-ять. Правило 2 при этом выбывает из числа кандидатов на рассмотрение.

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

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

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

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

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

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

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

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

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

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

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

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

Следует обратить внимание на то, что при обратном выводе правило 3, которое не оказывает прямого влияния на достижение цели, не принималось в расчет с самого начала. Таким образом, для обратных выводов характерна тенденция исключения и

ссмотрения правил, не имеющих прямого отношения к заданной цели, что позволя- повысить эффективность вывода.

Модели знаний. Продукционная модель – это наиболее часто используемый особ представления знаний в современных экспертных системах. Основными пре- уществами продукционной модели являются наглядность, высокая модульность, гкость внесения изменений и дополнений, простота механизма логического вывода.

Следующей эффективной моделью представления знаний является фреймовая дель. Фреймовая модель использовалась первоначально в психологии и философии я описания понятия абстрактного образа. Например, слово «автомобиль» вызывает слушающих образ устройства, способного перемещаться, имеющего четыре колеса, юн для шофера и пассажиров, двигатель, руль. Приведенное описание абстрактно- образа «автомобиль» является минимальным и из него ничего нельзя убрать без тери его сущности.

Фрейм – это модель абстрактного образа, минимально возможное описание сущ- сти какого-либо объекта, явления, события, ситуации, процесса. Фрейм состоит из ени 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 ота Имя *N*-го слота: значение *N*-го слота.

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

Например, на рис. 2.5.1. фрейм «Студент» имеет ссылки на вышестоящие фрей- : «Человек» и «Млекопитающее». Поэтому на вопрос: «Может ли студент мыс- ть?» – ответ будет положительным, так как этим свойством обладает вышестоящий ейм «Человек».

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

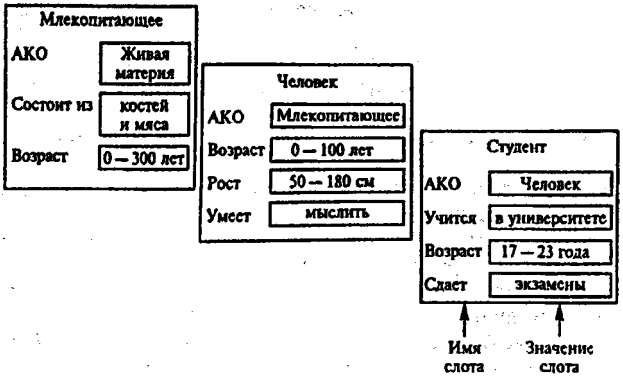


Рисунок 2.5.1 – Сеть фреймов

Основным преимуществом фреймов как способа представления знаний является глядность и гибкость в употреблении. Кроме того, фреймовая структура согласует- с современными представлениями о хранении информации в памяти человека.

В основе семантических сетей для представления знаний лежит идея о том, что бые знания можно представить в виде совокупности *понятий* (объектов) и *отно-*

шений (связей). Семантическая сеть представляет собой ориентированный граф, вершинами которого являются понятия, а дугами – отношения между ними. Сам термин «семантическая» означает смысловая.

Пример семантической сети приведен на рис. 2.5.2.

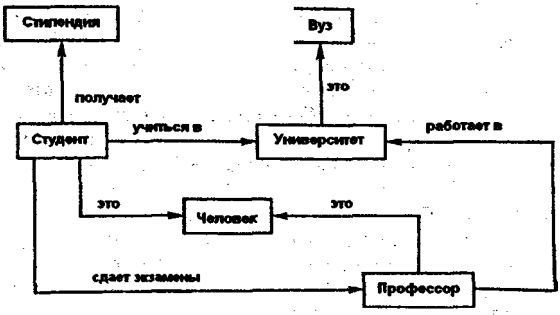


Рисунок 2.5.2 – Семантическая сеть

Основным преимуществом этой модели является наглядность представления знаний, 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.

Литература

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

Контрольные вопросы

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

Глава 3. ПРОБЛЕМЫ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

3.1. Общие подходы к созданию систем искусственного интеллекта

В течение многих столетий человек в основном занимался проблемами автоматиза-
и физического труда. Появление современной вычислительной техники, необходи-
йшего компонента автоматизации умственного труда, послужило могучим толчком к
стематическим исследованиям по автоматизации решения творческих задач.

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

В первых исследованиях в основном решался принципиальный вопрос о воз-
жности создания технических устройств, обладающих творческими способностя-
. Вскоре был получен и ряд обнадеживающих результатов прикладного плана при
пытках моделирования творческой деятельности человека по управлению различ-
ми объектами, проектированию, распознаванию образов, доказательству теорем,
же в шахматы и шашки, сочинению стихов, музыки и т. п.

Развитие робототехники в последние годы породило еще ряд сложных проблем,
занных с организацией поведения роботов. Они не возникали, пока специалисты по
усственному интеллекту имели дело с обычной ЭВМ, которая имитировала некото-
е виды интеллектуальной деятельности, но не обладала прямым контактом с внешней
дой. Появление роботов кардинально изменило ситуацию. Роботы, в отличие от
М, перемещаются в реальной физической среде, воздействуют с помощью эффекто-
на ее объекты, получают от среды с помощью рецепторов различную информацию.
ведение таких систем заинтересовало конструкторов роботов с точки зрения прием-
ости поведения подвижной системы искусственного интеллекта. В области искусст-
ного интеллекта началась работа по формальному представлению систем процедур,
спечивающих заданное нормативное поведение при заданных знаниях о внешней
де и целях функционирования системы. Это привело к развитию идей, связанных с
ождением эффективных систем представления знаний и планирования целесообраз-
деятельности. В остальных аспектах продвижение работ в области робототехники
ественно опирается на исследования в области машинного интеллекта.

Принципиальная возможность реализации различных алгоритмизируемых про-
сов на вычислительной машине была доказана еще английским математиком А.
юрингом. А. Тьюринг развивал функциональный подход к оценке деятельности ав-
атов. Такой подход нашел сторонников при решении вопросов о возможности
шинного мышления.

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

ются три комнаты. В первой комнате находится человек (экспериментатор), во
рой – машина, в третьей – человек. Экспериментатор точно не знает, в какой ком-
е находится человек, а в какой – машина. Ему предлагается задать конечную се-
задач (вопросов) и определить по ответам, в какой комнате принимаются реше-
человеком, а в какой – машиной. Естественно признать, что машина может «мыс-
ь», если экспериментатор однозначно не может указать, где находится машина.
стейшими примерами для эксперимента могут послужить решение шахматных
дов 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 функция 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 детальнее 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

айне желательно иметь эффективный алгоритм, гарантирующий получение реше-
ия, если оно существует в пределах разумных затрат времени.

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

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

При наличии такой системы подпрограмма распознавания подвергает исходные
нные последовательным испытаниям на присутствие различных признаков. После
го решается вопрос о принадлежности исследуемого объекта к одному из образов в
ответствии с весом выявленных признаков. Поэтому основной задачей многих эв-
стических программ является отыскание хорошей системы признаков, инвариант-
их относительно различных преобразований.

Обучение позволяет при решении новых задач использовать методы, оказавшие-
эффективными при решении аналогичных проблем. Такую эвристику обычно реа-
зуют на основе моделей, повышающих ее приоритет при успешном применении.
обая обучающая система должна использовать результаты прошлого опыта как ос-
ву для более общих предположений. Простейшим способом обобщения внутри
ожества признаков является построение «типичного» члена данного множества
е. усреднение). Все же возможности системы с простым повышением приоритета
раничены ее зависимостью от «учителя» (тренировочной последовательностью за-
ч). Один из путей преодоления этой трудности состоит в разработке процедуры
общения действий «учителя». Тогда при решении задач машина будет сама повы-
ть приоритет эвристик в процессе работы.

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

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

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

3.2. Попытки создания общих решателей задач

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 удается 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

апов, каждый из которых может полностью или частично выполняться вычисли-льной машиной. Если аналогичные результаты, полученные человеком на каком-то апе или на нескольких автоматизированных этапах, квалифицируются как творче-ие, то будем и машинный результат относить к разряду творческих. При невозмож-ости автоматизировать весь процесс, можно остановиться на автоматизации лишь о рутинных этапов. Однако во всех случаях машиной будет получаться либо твор-ческий результат, либо использование машины поможет человеку прийти к этому ре-ультату. На наш взгляд, в практике не существует резкой границы при классифика-ии задач на творческие и рутинные. Всегда разумно говорить о сложности той или ной конкретной задачи и необходимом уровне творчества для ее решения, т.е. счи-ть, что творчество носит многоуровневый характер. Термин «машина» становится олее расплывчатым, так как многие процессы могут выполняться программно и ап-аратно. Во многих случаях даже удобно под «машиной» понимать программу для ализации процесса. Возникает ряд новых задач, связанных с рациональным разбие-ием алгоритма на части, выполняемые человеком и машиной, а также с вопросами ффективного обмена информацией между человеком и машиной. От активного вме-ательства человека в процесс выполнения машиной программ во многом определя-ся качество результата и скорость его получения.

Поэтому в ближайшем будущем следует ориентироваться на участие человека в истемах проектирования, управления и других в качестве активного звена с доста-очно большой нагрузкой в части оценки текущей ситуации на определенном этапе ри решении задачи и принятия различных логических решений. Использование спе-иалистами на своих рабочих местах удаленных от ЭВМ терминалов или персональ-ых ЭВМ позволит использовать машину в качестве основного инструмента для вы-олнения практически любых творческих работ. Однако эффективность использова-ия этого инструмента будет определяться полнотой и качеством системы из отдель-ых модулей, ориентированных на решение частных вопросов, и наличием проблем-о-ориентированного входного языка, удобного для пользователей.

Проблема создания и использования диалоговых систем для достижения реше-ия лучших или эквивалентных по сравнению с получаемыми традиционным путем ребует отдельного рассмотрения.

3.3. Диалоговые методы решения задач и экспертные системы

Решение проблемы создания искусственного интеллекта в различных областях еятельности человека, как отмечалось выше, еще долгие годы будет идти по пути скрытия универсальных механизмов принятия решений, направленных на автомати-еское сужение области поиска результатов в соответствии с конкретной исходной туацией. Тем не менее, человеку приходится на практике решать целый ряд задач, оторые требуют использования вычислительных ресурсов. В этом случае становится огичным использование самого человека в качестве звена автоматизированной сис-емы, что порождает проблему обеспечения эффективного общения человека с ЭВМ диалоговом режиме.

Средства общения могут развиваться в следующих основных направлениях: соз-ание общего программного обеспечения для манипулирования с различными объек-ами в памяти ЭВМ на подмножествах естественного языка, опираясь на внешние стройства ЭВМ (дисплей, клавиатуру, звуковой ввод информации и т. п.) и специ-льное ПО, рассчитанное на использование средств естественного языка и графику ля выполнения некоторой части процесса решения конкретной задачи человеком.

Оба направления существенно опираются на программное и техническое обес-ечение диалога на уровне простейших операций, но построение программ в области втоматизации сложных процессов немыслимо на базе только элементарных дейст-

вий, 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.

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

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

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

Существуют неформализуемые знания, которые вообще не попадают в книги и руководства в связи с их неконкретностью, субъективностью, приблизительностью. Знания этого рода являются результатом многолетних наблюдений, опыта работы, интуиции. Они обычно представляют собой множество эмпирических и эвристических приемов и правил. Такие знания передаются из поколения в поколение в виде определенных навыков, ноу-хау, секретов ремесла. Есть также знания, которые не могут быть выражены ни в математическом виде, ни в терминах обычного человеческого языка. 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 ТЕЙРЕСИАС. Их успех у практикующих медиков в том, что они помогают врачу поставить правильный диагноз на основании рассуждений типа: «Если..., то можно предположить, что...». Программа дает врачу численную оценку вероятности каждого из заключений. За словами «можно предположить», например, стоит вероятностная оценка 0,1, а за словом «вероятно» оценка 0,8 и т.д. Программа также указывает, какие дополнительные наблюдения и тесты могут снизить неопределенность ответа.

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

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

Рассмотрим одну из возможных структур экспертной системы.

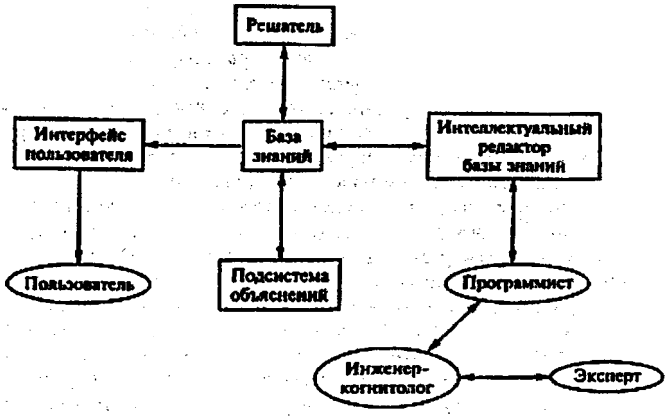


Рисунок 3.3.1 – Типичная блок-схема экспертной системы

Обобщенная блок-схема экспертной системы представлена на рис. 3.3.1. Обычно в ее состав входят следующие взаимосвязанные между собой модули:

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

– интеллектуальный редактор базы знаний – программа, представляющая инже-ру-когнитологу и программисту возможность создавать базу знаний в диалоговом жиме (Она включает в себя системы вложенных меню, шаблонов языка представ-ния знаний, подсказок (help-режим) и других сервисных средств, облегчающих ра-ту с базой знаний);

– интерфейс пользователя – комплекс программ, реализующих диалог пользова-ля с экспертной системой на стадии как ввода информации, так и получения ре-льтатов;

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

– подсистема объяснений – программа, позволяющая пользователю получать от-ты на вопросы: «Как была получена та или иная рекомендация?» и «Почему систе-приняла такое решение?». Ответ на вопрос «Как?» – это трассировка всего процес-получения решения с указанием исполняющих фрагментов базы знаний, т. е. всех ов цепи умозаключений. Ответ на вопрос «Почему?» – ссылка на умозаключение, посредственно предшествовавшее полученному решению, т. е. отход на один шаг зад.

В коллектив разработчиков экспертной системы входят как минимум четыре циалиста (или четыре группы специалистов): эксперт, инженер-когнитолог, про-ммист, пользователь. Возглавляет коллектив инженер-когнитолог – ключевая фи-ра при разработке систем, основанных на знаниях. Обычно это руководитель про-та, в задачу которого входит организация всего процесса создания экспертной сис-мы. С одной стороны, он должен быть специалистом в области искусственного ин-лекта, а с другой – разбираться в предметной области, общаться с экспертом, из-кая и формализуя его знания, передавать их программисту, кодирующему и поме-ющему их в базу знаний экспертной системы.

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

В режиме приобретения знаний происходит формирование базы знаний. В ре-ме решения задач общение с экспертной системой осуществляет конечный пользо-ель.

Обычно знания, которыми располагает эксперт, различаются степенью надежно-, важности, четкости. В этом случае они снабжаются некоторыми весовыми коэф-циентами, которые называют коэффициентами доверия. Такие знания обрабаты-ются с помощью алгоритмов нечеткой математики.

В процессе опытной эксплуатации коэффициенты доверия могут подвергаться ректировке. В этом случае говорят, что происходит обучение экспертной системы. оцесс обучения экспертной системы может производиться автоматически с помо-ю обучающего алгоритма либо путем вмешательства инженера-когнитолога, вы-няющего роль учителя.

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

Демонстрационный прототип – экспертная система, которая решает часть тре-мых задач, демонстрируя жизнеспособность метода инженерии знаний. Работает, я в базе знаний всего 50-100 правил.

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

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

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

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

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

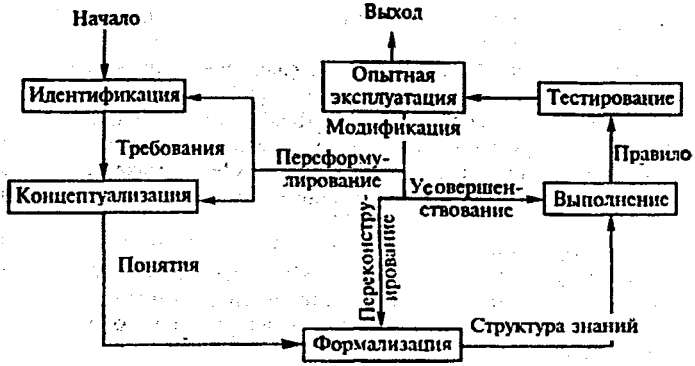


Рисунок 3.3.2 – Технология разработки экспертной системы

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

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

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

4. Выполнение. Осуществляется наполнение экспертом базы знаний. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, 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. этом случае речь идет о переформулировании понятий, используемых в системе, . перепроектировании системы заново.

3.4. Распознавание образов и обработка изображений

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

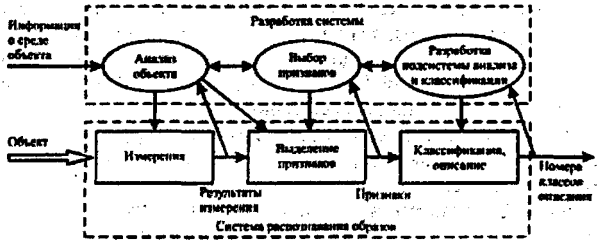


Рисунок 3.4.1 – Структура и этапы разработки системы распознавания

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

Задачи распознавания имеют следующие характерные черты.

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

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

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

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

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

Целесообразно выделить следующие типы задач распознавания.

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

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

3. Задача выбора информативного набора признаков при распознавании.

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

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

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

Основными задачами обработки изображений является обнаружение (определение границ) и локализация (определение местоположения) объектов. Основные требования к алгоритмам выделения объектов: высокая скорость при обработке больших изображений и 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 $f_1(x, y)$, ..., $f_n(x, y)$ с помощью некоторого набора или последовательности определенных операций.

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

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

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

Возможен и иной подход к обнаружению и локализации объектов на изображениях. Он заключается в проведении классификации отдельных пикселей или групп пикселей, после чего 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 $f(x, y)$; $S = \{S_1, S_2, ..., S_k\}$ – разбиение X на k непустых связных областей S_i, где $i = 1, 2, ..., k$; 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 $S = \{S_1, S_2, ..., S_k\}$, удовлетворяющее условиям:

а) $\bigcup_{i=1}^{k} S_i = X$;

б) $S_i \cap S_j = \varnothing$ для любых $i \neq j$;

в) $L_p(S_i)$ принимает истинные значения для любого i;

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

Отметим, что k областей, полученных в результате сегментации часто различных ассов, группируются в m классов, где $2 \leq m \leq k$.

Большинство известных методов сегментации и выделения контуров, можно зделить на две основные группы: областно-ориентированные, когда непосредственно строятся S_i, и гранично-ориентированные, когда определяются границы между В методах первой группы критерием однородности при задании L_p выступают обие свойства изображения: значение интенсивности (цветности) пикселя, тип текстуы, спектральные свойства изображения и т. п. Сюда относят методы порогового разления, метод разрастания (увеличения) областей, метод деления областей. В методх второй группы реализуется отслеживание контуров и при определении L_p используется характеристика изменения интенсивности – разность величин и направления адиента, т. е. два пикселя (x,y) и (x',y') подобны, если имеет место соотношение

$$L_p(S_i) = \left| \|\nabla f(x,y)\| - \|\nabla f(x',y')\| \right| \leq T;$$
$$\left| \varphi\nabla f(x,y) - \varphi\nabla f(x',y') \right| \leq A, \qquad (3.4.1)$$

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

Для вычисления градиента используются операторы локального анализа: Собе, Робертса, Кирша и т. д. В свою очередь, операторы свертки позволяют непосредвенно отобрать пиксели, которые, например, могут принадлежать прямым определенной ориентации и ширины (рис. 3.4.2).

-1	-1	-1
2	2	2
-1	-1	-1

а)

-1	2	-1
-1	2	-1
-1	2	-1

б)

-1	-1	2
-1	2	-1
2	-1	-1

в)

2	-1	-1
-1	2	-1
-1	-1	2

г)

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

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

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

Найденные контура обычно представлены в ортогональной решетке, которые зачастую аппроксимируются прямыми линиями. При этом делается попытка максимально ростить изображение контура объекта, т. е. представить его минимальным числом ямых линий. Аппроксимация базируется на использовании клеточной логики, притивов, преобразования Хафа.

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

Используются три типа моделей для представления информации о классах: де терминистская – в виде эталонов с фиксированными значениями признаков; нечетка (квази- и псевдо нечеткая, квазидетерминистская) – в виде эталонов, задаваемых точностью до неопределенных параметров или группы непрерывных; статистическая в виде статистических распределений множества параметров первичного описани (дискретных отсчетов, коэффициентов Фурье и т. п.). Выбор модели определяетс особенностями классов распознаваемых изображений, а также соображениями досто верности правильной классификации и простоты реализации. Так, например, в случа детерминистской и квазидетерминистской моделей необходимо сначала определит близость распознаваемого изображения f до эталонных $f_{i\,эт}$. Для этого формируютс некоторые функционалы – меры близости $\rho(f, f_{i\,эт})$. Наиболее часто используютс следующие меры:

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

$$\rho_i^2 = \left\{ \frac{1}{S_D} \iint\limits_{(D)} [f(x,y) - f_{i\,эт}(x,y)]^2 dxdy \right\}^{\frac{1}{2}}; \qquad (3.4.2)$$

$$\rho_i^3 = \frac{\left[\iint\limits_{(D)} f(x,y) f_{i\,эт}(x,y) dx, dy \right]^2}{\iint\limits_{(D)} f(x,y) dxdy \iint\limits_{(D)} f_{i\,эт}^2(x,y) dx, dy}.$$

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

Алгоритмы распознавания реализуют принцип похожести новых объектов с за данными (т. е. по прецедентности). Основным недостатком таких алгоритмов являет ся их эвристический характер (в основу положено то или иное дополнительное ин туитивное предположение – эвристика). При использовании таких алгоритмов нала гаются дополнительные условия: число ошибок отказов должно быть минимальным не превышать заданного числа при определенных ограничениях на распознаваемы объекты.

Каждый из используемых методов имеет свои достоинства и недостатки. Так корреляционные методы обладают относительно высокой вычислительной сложно стью, к тому же их эффективность существенно снижается при наличии различны уровней яркости и увеличении уровня шумов на анализируемом изображении. Алго ритмы, основанные на использовании геометрических признаков, менее чувствительны к вариациям яркости, однако очень чувствительны к уровню шумов. Использовани моментных инвариантов осложняется высокой чувствительностью этих признаков да же к небольшим вариациям формы выделяемых объектов относительно эталонов.

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

Построение множества информативных признаков. Под выделением призна ков подразумевают переход от исходных измерений X к новому выбору более инфор мативных в рассматриваемой задаче переменных $y = [y_1, y_2, \ldots, x_M]^T$. При это $y_i = F_i(x_1, x_2, \ldots, x_N) = F_i(\mathbf{x})$, $i = 1, \ldots, M$, где F_i – функции преобразования перемен

ых или признаковые операторы. Эти функции (операторы) могут осуществлять прото отбор части исходных измерений ($M < N$), взвешивание $y_i=\omega_i x_i$, $i = 1, ..., M$, глобальные (локальные) преобразования и т. д.

Для получения признаков могут быть использованы коэффициенты разложения о полным ортогональным системам функций. К ним следует отнести тригонометрические многочлены, полиномы Лежандра, полиномы Чебышева, коэффициенты бобщенного ряда Фурье, а также функции Хаара, Уолша и некоторые другие. Замеим, что также могут быть использованы для формирования признаков квазиортогональные системы функций, например, двумерные бинарные случайные функции. По равнению с функциями Уолша, Хаара и другими, такие функции обладают с точки рения 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 риентации, масштабу, допустимым топологическим преобразованиям и прочим атибутам. Использование инвариантных методов позволяет повысить достоверность аспознавания.

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*-ровневые фильтры, расстояния. Они обеспечивают как высокую скорость, так и невысокую сложность их вычисления на ЭВМ. Тем не менее, конструирование масок ля этого типа признаков является довольно трудной задачей.

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

Методы на основе глобальных преобразований и коэффициентов разложения в яд позволяют понизить размерность вектора признаков и в ряде случаев дают возожность получить признаки, инвариантные к некоторым глобальным деформациям зображения, таким как сдвиг или поворот. Глубокие теоретические разработки в бласти линейных операторов, одно- и двумерных преобразований, цифровой фильтрации, а также значительный прогресс в области вычислительной техники, появление оступных высокопроизводительных систем способствуют их внедрению в область аспознавания.

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

3.5. Нейронные сети как инструмент решения сложных задач

Нейроинтеллект. Интеллект характеризует способность в процессе мышления к енерированию 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 нтеллекта и пытается в той или иной мере моделировать мыслительные процессы ивых существ. В результате развития систем искусственного интеллекта возник

термин "искусственная жизнь" (Artificial Life), который отражает различные аспект поведения и взаимодействия искусственных систем. Нейронная организация и мех низмы обучения искусственных систем, процессы их самоорганизации развития р дились на стыке наук.

Интерес к нейроинтеллекту возник еще на заре развития вычислительной техн ки. В его основе лежит нейронная организация искусственных систем, которая име биологические предпосылки. Способность биологических систем к обучению, сам организации и адаптации имеет большое преимущество по сравнению с современн ми вычислительными системами. Достоинством компьютерных систем являет большая скорость распространения информации и возможность учета большого об ема знаний, накопленных человечеством в этой области. Разработка искусственны разумных систем, которые соединяют преимущества биологических существ и с временной вычислительной техники, создает потенциальные предпосылки для пер хода к качественно новому этапу эволюции в вычислительной технике.

Первые шаги в области искусственных нейронных сетей сделали в 1943 г. В. Ма Калох (W. MCculloch) и В. Питс (W. Pitts). Они показали, что при помощи пороговы нейронных элементов можно реализовать исчисление любых логических функций. 1949 г. Д. Хебб предложил правило обучения, которое стало математической осново для обучения ряда нейронных сетей. В 1957-1962 годах Ф. Розенблатт предложил исследовал модель нейронной сети, которую он назвал персептроном. Результаты и следований он обобщил в книге «Принципы нейродинамики», которая имеет больш значение для развития нейронных сетей. В 1959 г. В. Видроу (W. Widrow) и М. Хоф (M. Hoff) предложили процедуру обучения для линейного адаптивного элемента, к торый они назвали «ADALINE». Процедура обучения получила название «дель правило». В 1969 г. М. Минский (M. Minsky) и С. Пайперт (S. Papert) опубликовал монографию «Персептроны», в которой осуществили математический анализ персе трона и показали ограничения, присущие ему. Выводы их были довольно пессим стичными, и это сыграло негативную роль для дальнейшего развития исследований области нейронных сетей. Работы в области персептронных сетей были практическ приостановлены. В 70-е годы появился ряд работ в области ассоциативной памят Так, Андерсон (Anderson) предложил в 1977 г. модель линейной ассоциативной пам ти. В этом направлении продолжил исследования Т. Кохонен (T. Kohonen), которы предложил модель оптимальной линейной ассоциативной памяти. В 1976 г. С. Гросс берг (S. Grossberg) разработал теорию адаптивного резонанса, которая может бы использована для построения ассоциативной памяти.

В 80-е годы происходит значительное усиление интенсивности исследований области нейронных сетей. Д. Хопфилд (J. Hopfield) в 1982 году произвел анализ у тойчивости нейронных сетей с обратными связями и предложил использовать их д решения задач оптимизации. Тео Кохонен разработал и исследовал самоорганизу щиеся нейронные сети. Ряд авторов (Rumelhart, Hinton, Wiliams) предложил и алг ритм обратного распространения ошибки, который стал мощным средством для об чения многослойных нейронных сетей. В 1987 году под эгидой общества IEEE (Inst tute of Electrical and Electronic Enginneer's) проводится первая международная конф ренция в области нейронных сетей.

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

В настоящее время исследования в области искусственных нейронных сет ориентированы в основном на создание специализированных систем для решени конкретных задач. Разработано большое количество нейросистем, которые примен ются в различных областях: прогнозирование, управление, диагностика в медицине технике, распознавание образов и т.д. Рынок продуктов в области нейроинтеллект растет стремительным образом. Происходит постепенное накопление критическо

ассы 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. ля реализации такого рода задач можно использовать нейрокомпьютер, базовым ементом которого является нейронная сеть.

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

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

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

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

Схема нейронного элемента изображена на рис. 3.4.1 и состоит из сумматора и лока нелинейного преобразования F. Каждому i-му входу нейрона соответствует весовой коэффициент ω_i (синапс), который характеризует силу синаптической связи по налогии с биологическим нейроном.

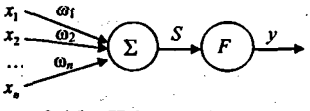


Рисунок 3.4.1 – Нейронный элемент

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

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

где $|W|$, $|X|$ – соответственно длины векторов W и X, α – угол между векторами W и X.

Длины весового и входного векторов определяются через их координаты:

$$|W| = \sqrt{\omega_1^2 + \omega_1^2 + ... + \omega_n^2},$$
$$|X| = \sqrt{x_1^2 + x_1^2 + ... + x_n^2}.$$

Так как для нейронного элемента длина весового вектора после обучения $|W| = const$, то величина взвешенной суммы определяется проекцией входного на совой 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 звешенной суммы будет зависеть только от угла между векторами X и W. Тогда при зличных входных сигналах взвешенная сумма будет изменяться по косинусоидальному закону. Максимального значения она будет достигать при коллинеарности одного и весового векторов.

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

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

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

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

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

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

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

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

Задача важна для сложных систем управления динамическими объектами.

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

Задача оптимизации – найти решение, удовлетворяющее ряду ограничений и 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.

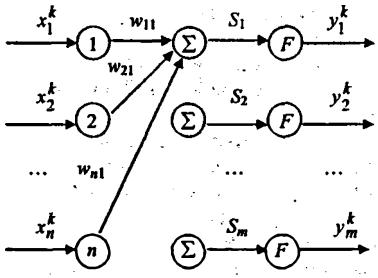


Рисунок 3.4.2 – Схема слоя НС

Для данной сети (рис. 3.4.2) каждый нейрон распределительного слоя имеет си-птические связи w_{ij}, $i = 1, \ldots, n, j = 1, \ldots, m$, со всеми нейронами обрабатывающего оя. В качестве нейронов выходного слоя используются элементы с некоторой нкцией активации F. На вход сети подаются входные образы, т. е. векторы $=(x_1^k, \ldots, x_n^k)$, $k = 1, \ldots, L$.

Выходное значение j-го нейрона сети для k-го образа определяется выражением $= F(S_j^k)$, где $S_j^k = \sum_{i=1}^{n} w_{ij} x_i^k - T_j$, $j = 1, \ldots, m, k = 1, \ldots, L$. Особенностью НС является то, о зависимость между их входами и выходами вычисляется в процессе обучения. ія обучения многослойных НС применятся алгоритм обратного распространения ибки, в основе которого лежит градиентный метод. Его недостатками для практи-ских задач являются медленная сходимость градиентного метода с постоянным ша-и обучения и сложность выбора подходящего шага для обеспечения приемлемой рости обучения. Для ускорения работы алгоритма используют эвристические ме-ды обучения на основе стандартного метода наискорейшего спуска, а также методы гимизации, например, метод сопряженных градиентов и их модификации. При м значительные усилия посвящены анализу сходимости обучения.

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

– персептронные, характеризующиеся градиентными методами обучения и нкционирующие аналогично персептрону: многослойный персептрон, рекуррент-и рециркуляционная НС (используются для классификации данных и аппрокси-ции функций);

– самоорганизующиеся НС Кохонена, характеризующиеся конкурентным мето-обучения и функционирования (используется для кластеризации данных, дискре-ации входного пространства и уменьшения размерности входного пространства);

– гибридные, отличающиеся применением двух подходов к обучению: конку-тного и градиентного: сети встречного распространения, сети с радиально базис-функцией активации (используются для кластеризации данных, интерполяции кций, распознавания образов, принятия решений);

– релаксационные, характеризующиеся правилом обучения Хебба и итеративным нципом функционирования. В таких сетях циркуляция информации происходит до пор, пока не перестанут изменяться (достигнут состояния равновесия) выходные ения НС (сети Хопфилда, Хэмминга, двунаправленная ассоциативная память).

Наиболее подходящими для распознавания являются многоуровневые НС, со-щие из каскадного соединения слоев нейронов. В основе их работы лежит прин-иерархической обработки, обеспечивающий извлечение инвариантных призна-

ков. При этом обработка обычно осуществляется в два этапа: сначала извлекаются признаки, затем достигается их инвариантность пространственным объединением. Распознаваемый образ подается на входной слой НС и далее послойно обрабатывается последующими ее слоями, при этом нейроны слоя на более глубоком уровне распознают более сложные признаки образа.

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.

Литература

[5, 7, 9, 15, 16, 17, 22, 25, 27, 28, 29, 31, 33, 34, 40]

Контрольные вопросы

1. В чем заключается особенность схемы эксперимента Тьюринга для оценки меры интеллектуальности системы?
2. В чем заключается особенность моделей ситуационного управления при описании 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. Назовите характерные черты задачи распознавания.
11. Назовите типы задач распознавания их этапы.
12. Какие преимущества даёт применение процессов обучения и самообучения НС при классификации и распознавании образов?

Глава 4. ОБЩИЕ ПОДХОДЫ К МОДЕЛИРОВАНИЮ ПРОЦЕССОВ И ОБЪЕКТОВ

4.1. Математическое и компьютерное моделирование

В последние годы резко возрос интерес к подготовке специалистов в области азработки математических и компьютерных моделей, в частности, для сферы авто-атизации проектирования, научных исследований, информационных компьютерных ехнологий, обучении и т.д. Внезапно для практики потребовалось значительное ко-ичество современных специалистов, способных разрабатывать:

– математические и алгоритмические модели научно-технических информаци-ных процессов;
– математические модели организационных и технических процессов;
– базы данных и базы знаний;
– программные системы и комплексы для различных сфер применения;
– системы с использованием сетей ЭВМ и средств связи;
– системы поддержки административного управления;
– системы трехмерного моделирования;
– системы с использованием средств искусственного интеллекта в принятии ре-ений и др.

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

Однако по компьютерной модели проще и удобнее проводить вычислительные сперименты, часть из которых в натуре иногда слишком дорога или неосуществима обще (угроза взрыва, потеря объекта и т. п.). На модели легче проверить реакцию учаемой системы на изменение ее параметров.

К основным этапам компьютерного моделирования относят:

– постановка задачи (иногда более половины успеха дела);
– определение границ объекта моделирования;
– определение процессов взаимодействия между частями модели;
– формализация элементов модели;
– разработка алгоритма воспроизведения процесса на модели;
– написание программы (подбор подходящей готовой системы и ее адаптация;
– проведение экспериментов и фиксация их результатов;
– анализ и интерпретация итогов экспериментов.

Иногда полезно различать аналитическое и имитационное моделирование. Под алитическим моделированием понимают разработку модели реального объекта пу-м описания его средствами математики (на основе подбора различных функций и параметров, связей между ними и т. д.). Имитационное моделирование обычно одится к построению значительного количества функций для описания отдельных ементов процесса. Его результатом является многократное воспроизведение взаи-одействий между элементами модели с последующим получением результата на осве усредненных показателей изучаемого явления в целом.

При создании моделей на практике необходимо соблюдать ряд наиболее важных инципов:

– гарантия осуществимости реализации модели с позицией достаточности ин-рмации для ее функционирования;

– возможность достижения поставленной цели за заданное время;

– допустимость модификации модели по мере уточнения информации и результатов;

– использование максимально возможного набора стандартных средств, функций, программ

Математические модели всего объекта и его частей выражают некоторые его существенные черты на языке уравнений и формул. Современные компьютеры при математическом моделировании используются не только для численных расчетов, но и для построения графиков, воспроизведения звуков и аналитических расчётов. Например, Mathcad и другие аналогичные системы являются хорошими готовыми средствами для таких целей. Эти новые возможности открывают пути к более широкому пониманию и использованию современных компьютерных технологий. В частности, использование графики и речи позволяет воспроизводить более сложные процессы. Например, имитацию работы технических устройств с участием человека, генерацию моделей на основе интеллектуальных банков данных.

Наиболее широкое продвижение математического моделирования нашло себя в системах автоматизированного проектирования (САПР) в области радиоэлектроники, так как такого рода системы в значительной мере влияют на технический прогресс в развитии вычислительной техники. Для этих целей было сделано большое количество разработок в создании математического обеспечения процессов проектирования в микроэлектронике, языков проектирования схем, специальных баз данных с типовыми решениями в области схемотехники, СУБД и др. Здесь впервые стало уделяться внимание комплексному решению задачи проектирования и производства радиоэлектронных схем с целью создания единой системы от проектирования микроэлектронных схем до их выпуска на оборудовании с числовым программным управлением, что в современных условиях позволяет в течение нескольких месяцев переходить к модернизации производства выпускаемых радиоэлектронных изделий (например, смартфонов) и резко сократить сроки производства новинок. Создаются системы проектирования креативного типа, когда они допускают изменение хода процесса проектирования за счет использования опыта и интеллекта инженеров. Многое в этом направлении сделано в решении задачи размещения компонентов и трассировки печатных плат и создании монолитных кристаллов для чипов.

Эти работы послужили толчком к развитию компьютерной графики, развитию нейросетевых методов решения задач и созданию специальных типов компьютеров (нано-компьютеров, биокомпьютеров, нейрокомпьютеров и др.), а также методов и программных средств 3*D* моделирования. Компьютерная модель является программной реализацией математической модели (абстрактной знаковой модели), которая частично отражает исходный объект моделирования, исходя из целей его исследования, т.е. математическая модель ограничена отражением только необходимых свойств объекта, которые влияют достаточно адекватно на исследуемый процесс. В этом смысле знаковые модели могут быть довольно схематичными (план цеха в виде чертежа для решения задачи размещения оборудования и т. п.), что упрощает моделирование задаваемого процесса. Процесс исследования на модели завершается получением параметров соответствующих заданному критерию с необходимой точностью. Его обеспечение предполагает выполнение некоторых стандартных шагов:

– выделение существенных свойств для объекта моделирования с позиций предполагаемых для решения задач и построение математической модели;

– построение и отладка компьютерной модели;

– оценка соответствия модели для решения класса задач;

– адаптация модели на предмет обеспечения полноты класса решаемых задач.

4.2. Модели оптимизации многомерных функций и принятие решений

Принятие решений в научных, технических, экономических и других областях ятельности часто сводится к поиску оптимальных (рациональных) решений с позиий лица, принимающего решения (ЛПР). Во многих случаях при наличии программых продуктов поддержки принятия решений ЛПР мотивируют правильность своего ешения, ссылаясь на это обстоятельство. Поэтому особо подчеркнём, что ЭВМ лишь нструмент для принятия решения, за которое несёт ответственность пользователь. уществует много методов поиска оптимальных значений функций методами клас ической высшей математики для непрерывных и дискретных значений их перемен ых. Поэтому выбор конкретной системы оптимизации оставим за пользователем. ы остановимся лишь на некоторых приёмах преодоления трудностей в решении ногомерных дискретных и непрерывных задач, когда функция вычислима.

Решение дискретных сложных задач адаптивными приближенными методами пирается на знания из теории оптимизации функций одной переменной. К сложным искретным задачам обычно относят те, которые в худшем случае точно решаются ри полном переборе всех вариантов решений. Их часто называют *NP*-полными. Та ие задачи пытаются решать различными методами, включая использование также и ейросетевых моделей.

При принятии решений часто цель управления выражается в виде функции мно их переменных (дискретных и/или непрерывных), пределы изменения которых из естны, что позволяет строить стратегию выбора рационального или оптимального езультата. Основная трудность в решении этих задач состоит в выборе приемлемого абора переменных $(x_1, x_2, \ldots, x_n)$ с позиций практики. Универсальный характер носит етод покоординатного спуска. Его применение можно свести к последовательному оиску экстремумов задач от одной переменной x_i, когда все оставшиеся переменные иксируются на время поиска. Полученное значение x_i делается фиксированным, а $_{+1}$ становится следующей переменной и т. д. Остановка наступает, когда за весь цикл иксируется достигнутый максимум (минимум). Можно делать несколько циклов одряд в зависимости от ограничения по времени. Критерий остановки выбирается ПР. Желательно также указывать шаг изменения свой для каждой переменной в теущем цикле, корректировать области их изменения и точность решения, т.е. в этом лучае окажется полезным диалоговый метод управления по ходу решения проблемы.

По близкой по характеру схеме решается задача градиентным методом оптими ации для непрерывных дифференцируемых функций от многих переменных. Её так е желательно решать в диалоговом режиме. Общность подходов наблюдается в вы оре с учётом ограничений исходной точки начала итерационного процесса (интуи ивно, случайным механизмом, по некоторому закону и т. п.). Аналогично решается и адача выбора длины шага на каждой итерации для рассматриваемой переменной x_i. писание таких механизмов и их реализация приведены в лабораторной работе № 7.

Остановимся более подробно на реализации случайных механизмов и их роли в ешении сложных задач. Когда о поведении функции недостаточно информации, ожно с равной вероятностью выбирать любое значение для назначения координаты ачальной точки дискретной переменной или выбирать её из заданного отрезка для епрерывной переменной. Такой приём позволяет в какой-то мере при старте из не кольких попыток повысить шансы избежать попадания в локальный экстремум. Ве оятностные приёмы могут быть и более сложными.

Вероятностное моделирование на ЭВМ обычно опирается на использование ге ератора случайных чисел равномерно распределенных в диапазоне (0, 1) или (0, a), оторый содержится в большинстве вычислительных систем в виде стандартной про едуры. На его основе обычно строятся случайные процессы с использованием стан-

дартных процедур для воспроизведения элементов моделей в виде конкретных случайных законов (экспоненциальный, нормальный и др.).

Второй путь использования генератора случайных чисел состоит в выборе случайных условий или путей продолжения процессов, когда требуется исключить влияние исследователя на выбор текущего продолжения процесса, например, стартовых точек для различных итерационных процессов вычислений для многомерных функций и т.п.

4.3. Методы защиты информации и электронная подпись документов

Защита информации и создание безбумажных систем документирования стали тесно переплетаться на основе общих средств используемого математического аппарата.

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

Успехи в защите информации достигаются там, где используется организованная система, опирающаяся на широкий комплекс с взаимодействием различных средств: привычных всем замков и сейфов, охрану объекта людьми, хитроумные системы на основе лазерной техники и различных камер наблюдения и, наконец, программные методы. Во многих случаях программные методы являются основными, так как с их помощью решается главная задача – защита от несанкционированного доступа даже украденной (скопированной) информации, а также авторизация (подпись) документов, существующих только на машинных носителях, и дистанционное управление банковским счетом.

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

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, наименова-

иями типов вооружений и их количества и т.п.). Практические трудности здесь стали ключаться в том, что надо не только затруднить раскрытие истинного сообщения, и само шифрование сделать достаточно легким.

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

Постепенно созрели идеи и о необходимости использования государственных андартов шифрования, чтобы уверенно и безопасно пользоваться программами и алритмами, прошедшими экспертизу в специальных государственных органах. В мире рвым таким стандартом стал DES (Data Encryption Standart, США). Он опирался на крытый для всех шифроалгоритм, основанный на реализации принципов рассеивая и перемешивания. В нем открытый текст, криптограмма и ключ представлялись в де двоичных последовательностей длиной соответственно 64, 64 и 56 битов. Спустя лее 30 лет после его опубликования, с помощью специального компьютера «Больй 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 (Adnced Encryption Standart – улучшенный стандарт шифрования), в котором для пере-ра всевозможных ключей (2^{256}) потребуется столько операций, которые практически льзя осуществить даже на специальном комплексе из многих суперкомпьютеров за иемлемое время.

Аналогичного типа стандарты стали использоваться в качестве ГОСТов и в дру-х странах (Россия и др.). Они относятся к классу систем с симметричным алгорит-м шифрования, т. е. один и тот же ключ используется для шифрования и дешифрования текста. В этом случае возникла другая сложность: как передать адресату ключ открытым каналам связи без риска его перехвата. Поэтому дальнейшее усовер-енствование криптосистем пошло в направлении создания двухключевых систем: первый ключ публикуется для применения всеми пользователями при расшифро-нии данных, а для их шифрования имеется секретный второй ключ, который нельзя олучить из первого, что сделало сам зашифрованный текст равнозначным подписан-му. Суть математических преобразований для этих целей опирается на теорию ункций с «лазейкой», когда $y = f(x)$ легко вычисляется, а обратная функция $x = f(y)$ *вычислимая без знания дополнительной информации (лазейки).* Простейший при-р такого типа представляет вычисление целочисленной функции $y = x \bmod n$, когда результатом является остаток от деления целого числа на целочисленный модуль n. апример, $y = 12 \bmod 7 = 5$, но обратная операция отыскания x как функции от остатка не выполняется однозначно (один и тот же остаток при делении на 7 могут дать 5, , 19,...) Неизвестное x можно вычислить лишь при знании «лазейки», например, мера числа в ряду 5, 12, 19, ...

Покажем реализацию одноключевой системы такого типа с открытой передачей ючей на основе функций с лазейкой. Для открытого распространения ключей Диф- и Хелман предложили использовать функцию $f(x) = a^x \pmod p$, где p – очень льшое простое число, x – целое от 1 до $(p - 1)$ число; a – целое число), степени ко-рого в некотором порядке равняются 1, 2, 3,... $(p - 1)$ по модулю p. Предполагает-, что всем пользователям закрытой системы известны a и p (сообщает администра-р системы при регистрации). Пользователь i случайным образом выбирает целое сло x_i $(1 < x_i < (p - 1))$ и держит его в секрете. Потом он вычисляет $y_i = a^{x_i} \pmod p$ и мещает его в открытый для пользователей сети справочник. При желании устано-ть секретную связь с другим пользователем сети j, он берет его число y_j и с помо-ью секретного ключа x_i вычисляет число $Z_{ij} = (y_i)^{x_i} \pmod p$. Аналогично j вычисляет (оно равно Z_{ij}), и далее это число они могут использовать как ключ для обмена за-ифрованными 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), $y_j = 3^5 \pmod 7 = 5$;

$Z_{ij} = 5^3 \pmod 7 = 125 \pmod 7 = 6$; $Z_{ij} = 6^5 \pmod 7 = 7776 \pmod 7 = 6$.

Предположим, цифра 6 будет означать страницу в каком-то справочнике или книге, где содержатся функции для шифрования текстов. (Администратор может выдавать им такой справочник и время от времени его менять; важно, что их ключ (6) остается секретным и для администратора сети, так как «6»не передается по открытому каналу).

Решение задач, связанных с признанием электронной подписи, идентификации и аутентификации, тесно связаны с системой шифрования RSA, ХЭШ-функциями и исследованиями по развитию такого рода механизмов.

Алгоритм RSA – компонента ряда криптосистем для цифровой подписи. Алгоритм RSA обеспечивает высокую степень защиты, поскольку в нем в качестве модуля используется произведение двух больших простых целых чисел. В такой системе каждый пользователь имеет свой ключ шифрования и дешифрования. Ключи дешифрования известны всем пользователям системы, а шифрующий ключ держит в секрете его владелец. Криптографические системы типа RSA подходят для реализации цифровой подписи, применяемой в системах электронных платежей и при передаче сообщений с помощью устройств телесвязи.

К недостаткам системы RSA и аналогичных ей относят ее существенно более 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. Определить другое число E такое, что $(ED) \bmod L = 1$;
5. Как только все числа найдены, мы имеем: секретный ключ – E; открытый ключ – пара чисел D и N.

Тогда при шифровании сообщения его разбивают на блоки M. В результате шифрования для каждого блока M получим число

$$C = (M^E) \bmod N.$$

При дешифрации получаем:

$$M^* = (C^D) \bmod N.$$

Рассмотрим это на примере алфавита из букв {Л,О,Я}={1,2,3} для передачи текста «ОЛЯ» (или 2,1,3). Цифровые обозначения букв или блоков обязательны, так как метод основывается на обработке натуральных чисел.

1. Выберем $P = 3$ и $Q = 11$.
2. Найдем $N = PQ$, $N = 33$; $L = (P - 1)(Q - 1)$; $L = 20$.
3. Выберем D взаимно простое с L: $D = 3$.
4. Выберем E такое, что $(ED) \bmod L = 1$, например, $E = 7$: действительно $(7 \times 3) \bmod 20 = 1$.

5. Тогда открытый ключ $\left.\begin{matrix} D=3 \\ N=33 \end{matrix}\right\}$, секретный $E=7$.

Производим шифрацию:

$$C3=(3^7) \mod 33=9,$$

$$C1=(M1^B) \mod N: \quad C2=(1^7) \mod 33=1,$$

$$C1=(2^7) \mod 33=29.$$

Зашифрованный текст получается (29, 1, 9).

Расшифровка:

$M1^* = (C1^D) \mod N$: $(29^3) \mod 33 = 2$,

$M2^* = (C2^D) \mod N$: $(1^3) \mod 33 = 1$,

$M3^* = (C3^D) \mod N$: $(9^3) \mod 33 = 3$.

В результате мы получили исходный текст: 2, 1, 3.

Остается только добавить, что для получения достаточно стойкой шифровки необходимо брать очень большие простые числа.

Выполнение соотношения $(ED) \mod L = 1$ позволяет использовать этот факт для проверки подлинности подписи без знания секретного ключа E с помощью аппарата хэш-функций.

В практической работе необходимо идентифицировать автора электронного документа и предприятие не по особенностям подписи и печати (например, по образцам подписей и печатей в банковской карточке клиента), а по наличию у него электронного ключа для подписывания документов. В этом случае конкретное число-подпись под данным документом в фиксированное время, может сделать только законный обладатель ключа (E).

Процедура электронной подписи включает в себя два этапа: первый – подписывание (вычисление параметров подписи, зависящих от текста конкретного документа, один из которых (E) хранится в секрете); второй – проверка получателем с помощью несекретных параметров (D, N) подлинности сообщения (подписи)

Сообщение шифруется по алгоритму RSA, где E подбирается и известно только правителю, а D, N знает и получатель. Получатель должен иметь возможность с помощью открытого ключа проверить подлинность сообщения. Для этой цели в сообщение добавляется еще одно число, которое является результатом вычисления хэш-функции $h(T)$, зависящей от текста T.

К хэш-функции предъявляется ряд требований:

– невозможность (или возможность за очень длительное время) найти по значению $h(T)$ само T (т. е. требуется построить практически необратимую функцию);

– для заданного T нельзя найти такое T', чтобы $h(T) = h(T')$;

– вообще нельзя найти пару различных слов T и T' такую, что $h(T) = h(T')$;

– сообщение T (например, текст договора, платежного поручения и т. п.) по заданной функции сжимается в целое число $m = h(T)$, причем $1 < h(T) < N$.

Число m позволяет с помощью открытого ключа констатировать подлинность документа.

С этой целью автор документа с помощью своего секретного ключа E получает второй параметр подписи $S = (m^E) \mod N$. 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 символьного объема зашифрованного сообщения (T_1) и «лазейки» для вычисления $h(T)$, $h(T_1)$.

Проверка параметра S производится путем идентификации условия: $(S^D) = m \mod N$.

Математически доказано, что результат проверки m и s будет положительным в том случае, когда в их формировании использовался секретный ключ E, соответст-

вующий открытому ключу D. Вероятность расшифровки секретного ключа E по открытым параметрам S, m, D и N считается ничтожно малой из-за затрат времени на решение задачи взлома системы по сравнению со временем полезного действия сообщения.

Покажем в упрощенном варианте проверку подписи сообщения T = (ОЛЯ), с дополнением его параметрами m и S. В качестве функции хеширования $h(T)$ возьмем произведение сумм из двух элементов для каждого шифруемого знака: его кода по ходу текста с простым числом (2,3,5,7,...) по порядку следования. В нашем случае их позиции О = 1, Л = 2, Я = 3, а коды Л = 1, О = 2, Я = 3, т. е. П = (2 + 2)(1 + 3)(3 + 5) = 128, чтобы получить m вычисляем его так: m = 128 mod 33 = 29. Можно убедиться, что эта функция удовлетворяет требованиям к ней по крайней мере для любого сообщения из трех разных букв. Это обнаруживается при вычислении всех возможных 6 разнобуквенных сообщений T: (ОЛЯ) –29, (ОЯЛ) – 12, (ЛОЯ) – 21, ЛЯО (27), ЯОЛ (18), ЯЛО (8), т. е. нет равных $h(T)$.

Следует заметить, что с ростом длины сообщения может оказаться, что такая функция не удовлетворяет требованию, когда два сообщения разной длины имеют одинаковое значение, т. е. $h(T) = h(T')$. Чтобы избежать такого явления можно разбивать сообщения на блоки заранее ограниченной длины. Поэтому выбор хорошей функции $h(T)$ является очень трудной задачей и ее решением занимаются специалисты, которые разрабатывают стандарты шифрования. В описанном нами примере ограничимся лишь демонстрацией процедуры признания подписи.

Текст T = ЛОЯ, который получает партнер позволяет проверить подлинность подписи по параметрам m, S и D = 3 (известно как открытая часть ключа), s приходит с текстом отправителя ($S = m^E \bmod N = 21^7 \bmod 33 = 21$; $m = 21 < 33$). 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 П34, 101.25-2008 для электронной подписи, в котором алгоритм RSA один из трех рекомендуемых для применения. Предприятия могут пользоваться различными системами шифрования, но в этом случае стороны обязаны подписывать договор об использовании избранного алгоритма подписи для своей корпоративной сети. Например, белорусские банки имеют такого типа договоры

4.4. Защита от несанкционированного доступа

Защита от несанкционированного доступа к информации, циркулирующей или хранящейся в различных часто удаленных друг от друга узлах сети, является одной из самых сложных. Это связано с тем, что необходимо решать не только задачу о допуске удаленного пользователя к разрешенным ему объектам для доступа в рамках его полномочий, но и определении личности пользователя, чтобы вместо него не работал за терминалом сети или самовольно подключенным терминалом злоумышленник. Для решения такого рода задач созданы специальные инструменты – сетевые протоколы.

Их составной частью являются элементы для идентификации и аутентификации пользователей. Идентификация пользователя, как правило, сводится к определению его полномочий на доступ к данным и устройствам по фамилии и паролю. Аналогичная задача может решаться и по отношению к удаленному терминалу на предмет его законного подключения к сети.

Аутентификация является более сильной формой опознания личности при совершении ответственных операций, так как в современных системах требуется предъ-

лять физические характеристики пользователя (палец, ладонь, произношение некоторых из заданных слов и т.п.)

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 оимости систем защиты информации. На практике используется критерий, опирающийся на оценку стоимости «взлома» системы. Если «взлом» системы обходится роже 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 ифрования, но и методы стеганографии, задачи которых – скрыть сам факт переылки секретного сообщения. Для этой цели используются два типа файлов: само сообщение (первый закодированный файл) и контейнер (второй файл), задача которого рыть содержимое основного файла, которое потом с помощью специального ключа влекается из вспомогательного файла, выглядящего как безобидное сообщение (наимер, открытка с поздравлением к 8 Марта и т.п.).

4.5. Безбумажные системы ведения документации

Особую роль в создании таких систем играют стандарты различных уровней редприятия, отрасли, страны, международные). С их помощью достигается качество нкционирования систем и их совместимость с другими системами.

Система стандартов менеджмента качества разработана Техническим комитем Международной Организации по Стандартизации (ISO, International Organization r Standardization).

Стандарты серии ISO 9000, принятые более чем 90 странами мира в качестве наональных, применимы к любым предприятиям, независимо от их численности, ъема выпуска и сферы деятельности.

При создании системы безбумажного документооборота на предприятии важную ль играют программные средства формирования электронной подписи и шифрования документов. Особенность этих средств по реализации электронной подписи соит в том, что они должны быть сертифицированы на государственном уровне, так к только в этом случае электронный документ признаётся судом в качестве эквивата бумажному. Организация системы электронной почты между всеми структурными подразделениями и доступ к различным базам данных фирмы может также базироваться на электронной подписи, системе шифрования и стандартных средствах ступа, аналогичных имеющимся для пользователей Интернета.

Комплексность характера защиты документооборота достигается одновременм применением физических средств (замки, пропуска, сигнализация, видеокамеи), аппаратно-программных средств (борьба с перехватом сообщений, шифрование, наружение «жучков» и т. п.), организационные средства (инструкции, приказы и .), наиболее специфическим средством защиты является шифрование, позволяюее сохранить втайне от злоумышленника даже украденную информацию.

Юридической основой для ведения «безбумажного» документооборота являются коны РБ «Об информации» и «Об электронном документе». Законом устанавливася обязательная структура электронного документа (ЭД), он имеет силу в юридичеом плане аналогичную обычным заверенным документам (например, в судах).

Общая часть ЭД должна иметь структуру и реквизиты, установленные законами стандартами для обычных документов (дата создания, утверждение, наименование, едения от его создателях), а также сведения о защите ЭД и средствах выполнения ектронной цифровой подписи (ЭЦП), сведения о технических и программных средвах, необходимых для воспроизведения ЭД и его составе. ЭЦП должна выполнять

две функции, что отличает её от обычной подписи. Во-первых, она должна удостове рять информацию общей (содержательной части) ЭД и выполнять роль печати и под писи. Во-вторых, ЭЦП предназначена для подтверждения подлинности и целостност ЭД, что необычно для обычного бумажного делопроизводства. Эта функция пресека ет возможность изменить ЭД, не оставляя никаких следов вмешательства. Это свой ство обеспечивает безупречность выполнения дистанционных платежей в электрон ной форме.

Проблемой в развитии новых подходов к управлению предприятиями является оценка затрат на реализацию информационных технологий, так как трудно оценивать затраты на платные информационные услуги в различных сетях ЭВМ.

Из зарубежных баз данных можно получать достаточно объективные сведения из сайтов предприятий, описаний патентов, новинках науки, биржевых операциях электронных изданиях в виде книг, рекламе и т.п.

В собственных базах данных имеется достаточно много полезной информации по различным аспектам деятельности предприятия за ряд лет. Сопоставив информа цию из этих баз данных, можно более уверенно строить разумные стратегии поведе ния в бизнесе.

Литература

[2, 3, 4, 5, 8, 9, 13, 14, 15, 19, 24, 25, 27, 34, 35, 36, 37, 40] .

Контрольные вопросы

1. Назовите основные этапы при реализации компьютерного моделирования
2. В чём различие между аналитическим и имитационным моделированием?
3. От чего зависит характер основных свойств, отражаемых моделью?
4. Назовите общие подходы в оптимизации многомерных целевых функций.
5. В чём выражается общность подходов при оптимизации дискретных и непре рывных многомерных функций?
6. Зачем используются генераторы случайных чисел в процессах оптимизации?
7. В чём выражается общность защиты и подписи документов?
8. Зачем вводится государственная сертификация систем защиты информации?
9. В чём различие между одноключевыми и многоключевыми системами шиф рования?
10. Назовите свойства функций с «лазейкой».
11. Как делается проверка подлинности ЭЦП?
12. Назовите функции механизма стеганографии.
13. Перечислите обязательные элементы в структуре электронного документа
14. Сравните процедуры идентификации и аутентификации по их функциям.

Глава 5. АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

5.1. Основные понятия автоматизированного проектирования

Системный подход к проектированию. Ручная разработка ВС обычно порожала следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

Перечисленные факторы способствовали появлению программно-технологичесих средств специального класса – CAD и CASE-средств, которые частично устрани- и названные выше проблемы. Термин CASE (Computer Aided Software Engineering) спользуется в настоящее время в весьма широком смысле. Первоначальное значение ермина CASE, ограниченное вопросами автоматизации разработки только лишь ПО, настоящее время приобрело новый смысл, охватывающий процесс разработки ложных ВС в целом. Теперь под термином CASE-средства понимаются программые средства, поддерживающие процессы создания и сопровождения ВС, включая нализ и формулировку требований, проектирование прикладного ПО (приложений) баз данных, генерацию кода, тестирование, документирование, обеспечение качеста, конфигурационное управление и управление проектом, а также другие процессы. ASE-средства вместе с системным ПО и техническими средствами образуют полую среду разработки ВС.

Проектирование, при котором все проектные решения или их часть получают утем взаимодействия человека и ЭВМ, называют автоматизированным, в отличие от учного (без использования ЭВМ) или автоматического (без участия человека). Сиема, реализующая автоматизированное проектирование, называется САПР (CAD ystem –Computer Aided Design System). В основном используется автоматическое роектирование, а автоматическое проектирование возможно лишь в отдельных слуаях для сравнительно несложных объектов.

Проектирование сложных объектов основано на применении принципа системого подхода, суть которого заключается в рассмотрении частей явления или сложой системы с учетом их взаимодействия. Системный подход включает в себя выявение структуры системы, типизацию связей, определение атрибутов, анализа влияия внешней среды. Интерпретация и конкретизация системного подхода имеют мето в ряде известных подходов с другими названиями. Таковы структурный, блочноерархический, объектно-ориентированный подходы.

При *структурном подходе*, как разновидности системного, требуется синтезировать варианты системы из компонентов (блоков) и оценивать варианты при их часичном переборе с предварительным прогнозированием характеристик компонентов.

Блочно-иерархический подход к проектированию использует идеи композиции ложных описаний объектов и соответственно средств их создания на иерархические ровни и аспекты, вводит понятие стиля проектирования (восходящее и нисходящее), станавливает связь между параметрами соседних иерархических уровней.

Ряд важных структурных принципов, используемых при разработке информацинных систем и прежде всего их ПО, выражен в *объектно-ориентированном* подходе к роектированию. Такой подход имеет следующие преимущества в решении проблем правления сложностью и интеграции ПО: вносит в модели приложений большую труктурную определенность, распределяя представленные в приложении данные и роцедуры между классами объектов; сокращает объем спецификаций, благодаря введению в описания иерархии объектов и отношений наследования между свойствами бъектов разных уровней иерархии; уменьшает вероятность искажения данных вслед-

ствие ошибочных действий за счет ограничения доступа к определенным категориям данных в объектах. Описание в каждом классе объектов допустимых обращений к ним и принятых форматов сообщений облегчает согласование и интеграцию ПО.

Наиболее распространенной методологией проектирования ПО на основе структурного подхода является INFRUN, разработанная фирмой SilverRun. Жизненный цикл ПО декомпозируется на стадии, которые связываются с результатами выполнения основных процессов. Технологии поддерживаются в различных CASE-средствах. Тем не менее, структурный подход не позволяет выделить абстракции и обеспечить ограничение доступа к данным; он также не предоставляет достаточных средств для организации параллелизма и он, как правило, неэффективен в объектных и объектно-ориентированных языках программирования.

Метод потоков данных предполагает рассмотрение программной системы в качестве преобразователя входных потоков в выходные. Метод потоков данных, как и структурный метод, с успехом применялся при решении ряда сложных задач, в частности, в системах информационного обеспечения, где существуют прямые связи между входными и выходными потоками системы, и где не требуется уделять особого внимания быстродействию.

В основе объектно-ориентированного подхода лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причем классы образуют иерархию. Этот 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.

Для всех подходов к проектированию сложных систем характерны следующие особенности.

1. *Структуризация* процесса проектирования, выражаемая декомпозицией проектных задач и документации, выделением стадий, этапов, проектных процедур. Эта структуризация является сущностью блочно-иерархического подхода к проектированию.

2. *Итерационный характер проектирования.*

3. *Типизация и унификация* проектных решений и средств проектирования.

Таким образом, при системном подходе основными являются:

– построение иерархической структуры систем, организация их проектирования;

– анализ и моделирование систем;

– синтез и оптимизация систем.

Моделирование имеет две четко различимые задачи: создание моделей сложных систем (modelling) и анализ свойств систем на основе исследования их моделей (simulation).

Синтез также подразделяют на две задачи: синтез структуры проектируемых систем (*структурный синтез*) и выбор численных значений параметров элементов систем (*параметрический синтез*). Эти задачи относятся к области принятия решений.

Моделирование и оптимизацию желательно выполнять с учетом статистической природы систем. Детерминированность – лишь частный случай. При проектировании характерны нехватка достоверных данных, неопределенность условий принятия решений. Учет статистического характера данных при моделировании в значительной мере основан на методе статистических испытаний (методе Монте-Карло), а принятие решений – на использовании нечетких множеств, экспертных систем, эволюционных вычислений.

Иерархическая структура системы. При блочно-иерархическом проектировании представление о проектируемой системе расчленяют на *иерархические уровни*. По

ти, используется декомпозиционный подход, который основан на разбиении слож-й задачи большой размерности на последовательно и (или) параллельно решаемые дачи меньшей размерности, что сокращает требования к используемым вычисли-льным и временным ресурсам. На верхнем уровне используют наименее детализи-ванное представление, отражающее только самые общие черты и особенности про-тируемой системы. На следующих уровнях степень подробности описания возрас-ет, при этом рассматривают уже отдельные блоки системы, но с учетом воздейст-й на каждый из них его соседей.

Можно говорить не только об иерархических уровнях спецификаций, но и об *ие-рархических уровнях проектирования*, понимая под каждым из них совокупность спе-фикаций некоторого иерархического уровня совместно с постановками задач, ме-дами получения описаний и решения возникающих проектных задач. Для боль-инства приложений характерно следующее наиболее крупное выделение уровней:

– *системный* уровень, на котором решают наиболее общие задачи проектирова-я ВС и процессов; результаты проектирования представляют в виде структурных ем, схем размещения оборудования, диаграмм потоков данных и т. п.;

– *макроуровень*, на котором проектируют отдельные устройства, узлы машин и риборов; результаты представляют в виде функциональных, принципиальных и ки-матических схем, сборочных чертежей и т. п.;

– *микроуровень*, на котором проектируются отдельные детали и элементы машин приборов.

В каждом приложении число выделяемых уровней и их наименования могут ть различными. В радиоэлектронике микроуровень часто называют компонентным, акроуровень – схемотехническим, между ними вводят уровень, называемый функ-ионально-логическим. Системный уровень подразделяют на уровни проектирования С и вычислительных сетей.

В зависимости от последовательности решения задач иерархических уровней зличают нисходящее («сверху-вниз», от верхних уровней к нижним), восходящее снизу-вверх», от нижних уровней к верхним) и смешанное проектирование (имеют-элементы как восходящего, так и нисходящего стиля проектирования). В большин-ве случаев для сложных систем предпочитают нисходящее проектирование. Отме-м, однако, что при наличии заранее спроектированных блоков (устройств) можно ворить о смешанном проектировании.

Неопределенность и нечеткость исходных данных при восходящем проектиро-нии (так как еще не спроектированы компоненты) или исходных требований при сходящем проектировании (поскольку техническое задание имеется на всю систе-у, а не на ее части) обуславливает необходимость прогнозирования недостающих нных с последующим их уточнением, т. е. последовательного приближения к окон-тельному решению (*итерационность* проектирования).

Наряду с декомпозицией описаний на иерархические уровни применяют разде-ение представлений о проектируемых объектах на аспекты.

Аспект описания (страта) – описание системы или ее части с некоторой оговo-енной точки зрения, определяемой функциональными, физическими или иного типа тношениями между свойствами и элементами.

Различают аспекты функциональный, информационный, структурный и поведен-еский (процессный). *Функциональное* описание относят к функциям системы и чаще его представляют его функциональными схемами. Информационное описание вклю-ает в себя основные понятия предметной области (сущности), словесное пояснение ли числовые значения характеристик (атрибутов) используемых объектов, а также писание связей между этими понятиями и характеристиками. Информационные моде-можно представлять графически (графы, диаграммы «сущность-отношение»), в виде блиц или списков. *Структурное* описание относится к морфологии системы, харак-

теризует процессы функционирования (алгоритмы) системы и (или) технологические процессы создания системы. Иногда аспекты описаний связывают с подсистемами, функционирование которых основано на различных физических процессах.

Отметим, что в общем случае выделение страт может быть неоднозначным. Так, помимо указанного подхода, очевидна целесообразность выделения таких аспектов, как *функциональное* (разработка принципов действия, структурных, функциональных, принципиальных схем), *конструкторское* (определение форм и пространственного расположения компонентов изделия), *алгоритмическое* (разработка алгоритмов и ПО) и *технологическое* (разработка технологических процессов) проектирование систем. Примерами страт в случае САПР могут служить также виды ПО автоматизированного проектирования.

5.2. Языки описания вычислительных систем

Структурное и поведенческое описания. *Поведенческая модель* показывает реакцию цифрового устройства на изменение входных сигналов с учетом задержки реакции во времени. Эта модель не содержит детального описания аппаратной реализации устройства. Уровень абстракции зависит от уровня детализации описания поведенческой модели. Например, на высшем уровне абстракции поведенческая модель может описывать процессор, выполняющий абстрактный алгоритм, с точки зрения нагрева корпуса и т. п., а на низшем уровне – это может быть модель процессора с детализацией системы команд (множества инструкций) и алгоритмов их выполнения. Точность детализации входных и выходных данных поведенческой модели зависит от уровня абстракции модели.

Функциональная модель устройства описывает его функции без определения способа реализации этих функций. Данная модель позволяет определять реакцию системы или ее компонента без учета временного фактора (определяет значение выхода устройства, но не время его установки). Уровень абстракции зависит от степени детализации модели устройства. Уровень детализации входных и выходных сигналов зависит от уровня абстракции модели в целом.

Структурные модели представляют компоненты устройств с точки зрения их иерархии и взаимосвязей между ними. Структурная модель может отвечать физической иерархии элементов в описываемом цифровом устройстве. Иерархия, в свою очередь, определяется физической организацией конкретной реализации разрабатываемого проекта. Структурная модель описывает физическую структуру конкретной реализации путем определения компонентов и топологии их взаимосвязей. Компоненты структурной модели могут быть описаны на структурном, функциональном или поведенческом уровне. Имитационное моделирование структурной модели цифровых устройств требует наличия поведенческих моделей всех низших уровней иерархии. Степень детализации аспектов модельного времени, значений объектов данных и функциональности структурной модели зависит от степени детализации моделей компонентов.

Модель производительности – данный тип моделей позволяет моделировать временные аспекты работы устройств, т.е. определять скорость реакции модели устройства или его компонента на изменение входного сигнала без вычисления значения выходного сигнала.

Модель интерфейса может содержать детализацию всех аспектов обмена информацией между проектируемым устройством и внешней средой, включая функциональность, временные характеристики, значение данных и т. п. Такая модель не содержит информации о внутренней структуре устройства.

Языки логического управления. В системах логического управления традиционно используются *булевы функции и системы булевых функций*, задаваемые в форме таблиц истинности для полностью определенных функций и таблиц решений для неполностью определенных функций. При этом таблицы истинности, описывающие автоматы с памя-

ю, носят название кодированных таблиц переходов, или кодированных таблиц перехо-в и выходов. Применение таблиц истинности ограничивается задачами небольшой раз-рности, а применение таблиц решений – в основном автоматами без памяти – комбина-ионными схемами. Табличное задание автоматов с памятью ненаглядно.

Аналитической формой представления булевых функций являются булевы фор-улы и системы булевых формул, которые позволяют описывать как комбинацион-ые схемы, так и автоматы с памятью большой размерности. Системы булевых фор-ул могут быть изоморфно реализованы лестничными или функциональными схема-и. Иногда используются также и другие аналитические формы представления буле-ых функций, например, пороговые, спектральные или арифметические. Основным раничением на применение систем булевых формул для автоматов с памятью явля-ся их низкая наглядность.

Широко используются *функциональные схемы* и разного типа *схемы алгорит-ов, сети Петри и графы операций*, рассмотренные в разделе 1.2. На этапе перехода автоматного описания к тексту программ целесообразно применять алгоритмиче-кие языки.

Язык ПРАЛУ параллельных алгоритмов логического управления. Язык РАЛУ удобно использовать для описания систем, характеризующихся сложным аимодействием, асинхронностью и параллелизмом. Он объединяет возможности мо-елей «если–то» с возможностями сетей Петри и обладает средствами для представле-ия последовательности текущих состояний ВС. С помощью языка ПРАЛУ возможно исание временной упорядоченности событий, возникающих при реализации прото-ола обмена сообщениями, абстрагируясь от всех деталей, кроме тех, что выражаются ричинно-следственными и временными отношениями. Алгоритмы на ПРАЛУ пред-тавляются в виде причинно-временных зависимостей между событиями.

Основными операциями языка ПРАЛУ являются операции ожидания и действия. перация ожидания «$-p_i$» сводится к ожиданию наступления некоторого события p_i, редставленного конъюнкцией логических переменных, и ограничивается проверкой словия его истинности, завершаясь после ее выполнения. Операция действия «$\rightarrow A_i$» риводит к наступлению некоторого события, представленного также конъюнкцией огических переменных, в описываемом объекте (каким-то изменениям его состояния) ыполняется в течение некоторого промежутка времени после ее инициализации.

Алгоритм управления представляется неупорядоченной совокупностью предло-ений. Каждое предложение состоит из одной или нескольких одинаково помечен-ых цепочек «μ_l: $\eta_l \rightarrow \nu_l$»; через η_l обозначен некоторый линейный алгоритм, состав-енный из операций языка; μ_l и ν_l – начальная и конечная метки, которыми служат не-устые подмножества из множества $M = \{1, 2, \ldots, m\}$ целых чисел.

Допускается, что $\nu_l = \varnothing$, что обозначается как «$\rightarrow$.» и является концом реализа-ии алгоритма.

Порядок выполнения цепочек алгоритма управления в процессе его реализации ределяется множеством N запуска, его текущие значения $N_t \subset M$. Среди предложе-ий алгоритма выделяется одно – начальное, его метка заносится в N перед реализа-ией алгоритма.

В процессе реализации алгоритма управления цепочки запускаются независимо уг от друга. Если в некоторый момент времени для некоторой цепочки «μ_i: $\eta_i \rightarrow \nu_i$» ыполняется условие $\mu_i \subseteq N_t$ и реализуется событие p_i, с ожидания которого начинает-я цепочка η_i, то она запускается. При этом N_t заменяется на $N_t \setminus \mu_i$, а после заверше-ия цепочки N_t становится равным $(N_t \setminus \mu_i) \cup \nu_i$. Предложенный механизм достаточен ля отображения альтернативного ветвления и распараллеливания процессов.

Синтаксически параллельный алгоритм характеризуется наличием меток $|\mu_i| > 1$, $|\nu_i| > 1$. Альтернативное ветвление обеспечивается ограничением $(i \neq j) \wedge (\mu_i \cap \mu_j \neq \varnothing) \rightarrow (p_i \wedge p_j = 0)$.

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: «⟶ *», «⟶ *γ» и «⟶ *γ'», где $\gamma \subseteq M$. Ее действие заключается в прекращении реализации всех или некоторых (из множества γ или $\gamma' = N_t \setminus \gamma$) активных цепочек алгоритма. Наряду с булевыми в ПРАЛУ допустимо использование и арифметических переменных, в частности введены операции: задержки "– *n*" – выдержки *n* единиц времени; счета событий: «⟶ (*x* = *n*)» – присвоения многозначной переменной *x* натурального значения *n*; «⟶ (*x* +)» и «⟶ (*x* –)» – единичного положительного и отрицательного «⟶ (*x* –)» 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 «μ_i: $-p \to a \to \nu_j$», где μ_i и ν_j – соответственно начальное и конечное состояния перехода, *p* – условие перехода, *a* – выходной сигнал.

Фрагмент ГСА, изображенный на рис. 5.2.1, представится частью таблицы задания автомата с абстрактным состоянием в виде табл. 5.2.1. Здесь первую строку можно совместить с остальными, поскольку объединение условий переходов в последних трех строках представляет собой тождественно выполнимое условие, что является типичным для ГСА. Это подтверждает независимость выходных переменных от входных для автомата Мура. Таким образом, рассматриваемый фрагмент примет вид табл. 5.2.2, что соответствует следующему ПРАЛУ-описанию:

$1 : -\wedge x_1 \to y_1 y_2 \to 1$
$\quad -x_1 x_2 \to y_1 y_2 \to 3$
$\quad -x_1 . \wedge x_2 \to y_1 y_2 \to 2$

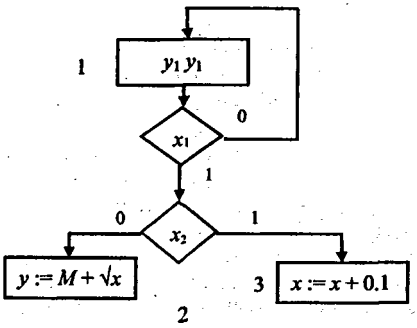


Рисунок 5.2.1 – Фрагмент ГСА

Таблица 5.2.1 – Автомат с абстрактными состояниями

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

Таблица 5.2.2 – Эквивалентное задание автомата с абстрактными состояниями

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

Графовые модели электронных изделий. Для решения задач автоматизированного проектирования сложных ВС и СБИС используют различные математические модели: графовые эквиваленты коммутационных схем соединений отдельных частей объекта (мультиграфы, гиперграфы, ультраграфы); графы пересечений для отображения взаимного расположения соединений; неориентированный топологический граф решетки для задания метрического пространства. Использование графотеоретических моделей схем соединений сохраняет наглядность и содержательность описываемых объектов и позволяет строить формальные алгоритмы обработки этих моделей, которые легко обрабатывается на ЭВМ.

Любую функциональную или принципиальную схему объекта можно представить состоящей из множества $E = \{e_1, e_2, \ldots, e_n\}$ элементов, множества $\{c_i\}$ выводов (контактов) каждого элемента e_i и множества цепей (эквипотенциальных контактов), соединяющих элементы схемы.

В зависимости от целей проектирования решаемых задач можно использовать различную информацию об исследуемой схеме. Получаемые при этом формальные представления будут иметь разный содержательный смысл. Рассмотрим наиболее потребительные 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, но имеет большое практическое значение: мультиграфы коммутационных схем применяются при решении задач декомпозиции и оптимального размещения элементов в пространстве.

Для построения мультиграфа коммутационной схемы необходимо внешние контакты схемы (входные и выходные контакты, соединяющие данную схему с другими) объединить в один дополнительный элемент, соответствующий разъему платы или внешним выводам интегральной микросхемы и поставить в соответствие каждому элементу схемы вершину мультиграфа. Ребро $u_{ij} = (x_i, x_j)$ мультиграфа схемы $G(X,U)$ будет указывать на наличие связей между вершинами x_i, x_j, соответствующими элементам схемы.

На рис. 5.2.2 приведен условный фрагмент коммутационной схемы, состоящий из пяти элементов $E = \{e_1, e_2, \ldots, e_5\}$ и шести цепей $V = \{v_1, v_2, \ldots, v_6\}$. Контакты $\{C_i\}$ каждого элемента имеют сквозную нумерации. Данному фрагменту схемы соответствует математическая модель в виде мультиграфа, представленного на рис. 5.2.3.

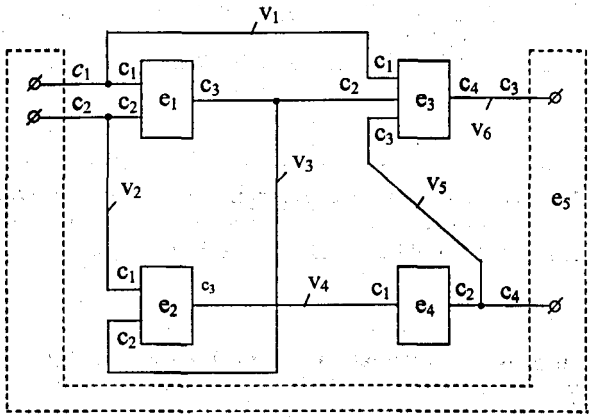


Рисунок 5.2.2 – Фрагмент коммутационной схемы

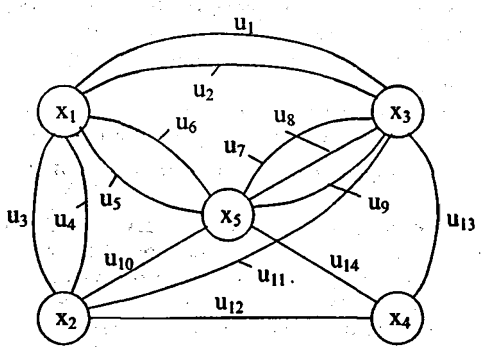


Рисунок 5.2.3 – Мультиграф коммутационной схемы

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

Автомат задается в виде последовательности S множеств $S(n) = \{t(n), U(n), q(n), V(n)\}$, где

n – номер кванта времени, $n = 1, 2, ...$;

$t(n)$ – 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. 5.2.4, представляется в виде последовательности, которую для наглядности представим в виде таблицы 5.2.3.

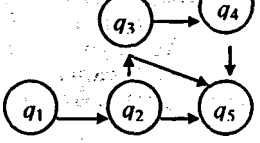


Рисунок 5.2.4 – Диаграмма автомата

Таблица 5.2.3 – Временная диаграмма

n	$t(n)$	$U(n)$	$q(n)$	$V(n)$
1	3	$\wedge u_1$	q_1	-
2	2	u_1	q_2	v_1
3	2	u_1	q_3	v_2
4	2	u_2	q_5	v_1
5	4	-	q_2	-
6	2	u_1	q_3	v_1
7	2	u_1	q_4	-
8	2	-	q_5	v_2
9	2	-	q_2	-

Строка 1 описывает цикл (1,1), строка 2 – линейный участок (1,2), строки 3-5 – цикл (2,3, 5, 2), строки 6-9 – цикл (2, 3, 4, 5, 2).

Язык *Verilog* был разработан фирмой *Gateway Design Automaton в 1995 г.* как внутренний язык симуляции ВС. К этому времени уже успел получить широкое распространение другой язык высокого уровня для описания принципиальных схем – *VHDL* (*Very high-speed IC Hardware Description Language*), появившийся еще в 1987 г. Несмотря на похожие названия, *Verilog HDL* и *VHDL* – различные языки. *Verilog* – достаточно простой язык, сходный с языком программирования *C* как по синтаксису, так и по «идеологии». Малое количество служебных слов и простота основных конструкций упрощают изучение и позволяют использовать *Verilog* в целях обучения. Но в то же время это эффективный и специализированный язык.

Язык Verilog поддерживает четыре уровня абстракции моделей:

- вентильный уровень;
- логический уровень;
- поведенческий, или алгоритмический, уровень;
- уровень потоков данных.

Поведенческий, или алгоритмический, уровень – это наивысший допустимый в Verilog уровень. На этом уровне модель описывается в виде алгоритма работы устройства, без детализации его аппаратной реализации. Разработка моделей на этом уровне подобна традиционному программированию на языке Си.

Уровень потоков данных – предполагает построение функциональной модели вырабатываемого устройства на основе предварительно формируемого графа потоков данных. Механизм задержек, включенный в языковый стандарт Verilog, позволя-

ет устанавливать время передачи сигнала по ребрам графа и, таким образом, модели ровать временные параметры устройства.

Логический уровень – этот уровень, как видно из его названия, предполагае формирование функционально-структурной модели устройства в виде цепей из стан дартных логических элементов (например, И, ИЛИ и Исключающее ИЛИ и т. п.). Та кие элементы реализованы в виде стандартных модулей в языке Verilog и могут быт включены в любую пользовательскую программу.

Вентильный уровень – это наиболее низкий из возможных в языке Verilo уровней абстракции. При разработке моделей на этом уровне проектировщик опери рует такими понятиями, как МОП-транзистор, КМОП-транзистор и т. п. Средств синтеза, разработанные для интегральных микросхем FPGA-типа, не позволяют син тезировать логические цепи на базе моделей вентильного уровня. В то же время таки модели полезны для понимания и исследования физических процессов, протекающи в цифровых цепях.

Отдельно выделяют также **уровень регистровых передач** (Register transfe level), представляющий собой комбинацию первых трех уровней абстракции. Verilog модели, разработанные на уровне регистровых передач, включают только поддержи ваемые средствами синтеза операторы языка.

Наиболее распространенная методология проектирования цифровых электрон ных устройств «сверху-вниз» или «от сложного к простому» предполагает структур ную декомпозицию устройства верхнего уровня, или, другими словами, разбиение его на составляющие компоненты, выделяемые по функциональным или топологическим соображениям. Полученные структурные компоненты, в свою очередь, могут быт разделены на более мелкие составляющие либо реализованы операторами языка Veri log на поведенческом (алгоритмическом) уровне. Следует понимать, что «на дне» лю бой структурной иерархии находятся операторы Verilog, описывающие алгоритм функционирования объекта. Именно последнее определяет основное отличие между поведенческим подходом к проектированию и схемотехническим подходом. При схемотехническом проектировании на нижнем уровне структурной иерархии нахо дится модель устройства, представленная в виде структурной схемы, включающей логические вентили, триггеры, регистры, мультиплексоры и другие стандартные циф ровые устройства.

Язык *VHDL* обладает большей универсальностью и может быть использован не только для описания моделей цифровых электронных схем, но и для других моделей (например, модели экосистемы). Однако из-за своих расширенных возможностей VHDL проигрывает в эффективности и простоте, то есть на описание одной и той же конструкции в Verilog потребуется в 3–4 раза меньше символов (*ASCII*), чем в VHDL. Наряду с языком Verilog, он является базовым языком при разработке аппаратуры со временных ВС.

VHDL – более универсальный и гибкий язык, но он проигрывал в быстродейст вии языку Verilog, особенно при моделировании на уровне вентилей и транзисторов. VHDL получил широкое распространение в университетах и исследовательских уч реждениях, так как это строгий, стройный, универсальный и расширяемый язык. Так, например, появились пакеты VHDL для аналогового моделирования, моделирования многозначной логики. Кроме того, симуляторы VHDL были гораздо дешевле симуля торов Verilog. Все современные САПР микроэлектроники имеют компиляторы как с Verilog, так и с VHDL. Программист, освоивший VHDL, без особого труда может пе рейти к программированию на языке Verilog.

Язык описания схем VHDL (Very high speed integrated circuits Hardware De scription Language),. Язык описания аппаратуры для высокоскоростных интегральных схем, называемый VHDL является формальной записью, которая может быть исполь зована на всех этапах разработки электронных систем – алгоритмическом, структур-

ом, регистровых передач (*RTL*) и потоков данных (dataflow), логическом, аналого-
их схем. Вследствие того, что язык легко воспринимается как машиной, так и чело-
ком он может использоваться на этапах проектирования, верификации, синтеза и
стирования аппаратуры также как и для передачи данных о проекте, модификации и
провождения. VHDL предназначен для описания функции и логической организа-
ии цифровой системы. Функция системы определяется, как преобразование значе-
ий на входах в значения на выходах. Причем время в этом преобразовании задается
но. Организация системы задается перечнем связанных компонентов.

Проект на VHDL – объединение структуры ВС и алгоритма его функционирования.

Для ВС, описанной на VHDL, необязательно выполнять проверку правильности
о функционирования, например, путем его макетирования. Чтобы определить, пра-
льно ли ВС выполняет заданный алгоритм, достаточно его VHDL-программу за-
стить на исполнение в симуляторе VHDL. Соответствующие САПР преобразуют
HDL-описание в комплект документации для изготовления работоспособного уст-
йства.

Синтаксический анализ, моделирование и компиляция в логическую схему бы-
ро выявляют ошибки проекта.

Далее, разработанный однажды вычислительный блок может быть использован
многих других проектах. При этом многие структурные и функциональные пара-
етры блока могут быть настраиваемыми (параметры разрядности, объема памяти,
ементная база, состав блока и структура межсоединений).

Проект ВС на VHDL может быть повторно использован через несколько лет.
орошее техническое решение (например, изобретение), описанное на VHDL, может
ть востребованным в течение десятилетий.

Из четырех конструкций верхнего уровня языка VHDL: Entity, архитектуры
rchitecture body), пакета и конфигурации на любой стадии проектирования обяза-
льно используются две: Entity и архитектура. Entity содержит описание интерфейса
ежду проектируемым устройством и внешним миром. Архитектура содержит описа-
ие алгоритма работы проектируемого устройства. Так как для выполнения одной и
й же функции могут быть предложены различные алгоритмы, а кроме того, один
горитм может быть записан различными способами, одному Entity может соответ-
вовать несколько архитектур.

Стили описания архитектур. При описании архитектуры средствами языка
HDL могут использоваться 3 стиля, которые обеспечиваются разными конструк-
циями языка:

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

При поведенческом описании алгоритм записывается последовательными опера-
рами (аналогичные которым имеются в большинстве языков программирования).

При потоковом описании архитектура представляемся в виде множества парал-
льных регистровых операций.

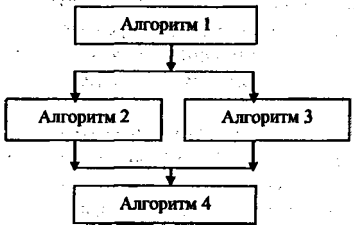
При структурном описании проект представляется в виде множества компонент,
ждому из которых соответствует своя пара Entity-архитектура. При этом должны
ть указаны все связи между компонентами, т. е. для любого входа любого компо-
нта (Entity) должно быть указано, с какого выхода какого компонента поступает
гнал на этот вход.

При этом для описания архитектуры компонента (проекта нижнего уровня) мо-
т использоваться любой из трех вышеперечисленных стилей и их комбинация. Та-
м образом, может быть построено целое дерево компонент, на нижнем уровне ко-

торого используются уже известные (библиотечные) элементы. (Пример такого описания приведен ниже).

Поведенческий стиль описания является основным для пользователей САПР, так как именно он используется для создания VHDL-описания, поступающего на вход САПР.

Рассмотрим следующий алгоритм с параллельным выполнением алгоритмов 2 и 3.



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 выполнения алгоритмов 2 и 3 начнется выполнение алгоритма 4. После выполнения алгоритма 4 работа приостанавливается до изменения значений входных сигналов схемы. Когда изменится какой-либо из входных сигналов, снова стартует процесс M1 и весь цикл повторится.

В качестве иллюстрации различных стилей описания архитектур рассмотрим схему подсчета числа единиц во входном битовом векторе *A*, имеющем длину, равную 3. Описание ее интерфейса имеет вид:

```
entity ONES_CNT is
port (A: in bit_vector (0 to 2);
C: out bit_vector (0 to 1));
end ONES_CNT;
```

Потоковое описание этой схемы использует параллельные операторы назначения сигнала, которые, как и остальные операторы в теле архитектуры, выполняются параллельно:

```
architecture POTOK of ONES_CNT is begin
C(1)<=( A(1) and A(0) ) or (A(2) and A(0) ) or (A(2) and A(1) );
C(0)<=( A(2) and not A(1)  and not A(0) )
or (not A(2) and not A(1)  and A(0) )
or (A(2) and  A(1)  and A(0) )
or (not A(2) and A(1)  and not A (0) );
end POTOK;
```

Поведенческое описание является основным способом задания входной информации для САПР. В этом случае архитектура содержит один процесс или несколько процессов, выполняемых параллельно. В свою очередь каждый процесс содержит операторы, выполняемые последовательно. Поведенческое описание данной схемы может иметь, в частности, следующий вид:

```
architecture ALG1 of ONES_CNT is
 begin process (A)
variable NUM: integer range 0 to 3;
 begin NUM:=0;
for  I in 0 to 2 loop
if A(I)='1' then NUM:=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:

```
architecture ALG2 of 0NES_CNT is begin
process (A)
 begin
C(1)<=( A(1) and A(0) ) or (A(2) and A(0) )
or (A(2) and A(1) );
 C(0)<=( A(2) and not A(1)  and not A(0) )
or (not A(2) and not A(1)  and A(0) )
or (A(2) and  A(1)  and A(0) )
or (not A(2) and A(1)  and not A(0) );
end process;
 end ALG2;
```

Описание автоматов. Описания автоматов на языке VHDL покажем на примере автомата 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;    -- inputs
    y : out integer range 1 to 2 );   -- outputs
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;
```

5.3. Технологии производства вычислительных систем

Стадии проектирования. Проектирование СБИС базируется на ее иерархическо декомпозиции, в основу которой положена концепция абстракции, при этом возможн разделение в двух направлениях: снизу-вверх и сверху-вниз. В зависимости от уров абстракции, СБИС представляется на системном, регистровом, вентильном и физич ском (библиотечных элементов кристалла) уровнях. Переход на очередной уровень а

тракции выполняется после ряда итераций (рис. 5.3.1). Перепроектирование – подход, зволяющий выполнить оптимизацию схем путем ее реконфигурирования.

В традиционной технологии проектирования проект продвигается поэтапно от дного уровня к другому и не происходит возврат на предыдущие уровни. Недостаток той методологии проектирования в том, что с увеличением сложности проекта увеличивается опасность появления ошибок и затрудняется процесс их поиска. Ошибки, бнаруженные в конце той или иной стадии проектирования, ведут к повторному ее ыполнению, что в ряде случаев влечет за собой неоднократный выпуск прототипов БИС, приводит к значительному замедлению сроков выполнения проекта и резко овышает его стоимость. Более того, для схем, изготавливаемых по субмикронной ехнологии, такой маршрут вообще не будет работать, поскольку особенности физической реализации должны учитываться уже на логическом уровне проектирования. Поэтому в последнее время используется методология Specify-Explore-Refine («опитал – опробовал – доделал»). После этапа постановки задачи

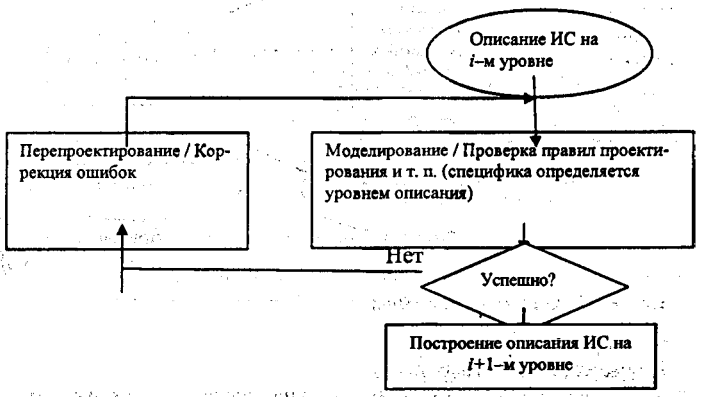


Рисунок 5.3.1 – Схема перепроектирования на i–м уровне абстракции

пецификации исходных требований), на стадии ее выполнения, происходит оценка азличных элементов системы для реализации функциональных возможностей в пределах указанных конструкторско-технологических ограничений (КТО). Технические ебования модифицируются на стадии доводки проекта в соответствии с решениями, ыполненными на стадии реализации. Маршрут проектирования при этом представется в виде спиралевидной модели, согласно которой проектирование выполняется новременно по четырем направлениям: разработка ПО, разработка RTL-кода, логический синтез, физический синтез. В процессе работы группы разработчиков обмениваются результатами проектирования. Существенным является то, что разрешен возврат на предыдущие стадии проектирования и корректировка результатов.

Широко используемая технология обратного проектирования СБИС интегрирует себе названные выше подходы с тем лишь отличием, что процессу проектирования редшествует этап восстановления описания ИС на физическом уровне.

Под обратным проектированием понимается процесс анализа системы для идентификации ее компонент и их взаимосвязей и создание описания в другой форме или более высоком уровне абстракции. Применительно для СБИС этот процесс используется для верификации корректности проекта путем получения изображения тологии и построения транзисторной схемы СБИС. Затем по этой схеме представля СБИС на вентильном и функциональном уровнях. Полученная функциональная дель снова верифицируется и сравнивается с исходным проектом. Обратное проек-

тирование применяется, когда частично или полностью неизвестно поведение, логи ческая структура, топологическая структура или процесс реализации. В таких случая может быть интересно проверить неизвестные аспекты и спроектировать лучшу схему. Другая причина заключается в более точной проверке функционировани СБИС и диагностики ее неисправностей. Особенно это актуально сейчас, когда пр проектировании используется VHDL и САПР высокого уровня и описание современ ных СБИС на транзисторном уровне отсутствует.

Методология обратного проектирования представлена на рис. 5.3.2.

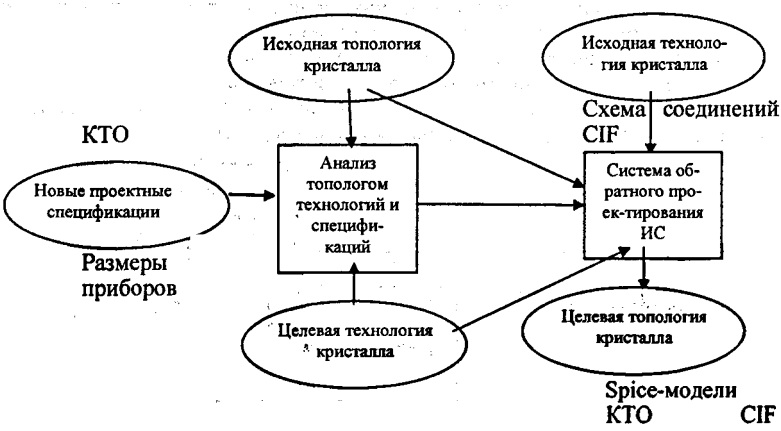


Рисунок 5.3.2 – Информационные потоки при обратном проектировании СБИС

Стадии проектирования разделяют на составные части, называемые проектным процедурами. Примерами проектных процедур может быть моделирование, оптими зация параметров и другие проектные задачи. В свою очередь, проектные процедур можно разделить на более мелкие компоненты, называемые проектными операциями например, представление моделирования в графической и текстовой формах.

Типовые проектные процедуры. Создать проект объекта (изделия или проце са) означает выбрать структуру объекта, определить значения всех его параметров представить результаты в установленной форме. Результаты (проектная документа ция) могут быть выражены в виде чертежей, схем, пояснительных записок, програм для программно-управляемого оборудования и других документов на бумаге или н машинных носителях информации.

Разработка или выбор структуры объекта есть проектная процедура, называема *структурным синтезом*, а расчет или выбор значений параметров элементов – про цедура *параметрического синтеза.*

Задача структурного синтеза формулируется в системотехнике *как задача при нятия решения* (ЗПР). Суть ее заключается в определении цели, множества возмож ных решений и ограничивающих условий.

Классификация ЗПР осуществляется по ряду признаков. По числу критерие различают задачи одно- и многокритериальные, по степени неопределенности – де терминированные и ЗПР в условиях неопределенности (при наличии в формулировк случайных параметров, при неполноте и недостоверности исходной информации).

Реальные задачи проектирования, как правило, являются многокритериальными Одна из основных проблем постановки многокритериальных задач – установлени правил предпочтения вариантов. Способы сведения многокритериальных задач к од

юкритериальным и последующие пути решения изучаются в дисциплинах, посвященных методам оптимизации и математическому программированию.

CASE-технология представляет собой методологию проектирования ВС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ВС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

В результате реальный процесс создания ПО принимал следующий вид (рис. 5.3.3):

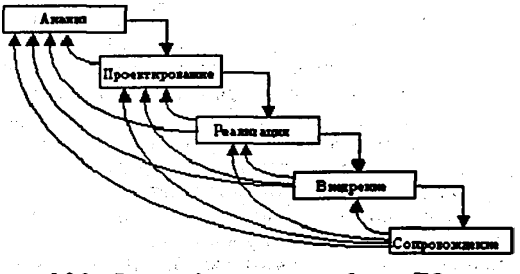


Рисунок 5.3.3 – Реальный процесс разработки ПО по каскадной схеме

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Для преодоления перечисленных проблем была предложена спиральная модель жизненного цикла, делающая упор на начальные этапы жизненного цикла: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации. Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Технология разработки систем на кристалле. Система на кристалле (СНК, *System-on-a-Chip, SoC.*) представляет собой электронную схему, выполняющую функции целого устройства (например, ЭВМ) и размещенную на одной интегральной схеме. ПО для функционирования СНК не менее важно, чем аппаратное обеспечение. Разработка, как правило, ведётся параллельно. Аппаратная часть собирается из стандартных отлаженных блоков, для сборки программной части используются готовые подпрограммы настройки соответствующих блоков, реализующие необходимые процедуры и функции, (драйвера).

Наиболее трудоемкими и ответственными этапами разработки СНК выступают этапы структурного проектирования и верификации соответствия ВС заданным алгоритмам функционирования. Поэтому эффективность САПРов микросхем и производительность разработчиков, выполняющих проектирование на уровне регистровых пере-

регистровых передач, постоянно растет приблизительно на 20% в год. Но начиная с се редины 90-х годов, производительность разработчиков стала заметно отставать от рост сложности СНК.

Первым направлением улучшения технологии разработки СНК, направленным н уменьшение зазора между ростом производительности проектирования на уровне реги стровых передач и ростом сложности СНК, является применение крупных библиотеч ных вычислительных модулей (Intellectual Property Cores, IP Cores). Эти модули долж ны быть надежно повторяемыми и настраиваемыми под решаемые задачи в ряде про ектов СНК. Повторное применение таких модулей (IP Core reuse), которые можно на звать вычислительными заготовками за их функциональную и технологическую адап тируемость, позволяет уменьшить трудозатраты и сроки проектирования СНК.

Второе направление – это разработка САПР совместного проектирования аппа ратно-программного обеспечения (Hardware-Software Codesign). Архитектура СНК как правило, включает в себя микропроцессорное ядро с периферийными устройст вами в различном сочетании. Обычно процесс разработки ВУ с такой архитектурой состоит из трех последовательных этапов: разработки математического обеспечения микропроцессора, проектирования электрической схемы аппаратуры и стыковки ма тематического обеспечения с аппаратурой. Для ускорения проектирования разраба тывают САПР, которая не только обеспечивает совместное выполнение этих этапов, но и моделирование работы СНК и ее верификацию в комплексе.

Наибольшее ускорение разработки СНК может дать внедрение САПР непосред ственного отображения алгоритмов в аппаратуру, т. е. САПР системного проектиро вания. Например, такая САПР может включать в себя трансляцию программы с языка высокого уровня, например, C++ , с автоматическим разделением вычислительных задач между микропроцессорным ядром и спецпроцессорами и другими периферий ными устройствами.

В крупных фирмах, долгие годы занимающихся разработкой СБИС, а теперь и СНК, наработаны большие библиотеки стандартных модулей, как-то: ОЗУ, АЛУ, пе риферийные устройства. В новых проектах СНК некоторые блоки приходится разра батывать заново, а остальные – берутся из библиотеки. При этом если модуль неясно описан, не имеет хорошего интерфейса, документации, комментариев, испытательно го стенда с надежными тестами, то он повторно применяться не будет. Если такой модуль изначально оформлен в виде вычислительной заготовки, то он будет без лиш них проблем вставляться в любой новый проект. Более того, лицензию на него можно предлагать другим фирмам-разработчикам СНК. Вычислительные заготовки разли чаются по степени гибкости своей настройки под условия потребителя как:

– гибкие (описанные языком описания аппаратуры, таком как VHDL, на уровне регистровых передач);

– жесткие (логическая схема, EDIF-файл);

– твердые (маски под определенную технологию, прошивки ПЛИС).

Гибкие заготовки обычно подстраиваются к условиям нового проекта в широких пределах и независимы от его технологии (серия ПЛИС, технология СБИС). Минимизация аппаратурных затрат вычислительных заготовок обеспечивает не только уменьшение стоимости СНК, но и уменьшение его энергопотребления, что является важным фактором для портативных и энергонезависимых приложений.

В сегодняшних условиях, чтобы быстрее перейти от идеи к «железу», эффектив нее провести проектирование новой СНК, необходимо эту СНК «собрать» из имею щихся вычислительных заготовок, а отсутствующие – приобрести на рынке IP-cores, который бурно развивается в последние годы.

Проблема функциональной верификации. Степень интеграции современных СБИС растет экспоненциально. Сегодня переход к субмикронным технологическим нормам позволяет размещать на кристалле практически любой по сложности проект. Тем не менее, имеется сдерживающий фактор – проблема функциональной верифи-

ающих не только цифровые, но и аналоговые, смешанные, а также процессорные яд-а со встроенным ПО, означенная проблема стоит наиболее остро. По последним анным, примерно половина всего инженерного состава, работающего над крупными роектами, занята функциональной верификацией, а временные затраты на нее в об-ем цикле проектирования впечатляют еще сильнее – более 60 %.Анализ проблемы оказал, что технологии верификации проекта на нынешний день заметно отстают от ехнологий и вычислительных возможностей систем проектирования. Еще более ерьезным видится отставание возможностей верификации от технологических воз-ожностей производства СБИС.

В то же время затраты на создание комплекта фотошаблонов для субмикронных БИС настолько высоки, что их повторное изготовление из-за обнаруженных ошибок частую недопустимо как с точки зрения задержки выхода изделия на рынок, так и в вязи с ростом его конечной стоимости. Поэтому необходим поиск принципиально овых решений, разработка передовых технологий верификации и подход к новым етодам анализа проекта.

Основным недостатком традиционного подхода к функциональной верификации вляется общепринятый алгоритм проектирования. Дело в том, что здесь система начала разрабатывается и лишь затем – тестируется. При 60-70 % загрузки по време-и на верификацию такая последовательность действий недопустима. При увеличе-ии числа транзисторов на кристалле на первое место выходит факт резкого увеличе-ия необходимого числа и длины тестовых векторов и, как следствие, размера и ложности программно-аппаратных средств тестирования. Параллельно усложняется роцесс поиска неисправности, обнаруженной тестом.

Современные средства проектирования должны обеспечивать сквозную верифи-ацию на всех уровнях по преимущественным направлениям. Здесь необходимо ис-ользовать технологии имитационного моделирования, аппаратной эмуляции, интег-ированной программно-аппаратной верификации и обязательно аналого-цифрового мешанного моделирования. Программные системы создания и верификации проекта олжны поддерживать все стандартные языки проектирования, в том числе весь ряд DL (VHDL, Verilog, VHDL_AMS, Verilog_A). Также должны поддерживаться сис-емные языки (Spice, C, C++, SystemC, System Verilog, MATLAB, PSL assertions и др.)

На стадии проектирования, особенно в САПР, предпочтение отдается цифровым оделям, исходя из общности технических средств проектирования и моделирования, нформационной базы данных на всех этапах проектирования, большого быстродей-твия и высокой точности расчетов современных ЭВМ, широких возможностей на-лядного, графического представления моделируемых процессов.

Основные методы исследования математических моделей:

– аналитическое исследование;

– имитационное моделирование.

В зависимости от принадлежности к тому или иному иерархическому уровню ыделяют модели системного, функционально-логического, макроуровня (сосредото-енного) и микроуровня (распределенного). Кроме того, используются понятия пол-ых моделей и макромоделей, моделей статических и динамических, детерминиро-анных и стохастических, аналоговых и дискретных. Особое место занимают геомет-ические модели, используемые в системах конструирования.

5.4. Инструментальные средства проектирования

Универсальные программные средства проектирования. В настоящее время рактически все современные САПР имеют встроенные функции символьных вычис-ений. Однако наиболее известными и приспособленными для математических сим-ольных вычислений считаются Maple, MathCad, Mathematica и MatLab. Отметим, что ектр задач, решаемых подобными системами, очень широк:

– проведение математических исследований, требующих вычислений и аналитических выкладок;
– разработка и анализ алгоритмов;
– математическое моделирование и компьютерный эксперимент;
– анализ и обработка данных;
– визуализация, научная и инженерная графика;
– разработка графических и расчетных приложений.

Ниже приведен краткий обзор возможностей систем MathCad и MatLab, рекомендуемые при обучении.

Пакет MathLab. Система MatLab относится к среднему уровню продуктов, предназначенных для символьной математики, но рассчитана на широкое применение в сфере инженерного проектирования. MatLab построена на расширенном представлении и применении матричных операций. Это нашло отражение и в самом названии системы – MATrix LABoratory, то есть матричная лаборатория.

Несмотря на то, что изначально MatLab предназначалась исключительно для вычислений, в процессе эволюции к ней была подключена библиотека Simulink, позволяющая построить логическую схему сложной системы управления из одних только стандартных блоков, не написав при этом ни строчки кода. После конструирования такой схемы можно детально проанализировать ее работу.

Благодаря тесной интеграции с MATLAB, Simulink имеет непосредственный доступ к широкому диапазону средств проектирования и анализа. Традиционный подход к проектированию систем обычно заключается в создании прототипа, за которым следует всестороннее тестирование и внесение соответствующих изменений. Этот подход требует больших временных и финансовых затрат. Эффективной и общепринятой альтернативой является имитационное моделирование. Simulink – мощный инструмент для моделирования, обеспечивающий быстрое построение и тестирование виртуальных прототипов, и дающий доступ к любому уровню детализации проекта с минимальными усилиями. Используя Simulink для итеративного исправления проекта до построения

В системе MatLab также существуют широкие возможности для программирования. Ее библиотека C Math (компилятор MatLab) является объектной и содержит свыше 300 процедур обработки данных на языке С. Внутри пакета можно использовать как процедуры самой MatLab, так и стандартные процедуры языка С, что делает этот инструмент мощнейшим подспорьем при разработке приложений (используя компилятор C Math, можно встраивать любые процедуры MatLab в готовые приложения).

С системой MATLAB могут интегрироваться другие популярные математические системы, такие как Mathcad, Maple V и Mathematica. Есть тенденция и к объединению математических систем с современными текстовыми процессорами. Так, новое средство последних версий MATLAB — Notebook — позволяет готовить документы в текстовом процессоре Word 95/97/2000 со вставками в виде документов MATLAB и результатов вычислений, представленных в численном, табличном или графическом виде. Таким образом, становится возможной подготовка «живых» электронных книг, в которых демонстрируемые примеры могут быть оперативно изменены.

В MATLAB задачи расширения системы решаются с помощью специализированных пакетов расширения — наборов инструментов (Toolbox). Многие из них содержат специальные средства для интеграции с другими программами, поддержки объектно-ориентированного и визуального программирования, для генерации различных приложений.

Пакет MATHCAD. MATHCAD – универсальный математический пакет, предназначенный для выполнения инженерных и научных расчетов. Математическое обеспечение пакета позволяет решать многие вычислительные задачи с различными с произвольной точностью типами данных (комплексные, векторы, матрицы).

Основное преимущество пакета перед типичными языками программирования — стественный математический язык, на котором формулируется решаемая задача.

Пакет объединяет в себе: редактор математических формул, интерпретатор для ычислений, библиотеку математических функций, процессор символьных преобра-ований, текстовый редактор, графические средства представления результатов. Па-ет MATHCAD относится к интегрированным пакетам, т. е. позволяет не только про-вести вычисления, но и получить документ – итоговый отчет с комментариями, ормулами, таблицами и графиками. В отличие от издательских систем формулы в ATHCAD работают!

К положительным качествам MATHCAD следует отнести открытость – все при-денное в документе может быть воспроизведено, а интеграция в одном документе сходных данных, метода решения и результатов позволяет сохранить настройки для ешения подобных задач.

Литература

[2, 3, 8, 11, 12, 14, 15, 16, 25, 40]

Контрольные вопросы

1. Укажите специфические свойства языков логического управления.
2. Какие главные отличительные свойства языка ПРАЛУ.
3. В чем заключается преимущества графовых моделей электронных изделий?
4. Почему язык VHDL получил широкое распространение в проектировании вы-ислительны систем?
5. Охарактеризуйте, в чем выражается полезность использования систем МАТ-АВ и MATHCAD.

Глава 6. РЕАЛИЗАЦИЯ СЕТЕВЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

6.1. Основные понятия об интеллектуальных информационных технологиях

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 ВС для игры в шахматы демонстрируют результаты на уровне чемпиона мира. Увлечение ученых созданием шахматных программ базируется на уверенности, что полученные здесь результаты приведут к прорыву и в других областях деятельности человека, т.е. шахматы в вычислительной технике рассматриваются как подопытный кролик в медицине. Основная причина выбора такого объекта состоит в том, что правила игры формализованы, накоплен значительный багаж сыгранных гроссмейстерами шахматных партий и окончаний, создана теория стратегии и тактики игры, что позволило создать системы принятия

решений и копирования лучших достижений в игре за счет громадной базы данных и знаний, а также быстродействия суперкомпьютера. Это достижение явилось плодом работы многих специалистов в области шахмат, теории принятия решений, «искусственного» интеллекта, программирования, передачи и обработки данных, сетей связи, математиков, электронщиков и многих других.

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 в юридическом плане как обычных документов, так и «электронных». Предполагается, *что* при использовании сертифицированных процедур электронной цифровой подписи электронные документы будут эквивалентны обычным и будут иметь такую же доказательную силу.

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

Подписание электронных документов, кроме того, тесно связано с вопросами сохранения коммерческой тайны и часто основывается на применении аналогичных математических методов шифрования информации. Действуя в этом направлении, Национальный банк Республики Беларусь издал инструкцию о порядке передачи служебной информации ограниченного распространения в электронном виде по открытым каналам связи. Пока еще она при реализации имеет под собой договорную основу между банками РБ и Национальным банком РБ, который устанавливает в других банках свой программно-технический комплекс, который обеспечивает конфиденциальность передачи информации с клиентских рабочих мест других банков. Под управлением центрального банка проводится как установка клиентского программного обеспечения, так и генерация открытых и тайных личных ключей шифрования. Электронная цифровая подпись осуществляется на клиентском рабочем месте ответственным за него работником. Чтобы избежать злоупотреблений с повторами и перехватами сообщений, пользователем указывается системное время отправки документа и выполнения наложения ЭЦП на него. Подписанное зафиксированное сообщение отсылается клиенту, от кото-

рого 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 (микропроцессора) и фотографии клиента, так как эти элементы повышают защищенность пользователя от кражи или потери карты, а также от использования ее другими лицами в магазинах и т. п.

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 помещения и временного использования компьютера работника, особенностями налогообложения и др.

Кроме того, возникает и ряд правовых проблем: следует регулировать отношения между работником и работодателем, вопросы по налогам за двоякое использование оборудования для домашнего быта и предприятия и т.п.

Остановимся на некоторых из этих особенностей, когда рабочее место располагается в квартире работника (на дому). В этом случае работник может иметь собственный компьютер с возможностью подключения к сетям связи. 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., она доступна для инвалидов.

Иногда встречается и промежуточный вариант телеработы, когда рабочее место может находиться в квартире работника или на предприятии с регламентацией времени использования того 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. выявилась потребность в 875 000 дистанционных рабочих мест, из которых 500 000 относились к мобильной работе, 350 000 – к работе на альтернативной основе частично дома и на предприятии, 22 000 – к чисто дистанционной работе и 3 000 – к прочим видам работ.

Широко распространилась дистанционная работа в США 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 жилой 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 потенциаль-

ных дистанционных работников в Евросоюзе 8 миллионов, из которых только в Германии около 2,5 миллионов.

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

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

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

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

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

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

Из перечисленных направлений развития исследований и прикладных работ следует выделить особо важные ввиду их всеобщей применимости: защита и безопасность данных; общие аспекты создания и функционирования дистанционных рабочих мест (экономические, правовые и информационные); использование автоматизированных средств идентификации личности с учетом уникальных физических особенностей человека; создание средств для современной рекламы в сетях с использованием медиасредств и демонстрационных версий; автоматизация дистанционной оплаты коммунальных услуг, получения различных справок и документов, оплаты покупок, предварительных заказов на различные услуги, управления банковскими счетами и т.п.

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

6.2. Нейронные системы и нейрокомпьютеры в сетях

Причины развития работ по введению интеллектуальных компонентов в высо-опроизводительные и обычные вычислительные сети – это в первую очередь обес-ечение гибкости новых структур в динамике.

Ставка на параллелизм вычислений за счет использования жестко связанных ком-онентов сети (постоянное соединение) и ориентация на узкие классы задач исчерпала ебя при переходе к решению более широких классов задач, для которых возникла про-лема разработки методов алгоритмизации задач с учетом особенностей системы и орга-изации вычислений с распараллеливанием; трудоемкое создание трансляторов и обуче-ия кадров программистов; проблемы переносимости старого и вновь созданного про-раммного обеспечения на другие машины; появились трудности в организации соеди-ений между отдельными узлами сети с учетом конкретной задачи.

Поэтому взоры исследователей обратились к идее использования в системе эле-ентов искусственного интеллекта. Привлекательной стороной явились два основных омента: возможность адаптации к решению нужного класса задач методами искусст-енного интеллекта без больших усилий человека и возможность без разработки алго-итма и программы после настройки системы получить алгоритм и программу в скры-ой форме в виде соответствующей нейросети для реализации на нейрокомпьютере.

Нейрокомпьютером (НК) называют ЭВМ (аналоговую или цифровую), основной перационный блок (центральный процессор), который построен на основе нейрон-ой сети и реализует нейросетевые алгоритмы. НК качественно отличается от дейст-ующих классических систем параллельного типа тем, что для решения задач испол-уются не заранее разработанные алгоритмы, а специальная нейронная сеть, обучение оторой проводится для решения избранного класса задач на специально подобран-ых примерах.

В качестве важных направлений развития НК выделяют следующие четыре:

1. Решение традиционных задач искусственного интеллекта: распознавание об-азов, классификация (извлечение знаний из данных и т. п.). На этой основе создают-я модели искусственных органов человека: искусственный глаз, ухо, нос и др.

2. Решение сложных вычислительных задач (систем линейных уравнений, моле-улярное конструирование лекарств, с получением легкого распараллеливания рабо-ы алгоритмов на основе НС).

3. Использование НК как инструмента для моделирования работы структур че-овеческого мозга.

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

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

НК в основном используются для решения задач третьего класса (в распознава-ии рукописных и печатных символов при оптическом вводе текстов в ЭВМ, речи, в адачах страхования и учета рисков, аппроксимациях неизвестных в явном виде ункций и т.п.).

Опишем общую структурную схему абстрактного НК (рис. 6.2.1).

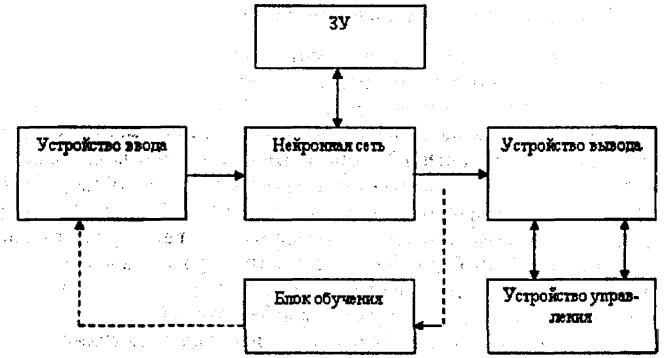


Рисунок 6.2.1 – Структурная схема абстрактного НК

Операционным блоком НК (процессором) является искусственная НС, состоящая из формальных нейронов, соединенных каналами передачи информации. Этот блок в обычном понимании не производит вычислений. Он трансформирует входной сигнал (образ) в выходной в соответствии с топологией НС и значениями коэффициентов межнейронной связи. В ЗУ хранится не программа решения задачи, а программа изменений коэффициентов связи между нейронами. Устройства ввода и вывода информации выполняют обычные функции. Устройство управления служит для синхронизации работы всех структурных блоков НК при решении конкретной задачи.

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

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

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

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

В качестве транспортных средств могут использоваться мобильные роботы или автомобиль, которые оснащены сенсорными устройствами для отображения окружающей обстановки. После обучения нейронная сеть на основе информации от сенсорных устройств должна обеспечивать корректное управление движением. Возможность создания таких систем базируется на обобщающей способности нейронных сетей, которая позволяет интегрировать частные данные для определения закономерностей процесса. В результате этого нейронная сеть способна выдавать правильную реакцию на входных данных, которые не входили в обучающую выборку. Общая модель взаимодействия таких систем с внешней средой изображена на рис. 6.2.2.

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

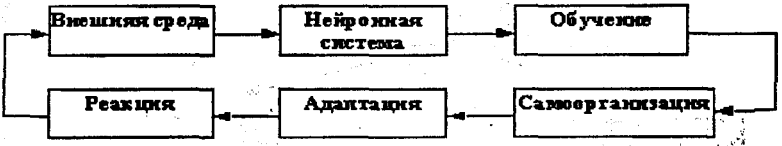


Рисунок 6.2.2 – Взаимодействие нейронной системы с внешней средой

Достаточно задать системе управления координаты конечной точки движения – и томобиль, к примеру, без участия человека будет двигаться к цели. Такая система зработана в университете Карнеги-Меллона в рамках ALVINN проекта (Autonomous and Vehicle In a Neural Networks). В ней предполагается, что автомобиль оборудован идеокамерой, которая отображает дорогу с разметкой. Центральным элементом такой истемы является трехслойная НС с прямыми связями (рис. 6.2.3).

Входной слой содержит 30×32 нейронных элемента, на которые подается преобзованное от видеокамеры изображение пути. Скрытый слой состоит из пяти, а выдной слой из 30-ти нейронных элементов. В качестве функции активации используся сигмоидная функция. Активность выходных нейронов характеризует поворот ля в ту или иную сторону. Так, если максимальной активностью обладает ценальный нейрон, то это означает движение прямо. Когда наибольшую активность еет крайний левый нейрон, то это соответствует повороту налево на определенное исло градусов. Нейронная сеть обучается при управлении автомобилем оператором. ри этом оператор управлял автомобилем при движении со скоростью 9.5 км/ч, молируя различные ситуации. Изображение от видеокамеры использовалось как вход, екущее направление руля – как желаемый выход. С целью упрощения получения учающей выборки используется программное вращение изображения от видеокаеры (рис. 6.2.4), и соответствующим образом меняется реакция нейронной системы.

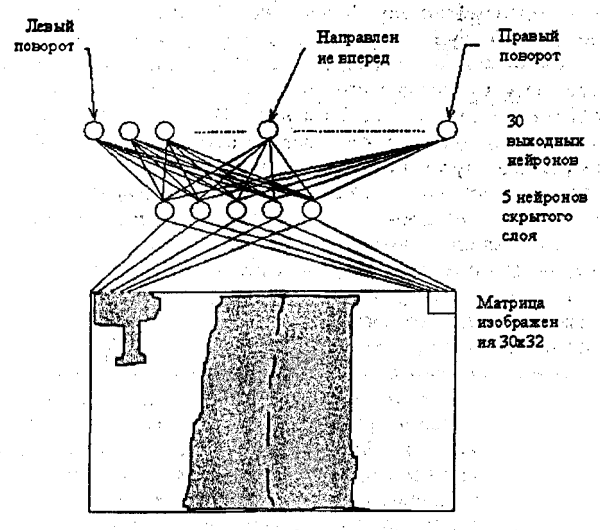


Рисунок 6.2.3 – Отображение изображения дороги на нейронную сеть

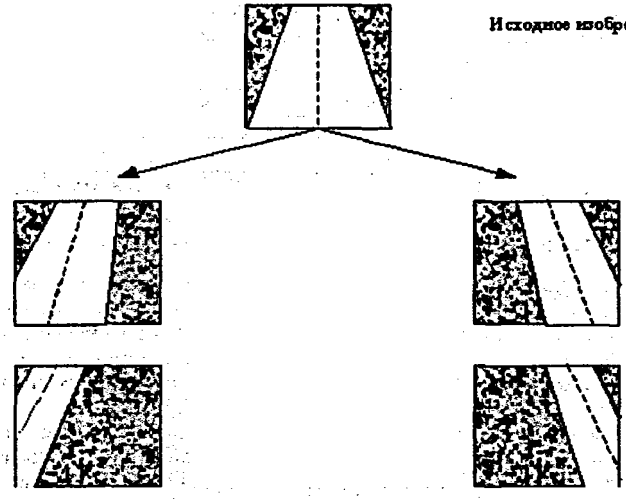


Рисунок 6.2.4 – Вращение исходного изображения от видеокамеры

В результате была создана обучающая выборка, объем которой составил 1200 тренировочных наборов. Обучение нейронной сети проводилось с использованием трех станций «Sun-4». Время обучения методом обратного распространения ошибки составило пять минут. После обучения, как показали эксперименты, нейронная сеть может автономно управлять автомобилем. В настоящее время в рамках этого проекта была достигнута скорость движения автомобиля до 70 миль/ч. При этом автомобиль проехал 90 миль к северу от Питтсбурга, что свидетельствует о большом потенциале нейронных сетей для решения различных задач.

Нейросетевая идентификация. *Нейросетевой подход к идентификации объектов* предполагает разработку НС, алгоритма ее обучения и технологии ее применения. Поиск объектов на основе НС состоит в сканировании изображения окном, размер которого равен размеру обучающего изображения, и определения по отклику НС, является ли изображение внутри окна элементом. Выбор архитектуры НС определяется типом объекта и формой его представления на разных этапах обработки и осуществляется, как правило, путем экспериментальных просчетов и оценок их качества. Учитывая многообразие объектов, используются разнообразные НС. Рассмотрим решение задачи идентификации примере топологических слоев, предлагается для решения задачи идентификации совокупность НС, включающая многослойный персептрон и неокогнитрон, которые апробированы и рекомендуются для использования.

Многослойный персептрон (рис. 6.2.5) состоит из множества слоев нейронных элементов, причем известно, что достаточно трех слоев для создания сколь угодно сложной решающей граничной поверхности в пространстве обучающих образов. При ее построении количество n нейронов первого слоя определяется в соответствии с количеством информативных признаков (яркостные значения (r, g, b), семантические дескрипторы и др.). Количество нейронов промежуточных слоев определяется эмпирически. Выходной слой имеет p нейронов, активность которых определяет принадлежность изображения окна к элементу в виде некоторой функции, значение которой сравнивается с некоторым заданным порогом. Каноническим методом обучения такой сети считается алгоритм обратного распространения ошибки. Следует отметить, что данная НС эффективно используется для решения задач прогнозирования.

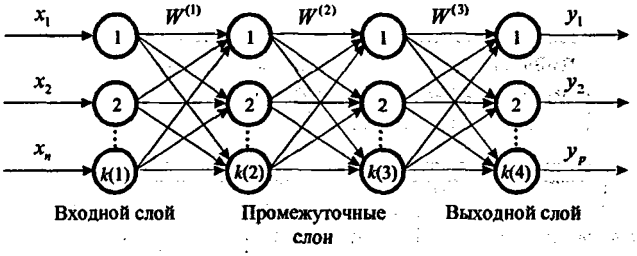


Рисунок 6.2.5 – Архитектура многослойного персептрона

Неокогнитрон (рис. 6.2.6) позволяет выполнить при поиске элементов сравнение изображения в окне сканирования с эталонным образцом с учетом иерархии признаков, формируемой автоматически в процессе обучения. Слой R является входным. S_1-слой предназначен для выделения общих признаков, таких, как линейные границы перепадов яркостей различной ориентации. Все подслои этого слоя состоят из нейронов с одинаковыми размерами подгрупп рецепторного поля 4×4 нейрона. Обучающие образы для этих подслоев изображены на рис. 6.2.7.

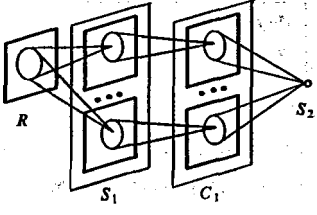


Рисунок 6.2.6 – Используемая архитектура сети

Рисунок 6.2.7 – Обучающие образы S_1-слоя

C_1-слой предназначен для обобщения признаков, выделяемых в S_1-слое, а рецепторные подгруппы его нейронов организованы таким образом, чтобы объединять такие парные признаки, как перепады яркости с темного на светлый или наоборот (такие признаки объединены на рис. 6.2.7 горизонтальной прямоугольной скобкой). В результате получаем четыре подслоя в C_1-слое. Размер подгрупп рецепторного поля выбран такой, чтобы активность нейронов из этого слоя была инвариантна по отношению к малым сдвигам признаков, выделяемых в предыдущем слое, и составляет поле 2×2. S_2-слой предназначен для выделения совокупности признаков, присущих данному конкретному элементу. Он формирует выходное значение сети и состоит из одного нейрона, а размеры подгрупп рецепторного поля этого нейрона совпадают с размерами подслоев C_1-слоя. Высокая точность идентификации достигается вследствие того, что НС *является инвариантной к искажениям формы и яркости изображения* распознаваемого объекта. При этом для достижения необходимой точности идентификации требуется меньшее количество обучающих образцов. Типовой алгоритм идентификации (поиска) объектов на изображениях топологических слоев СБИС (структурная схема представлена на рис. 6.2.8).

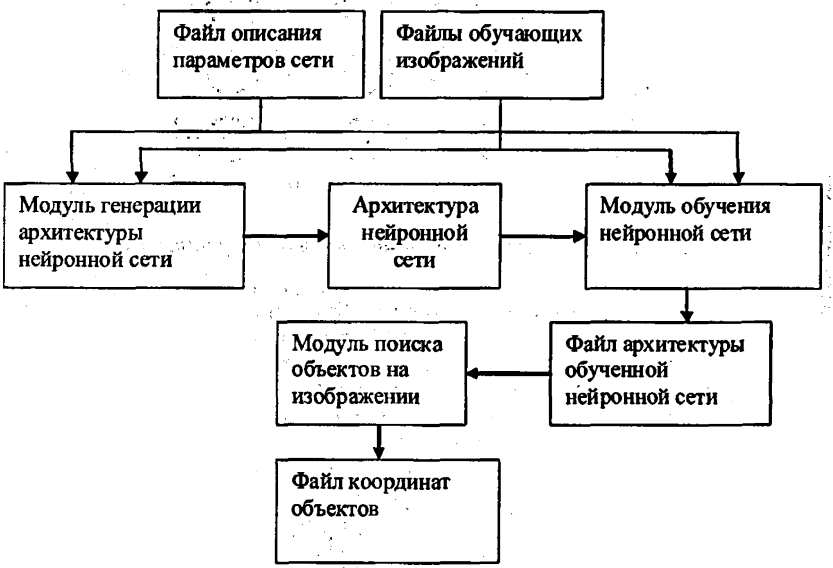


Рисунок 6.2.8 – Схема выделения топологических объектов

Обобщенная структура неокогнитрона для распознавания. Рассмотрим две матрицы

$$W = \begin{pmatrix} w_{1,1} & \cdots & w_{1,Y} \\ \vdots & \ddots & \\ w_{X,1} & & w_{X,Y} \end{pmatrix} \text{ и } A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,Y1} \\ \vdots & \ddots & \\ a_{X1,1} & & a_{X1,Y1} \end{pmatrix}, \qquad (6.2.1)$$

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 $a_{x1,\,y1}$, $a_{x2,\,y2}$ и их смещений $\vec{v}_{x1,y1}$, $\vec{v}_{x2,y2}$ справедливо $|\vec{v}_{x1,y1} - \vec{v}_{x2,y2}| \le \theta$, где $\theta \le \max(|\vec{v}_{x1,y1}|, |\vec{v}_{x2,y2}|)$. Данное свойство указывает на однородность геометрических искажений, т. е. на почти одинаковое смещение пикселей.

Построим матрицу

$$D=\begin{pmatrix} d_{1,1} & \cdots & d_{1,Y} \\ \vdots & \ddots & \\ d_{X,1} & & d_{X,Y} \end{pmatrix},$$

$$d_{x,y}=\begin{cases} 0, & \text{если } \min\limits_{i,j=0,\ldots,2R}(a_{x+i,y+j}) \le w_{x,y} \le \max\limits_{i,j=0,\ldots,2R}(a_{x+i,y+j}); \\ 0, & \text{если } |w_{x,y}-a_{x+R,y+R}| < B; \\ 1 & \text{в остальных случаях.} \end{cases} \qquad (6.2.2)$$

Теперь функция нечеткого различия матриц A и W определяется как

$$fd(A,W)=\frac{100}{X\cdot Y}\cdot\sum_{x,y} d_{x,y} \qquad (6.2.3)$$

Структура нейрона, реализующего описанное выше правило активации, приведена на рис. 6.2.9, где значения $[\min_k, \max_k]$ определяются так же, как в формуле 6.2.2): $\min_k = \min_i(a_{k+i}), i = 0, \ldots, 2R$; $\max_k = \max_i(a_{k+i}), i = 0, \ldots, 2R$, $k = 1, \ldots, N$, 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 $\{W^t \mid t = 0, \ldots, T\}$, где T – общее число итераций.

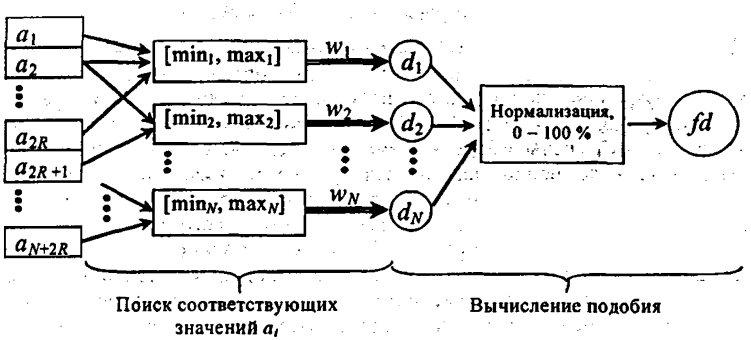


Рисунок 6.2.9 – Общая структура нечеткого нейрона

Определим функцию нечеткого динамического среднего:

$$w_{x,y}(t+1)=\begin{cases} w_{x,y}(t), & \text{если } \min\limits_{i,j=0\ldots2R}(a_{x+i,y+j}(t)) \le w_{x,y}(t) \le \max\limits_{i,j=0\ldots2R}(a_{x+i,y+j}(t)), \\ w_{x,y}(t), & \text{если } |w_{x,y}(t)-a_{x+R,y+R}(t)| < B, \\ \dfrac{w_{x,y}(t)\cdot t + a'_{x,y}}{t+1}, & \text{в остальных случаях} \end{cases} \qquad (6.2.4)$$

где t – порядковый номер итерации усреднения;

$a'_{x,y} \in \{ a_{x+i,y+j}(t) \mid i,j = 0, \ldots, 2R \}$, 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 неокогнитрона с тем лишь отличием, что из-за упрощения структуры нейрона в ней отсутствуют обобщающие *С*-слои (рис. 6.2.10).

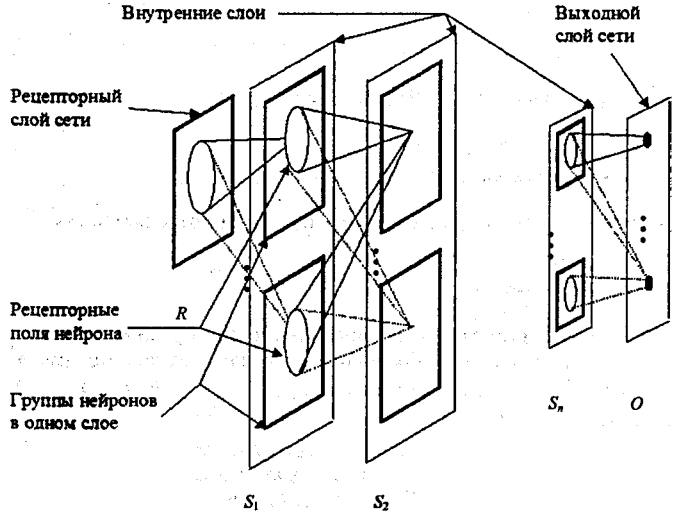


Рисунок 6.2.10 – Структура НС

Рецепторный слой *R* является входным слоем сети, на который подается обрабатываемый образ. Он преобразует яркости пикселей изображения в значения активностей нейронов. Внутренние слои $S_1, \ldots, S_{n-1}$ производят последовательную обработку поступающей информации, а выходной слой сети *О* генерирует ответ, к какому классу относится входное изображение в виде максимальной активности одного из своих нейронов.

Нейроны, выделяющие одинаковый признак на изображении, объединяются в одну группу. Исходными данными для активации каждого нейрона из отдельной группы является локальная часть данных из предыдущего слоя с соответствующим этому нейрону смещением. На рис. 6.2.11 приведен пример структуры связей нейронов из одной группы в случае одномерных входных данных. Нейроны являются классификаторами признаков определенного уровня иерархии, входные данные для которых поступают с предыдущего слоя и объединяют совокупность признаков предыдущего уровня иерархии. Активность нейрона определяет степень принадлежности рассматриваемой совокупности признаков следующему уровню иерархии.

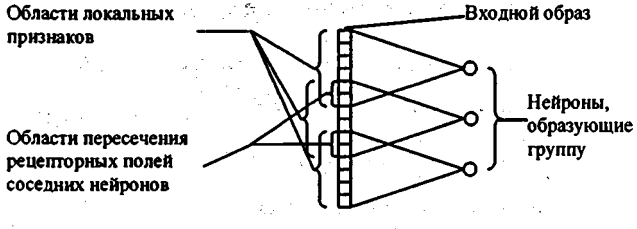


Рисунок 6.2.11 – Схема объединения нейронов в группу

Функционирование нейрона определяется соотношениями (6.2.2)–(6.2.4), в которых переменные R и B являются параметрами. Значение B зададим соотношением

$$B = \frac{1}{Y \cdot X} \cdot \sum_{x,y} \left[\frac{1}{L} \cdot \sum_{l}^{L} \left| a_{x,y} - \frac{1}{L} \cdot \sum_{l}^{L} a_{x,y} \right| \right], \tag{6.2.5}$$

где L – число обучающих изображений размера $X \times Y$. Обозначим через F среднюю частоту смены уровня яркости изображения, для которой соседние точки изображения считаются принадлежащими одному уровню при условии, что абсолютная разница их яркости меньше B. Теперь R зададим как

$$R = [0{,}3 \times 1 / F], \tag{6.2.6}$$

где $[\times]$ – 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

$$c = 1 + \frac{n \sum_{x,y} \left(w_{x,y} a'_{x,y} \right) - \sum_{x,y} w_{x,y} \sum_{x,y} a'_{x,y}}{\sqrt{\left(n \sum_{x,y} \left(w_{x,y} a'_{x,y} \right) - \left(\sum_{x,y} w_{x,y} \right)^2 \right) \left(n \sum_{x,y} \left(w_{x,y} a'_{x,y} \right) - \left(\sum_{x,y} a'_{x,y} \right)^2 \right)}}, \tag{6.2.7}$$

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

$$a'_{x,y} = \begin{cases} 0, \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}) \right] \cap \\ \cap \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}) \right] \neq \varnothing; \\ a_{x,y}, \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (a_{x+rx,y+ry}) \right] \cap \\ \cap \left[\min\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}); \max\limits_{rx=-R,\ldots,R;\ ry=-R,\ldots,R} (w_{x+rx,y+ry}) \right] = \varnothing. \end{cases}$$

В результате применения функции (6.2.7) получаем значения в интервале $c \in [0; 2]$, где значения $c \in [0; 1)$ говорят об обратной корреляции, значения $c = 1$ – об отсутствии корреляции, а значения $c \in (1; 2]$ – о прямой корреляции. Таким образом, данный подход позволяет отказаться от подбора диаметра яркостных искажений.

Определим значение радиуса геометрических искажений через значение частоты перепадов, вычисляемое по формуле

$$F = \frac{F_h + F_v}{2}, \tag{6.2.8}$$

где $F_h = \frac{1}{X}\left(1+\frac{1}{Y}\sum_{x=1,y=1}^{X-1,Y}\delta(g^h_{x,y}, g^h_{x+1,y})\right)$, $F_v = \frac{1}{Y}\left(1+\frac{1}{X}\sum_{x=1,y=1}^{X,Y-1}\delta(g^v_{x,y}, g^v_{x,y+1})\right)$

$g^h_{x,y}$, $g^v_{x,y}$ – значения вертикального и горизонтального градиентов в точке (x, y)

$$\delta(a,b)=\begin{cases}1, sign(a)\neq sign(b);\\ 0 \text{ в остальных случаях}\end{cases}$$

Значение R зададим, как и ранее, формулой (6.2.6).

Метод **генерации обучающего множества.** В рассматриваемой постановке задачи распознавания предполагается, что существует набор изображений объектов нескольких типов, причем объекты одного типа имеют разновидности, отличающиеся друг от друга как по яркостным, так и геометрическим характеристикам. Имея такой набор образцов, необходимо разработать метод автоматической генерации обучающих множеств для НС. Для каждого слоя сети можно определить размеры проекции рецепторного поля нейрона на входной слой НС (см. рис. 6.2.12 для случая одномерных входных данных).

Таким образом, для одного обучающего изображения получим K обучающих изображений для некоторого i-го слоя сети: $K = N_{in} - N_i + 1$, где N_{in} – размер входных данных; N_i – размер проекции рецепторного поля i-го слоя сети. В обучающее множество включаем изображения повышенной информативности: данные для i-го слоя сети являются информативными, если выполняется условие: $F_{in} / F_k \le 0{,}8$ или $F_k / F_{in} \le 1{,}25$.

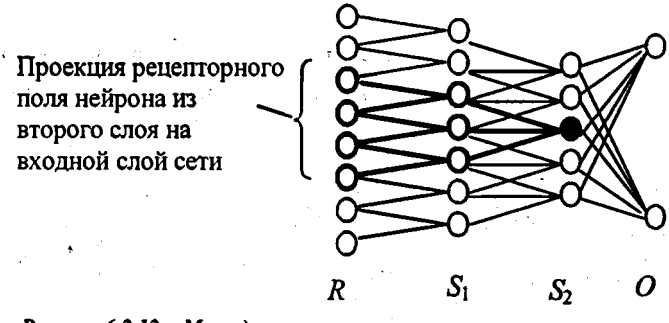


Рисунок 6.2.12 – Метод построения проекции рецепторного поля

Получив обучающее множество для каждого слоя, легко заметить, что *задача обучения слоя* сводится к задаче кластеризации входных данных слоя, получаемых из обучающего множества. В связи с тем, что входящие в одну группу нейроны имеют одинаковые матрицы весовых коэффициентов, всю группу можно представить в виде одного нейрона, а слой – в виде однослойной НС, в которой каждый нейрон соответствует одной из групп. Для обучения такой сети будем использовать алгоритм кластеризации (κ-средних, пикового группирования или типа FOREL в зависимости от свойств обучающего множества), где каждый кластер будет соответствовать одному нейрону. После обучения будем использовать такую НС как прототип для построения искомого слоя в многослойной сети, где количество нейронов в прототипе будет соответствовать количеству групп, а весовые коэффициенты нейронов из прототипа – весовым коэффициентам у нейронов из соответствующих групп.

Генерация обучающего множества для первого слоя НС на примере одного обучающего изображения показана на рис. 6.2.13.

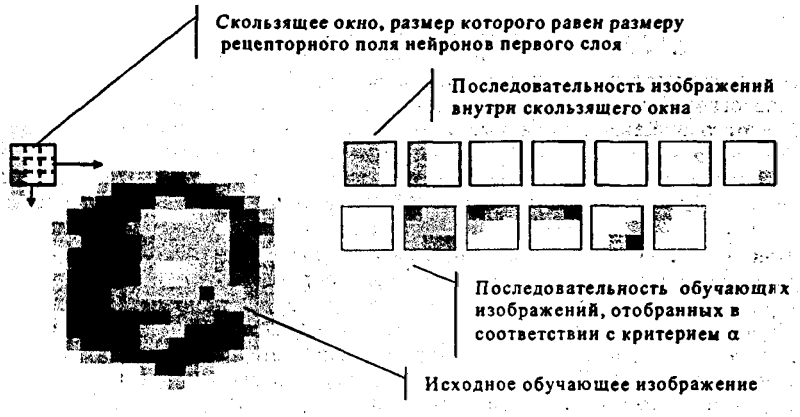


Рисунок 6.2.13 – Генерация обучающего множества слоя

Эксперименты показали, что использование функции (6.8) 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_{in} и F_k.

6.3. Системы технического зрения

Методы оптического контроля. Важной проблемой при производстве СБИС является контроль качества на различных технологических этапах. Для этого могут применяться как контактные (электрические), так и бесконтактные методы (оптические, рентгеновские, томографические, ультразвуковые и инфракрасные).

Задача оптического контроля заключается в проверке качества металлизации соединений и контактных окон, элементов топологии поликремниевых и диффузионных областей в ходе изготовления СБИС или подготовки данных и определении дефектов.

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

– более низкую стоимость обнаружения дефекта, обусловленную ранним детектированием, что дает возможность избегать дефектов на поздних шагах, когда их стоимость выше;

– возможность непрерывного контроля процесса производства;

– сокращение времени контроля.

Оптический контроль базируется на методах морфологических описаний, сравнения с эталоном и контроля проектных норм (конструкторско-технологических ограничений).

Метод морфологических описаний в комбинации с некоторыми базовыми критериями правил проектирования позволяет определять ошибки проектирования. Суть метода заключается в том, что изображение топологии анализируется и представляется как список признаков или образов. Каждый образ классифицируется, и определяется его позиция. Из эталонного изображения топологии СБИС, полученного САПР, формируется список эталонных признаков. Далее 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, 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 требуется постоянная доработка алгоритма контроля. Применение процесса верификации правил проектирования непосредственно к изображению образов является затратным по времени и требует больших вычислительных мощностей. Кроме того, проверка правил проектирования приводит к большому числу ложных дефектов. Поэтому важно комбинирование различных методов контроля для достижения лучшего результата (рис. 6.3.1).

Рисунок 6.3.1 – Схема комбинирования методов оптического контроля

Однако ни один из методов не даст ответа о наличии дефекта, когда на одном изображении одни контактные площадки соединены с другими, а другие являются изолированными. В этом случае оптический контроль дополняется электрическим тестированием. Такой вид контроля получил название автоматического оптического тестирования, при котором анализируется связность объектов топологии.

Общая функциональная схема системы оптического контроля. В настоящее время успехи в машинном зрении позволяют обеспечить автоматизированный процесс производства как электронных компонент, так и СБИС в целом. Типовая система технического зрения (СТЗ) контроля выполняет следующие три основные функции (рис. 6.3.2):

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

На основе полученной информации контроллер связи выбирает управляющие сигналы, которые приводят в действие исполнительные механизмы, осуществляющие целенаправленное воздействие на объект. При необходимости обработанная информация об объекте высвечивается на устройстве визуального контроля, записывается на носители информации и выводится на печатающее устройство или иные устройства вывода информации.

В функции блока управления входит управление параметрами обработки, а также синхронизация процессов, выполняющихся в системе.

Следует отметить, что функции отдельных блоков системы могут существенно варьироваться. Это зависит от характера изображений, априорной информации, удачности выбора первичных параметров и класса решающих правил. Кроме того, возможно выполнение одними и теми же блоками нескольких функций.

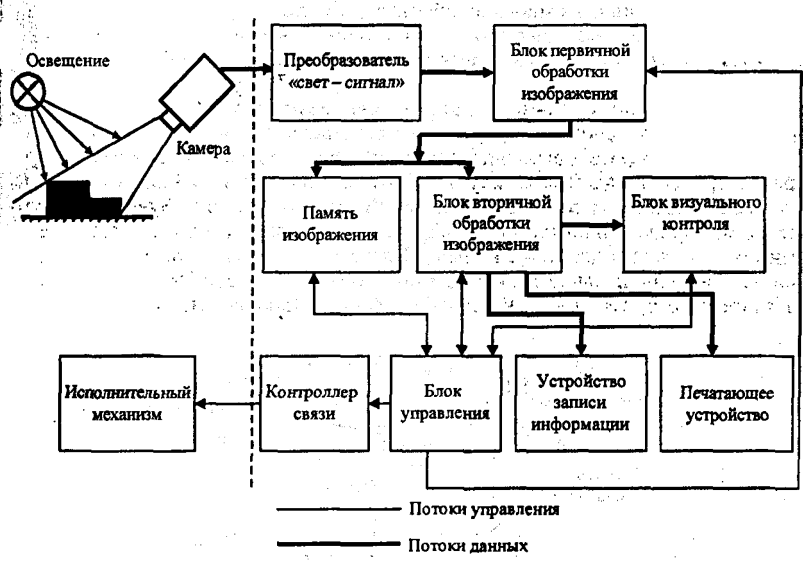


Рисунок 6.3.2 – Функциональная схема СТЗ контроля

Для увеличения скорости обработки используется специальное аппаратное обеспечение. Для эффективной реализации цифровой обработки сигналов хорошо подходят систолические структуры благодаря параллелизму и регулярности потока данных. Для них характерно наличие однотипных модульных локально-связанных процессорных элементов и совмещение в своей архитектуре двух принципов параллельной обработ-

ки: конвейерного и матричного. Среди матричных структур, реализующих логические преобразователи информации, широкое применение находят программируемые логические матрицы, программируемые логические интегральные схемы и т.п.

С аппаратным синтезом таких вычислительных систем связан ряд проблем, в частности, проблемы межпроцессорных соединений, синхронизации и их модульной организации. Критериями синтеза выступают минимизация потребляемой мощности, задержек и габаритов. В этой связи является актуальным разработка методов и алгоритмы декомпозиции систем булевых функций и покрытия логических схем библиотечными элементами полузаказных 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 рис. 6.3.3 с использованием системы обозначений, предложенной Г. Бучем (она соответствует схеме, приведенной на рис. 6.3.2). Здесь показаны два типа связей между объектами (экземплярами классов системы). Классы по своему назначению соответствуют блокам функциональной схемы (для построения диаграмм было использовано средство Rational Rose / C++ Demo Version 4.0.3 фирмы Rational Software Corp.)

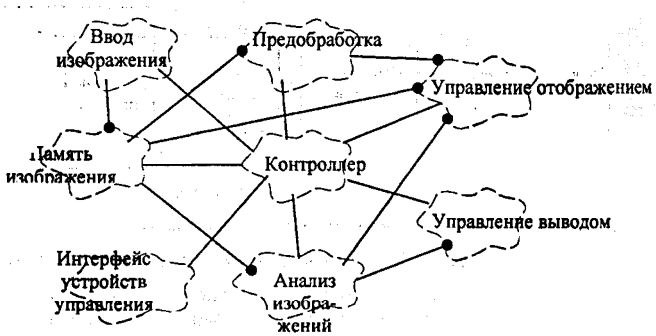


Рисунок 6.3.3 – Архитектура ПО СТЗ

Пример клиент-серверных отношений в модели: объект класса «управление ото-жением» (ответственный за отображение информации в процессе обработки) ис-льзует данные, предоставляемые объектами классов «предобработка» (соответст-щего блоку предварительной обработки), «память изображения» и «анализ изо-жения».

Базовые методы обработки изображений представлены на рис. 6.3.4. Конкретные ы зависят от технологий, реализованных в СТЗ.

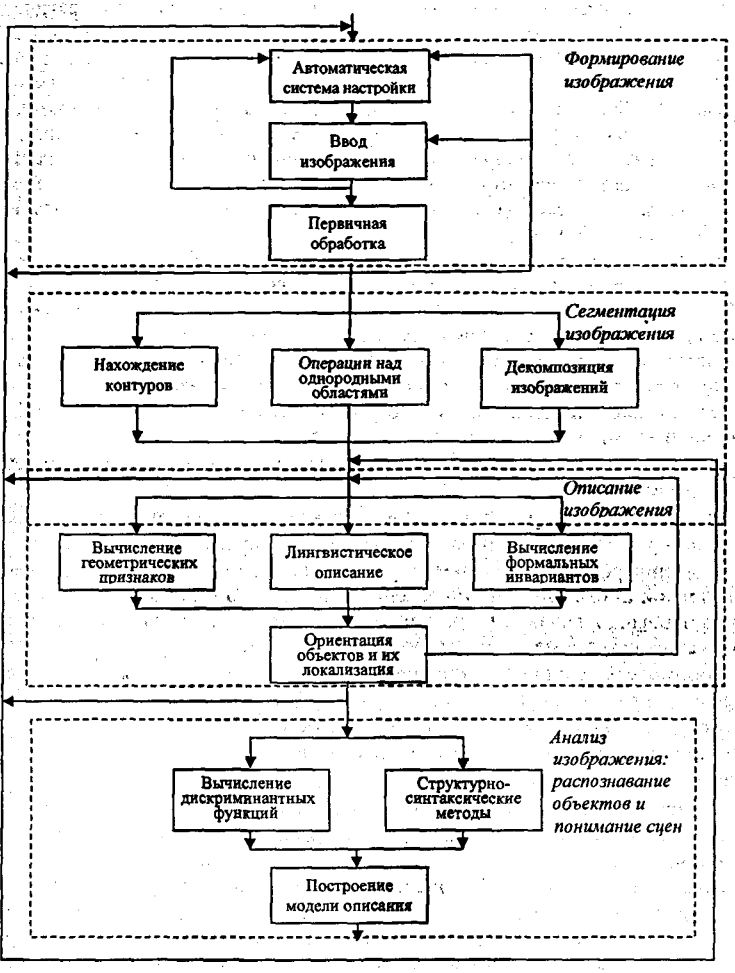


Рисунок 6.3.4 – Структурная схема обработки изображений

Принимая во внимание сложность СТЗ обработки изображений топологии, целе-бразно использовать комплексный подход для построения ПО СТЗ: приняв за ос-у объектно-ориентированное проектирование, использовать метод потоков дан-х для организации параллельной обработки в рамках многоагентного подхода.

6.4. Мультиагентные системы

На заре развития теории искусственного интеллекта решение задачи сводилось к созданию одной системы, способной её решить. В дальнейшем по мере возрастания потребностей из различных областей (наука, техника, экономика, управление государством и крупными фирмами в мировом масштабе) выявился специфический класс процессов управления, когда элементы системы, владея частичной информацией о глобальной проблеме должны эффективно принимать свои решения с позиций интересов функционирования общего большого объекта. Выяснилась необходимость организации работы многих агентов (программ, людей), обладающих правом на самостоятельные действия при сохранении ответственности за них. Решение такого рода задач вызвало необходимость создания теории мультиагентных систем (раздела теории распределённого искусственного интеллекта, учитывающего возможности самообучения отдельных агентов при принятии решений в рамках своей компетенции). Сложность этих задач возросла и из-за необходимости использования знаний из теорий параллельных вычислений, сетевых компьютерных технологий, защиты информации, мобильной связи и ряда других.

Это привело к развитию новых подходов в создании интеллектуальных систем, в частности, мультиагентных систем прикладного назначения (поддержка принятия решений в условиях недостатка информации из-за её недоступности и/или непредсказуемости, особенности принятия коллективных и личных решений).

Под агентом часто понимается программа (робот или человек), способная действовать в интересах достижения целей, поставленных пользователем.

Индустрия многоагентных систем пока находится в стадии становления. Она постепенно начинает проникновение в следующие области:

– подготовка вариантов информации для отдельных ЛПР или коллективов, принимающих решения;

– поддержка конкурентоспособности производств и завоевание рынков сбыта;

– поиск сервисов и дефицитных продуктов для сложных производств;

– управление роботами и людьми в различных сложных технических объектах.

При полной автоматизации управления сложными объектами интеллектуальными агентами обычно являются программы и роботы. Для мультиагентных систем сформировались некоторые общие требования к агентам:

– автономность (функционирование без прямого вмешательства других агентов);

– взаимодействие с другими агентами на общем коммуникационном языке;

– восприятие внешней среды через различные типы интерфейсов;

– реакция на происходящие изменения в среде;

– проявление инициативы;

– наличие собственной картины окружающего мира;

– способность к интеллектуальному поведению на основе обучения и логических выводов;

– каждый агент способен обмениваться своими знаниями с другими агентами.

Кроме того, при отказе одного или даже нескольких агентов система должна продолжить функционирование, независимые задачи могут выполняться различными агентами. Цели выдвигаются пользователем системы. Замечено, что при требовании согласования решений между несколькими агентами временные затраты на функционирование системы растут примерно по экспоненциальному закону.

Проиллюстрируем схему организация вычислений на базе многоагентной архитектуры.

Общая архитектура вычислительной платформы показана на рис. 6.4.1. Интерфейс пользователя (блок *A*) представляет собой средство визуального проектиро-

вания 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 *B*), задать параметры вызова гранул для различных типов обрабатываемых данных.

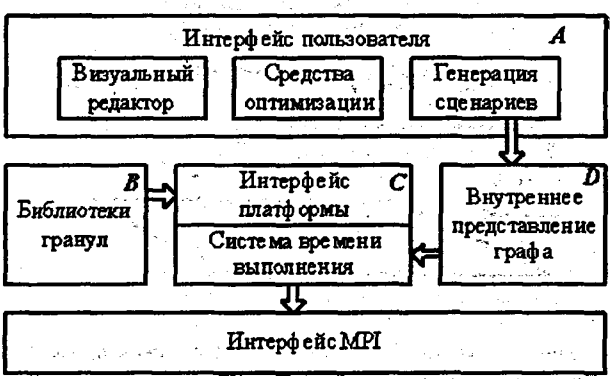


Рисунок 6.4.1 – Архитектура платформы организации параллельных вычислений

Средства оптимизации состоят из алгоритма оптимизации расписания на основе виртуальных ассоциативных сетей и имитационной модели. Имитационная модель используется для оценки вариантов расписания в процессе оптимизации, а также для визуализации расписания. Расписание представляется в виде диаграмм Ганта. Алгоритм оптимизации реализует технологию эволюционной оптимизации и относится к гибридным генетическим алгоритмам, которые имеют преимущества перед эвристическими алгоритмами назначения задач на вычислительные узлы и алгоритмами статической оптимизации вычислений, принимая во внимание специфику обрабатываемого потока (неоднородность данных и задержек при 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 ВС. Файл XML при выполнении программы транслируется во внутреннее представление (блок *D*), позволяющее каждому элементу системы времени выполнения (блок *C*) получать необходимую информацию во время работы.

Система времени выполнения является многоагентной системой, использующей средства MPI для осуществления координации взаимодействия элементов. Она содержит два типа программных агентов: координатор и исполнитель. Количество агентов-исполнителей при выполнении параллельного приложения равно количеству физических процессоров ВС плюс один агент-координатор. Взаимодействие между агентами осуществляется с помощью передачи сообщений, реализующих простой протокол взаимодействия (рис. 6.4.2).

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.

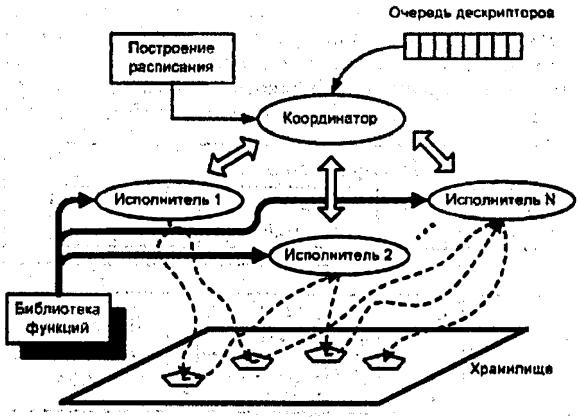


Рисунок 6.4.2 – Архитектура системы времени выполнения

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 (рис. 6.4.3) входят клиент (на основе платформы Windows) и параллельный сервер (на основе платформы Linux), взаимодействие между которыми осуществляется через файловую систему. Клиент выполняет задачи, обеспечивающие интерфейс системы с пользователем. Он содержит хранилище данных, ПО обработки изображений слоев СБИС и средства удаленного вызова процедур параллельного сервера.

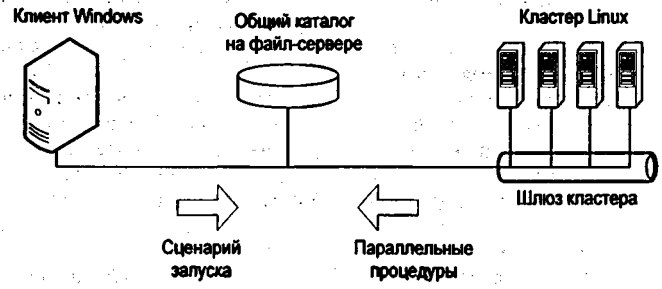


Рисунок 6.4.3 – Архитектура вычислительного комплекса

Клиент имеет доступ к общему каталогу на сервере с помощью средств сервера Samba. Сервер настраивает определенный каталог для общего доступа, и клиент может монтировать его при загрузке Windows. В данный каталог помещаются файлы

обработки (например, кадры), а также сценарии обработки, представленные в ви- текстового файла в синтаксисе Linux shell и содержащие команды запуска проце- параллельного сервера с требуемыми параметрами командной строки, указанны- в описании процедур обработки. Выполнение сценария со стороны клиента осу- ствляется посредством протокола SSH (Secure SHell). Реализация этого протокола ествует как для платформ Linux, так и для Windows. Со стороны клиента Windows ользуется программа-клиент Putty, позволяющая настроить доступ к удаленной шине и выполнять на ней команды и сценарии.

Сервер представляет собой параллельный вычислительный кластер на базе ОС их, который использует библиотеку поддержки параллельных вычислений MPICH и страивается на использование SSH без пароля в качестве команды удаленного запуска.

6.5. Транзакции в сетях

Информационные компьютерные технологии в сетях существенно отличаются штабами, формой организации и средствами управления БД. Наряду с традицион- ми БД, используемыми в оперативном режиме для решения задачи, создаются до- нительные хранилища и витрины данных. Второй особенностью является исполь- ание в локальных сетях информационных технологий и сервиса Интернет. Появ- тся возможность реализации дополнительных функций типа электронной коммер- и, совместной обработки документов, электронного безбумажного документообо- та, телеконференций и т.д.

В третьих появляются специфические алгоритмы поиска и обработки данных, пример, обращение к платным хранилищам и услугам.

Специфичен транзакционный подход: на период решения задачи запрещается менение некоторых функций для других проблем и в случае неудачной попытки шить задачу отката системы в исходное состояние.

Основу любых дистанционных операций составляет транзакция. Этот термин пользуют как в деловом мире электронной торговли, банковских платежах, обра- ениях с запросом к банкам данных. Однако пользователи термина могут вкладывать его несколько отличный смысл, исходя из характера выполняемой операции. На- мер, транзакция это: сделка, единица коммуникативного процесса; единичное ствие, имеющее два состояния – выполнено (1) и не выполнено (0); минимальная тически осмысленная операция; группа взаимосвязанных операций, которые обла- от свойствами атомарности, непротиворечивости, изолированности.

Нами предлагается использовать в дистанционной работе и дистанционных ммерческих операциях термин дистанционная транзакция, включающий обеспече- е не только свершение операций (сделки), но и соблюдение ряда требований к ее более ответственным процедурам, касающимся обеспечения конфиденциальности елки, обеспечения юридической ответственности за передаваемую информацию, озможность до завершения процедуры всей транзакции начать другую транзак- ю, а при срыве выполнения хотя бы одного этапа всю систему затронутых элемен- вернуть в исходное состояние. Похожая схема сейчас реализуется при операциях азами данных в информационных системах. Примерно такая же схема рекоменду- ся для сделок между хозяйствующими субъектами с требованиями к неделимости, гласованности, надежности и изолированности. Примеры использования транзак- й рассмотрим в главе 7 в различных коммерческих и банковских операциях, а так- е при некоторых видах дистанционной работы.

Литература

[3, 4, 5, 7, 8, 9, 11, 12, 13, 14, 15, 17, 24, 25, 27, 28, 29, 31, 33, 34, 37, 41, 42]

Контрольные вопросы

1. Какое принципиальное отличие имеет интеллектуальная информационная технология от традиционно используемой в промышленности?
2. За счет чего достигается эффективность работы современных шахматных программ?
3. Что является основой для создания дистанционных рабочих мест и средств обучения?
4. В чем состоит отличие документа с электронной подписью от обычного?
5. Какова роль транзакций в дистанционных операциях?
6. Роль отката системы в транзакциях.
7. Причины, побудившие развитие теории мультиагентных систем.
8. Назовите требования к агентам мультиагентных систем.

Глава 7. ПЕРСПЕКТИВЫ РАЗВИТИЯ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

7.1. Влияние нанотехнологий и мобильной связи на развитие информационных технологий

Нанотехнология – область науки и техники, занимающаяся разработкой теоретических и практических методов создания и применения продуктов с заданной атомной структурой путём контролируемого манипулирования отдельными атомами и молекулами. Размеры объектов в нанотехнологиях измеряются в нанометрах (1нм – миллиардная часть метра) и их диапазон колеблется от 1 до 100 нанометров. Принципиальная особенность нанотехнологической революции – переход от движения в сторону миниатюризации в обратном направлении: складывание больших объектов из молекул и атомов. Развитие нанотехнологий идёт на стыке ряда научных дисциплин и носит надотраслевой характер. Ожидается одно из ярких их проявлений в робото- и компьютерной технике благодаря дальнейшей миниатюризации этих устройств и созданию нанотранзисторов.

Нанокомпьютер – вычислительное устройство на основе электронных (механических, биохимических, биологических, квантовых и др.) нанотехнологий с размерами логических элементов в несколько нанометров (1-100). Их основой являются нанотранзисторы с реализацией логических переключательных функций на базовых элементах разной природы (биологических, электронных и других).

Большинство из них из-за наноэффектов обнаруживают ряд нетрадиционных свойств: энергонезависимое поддержание своего состояния, возможность для одного и того же материала быть сверхпроводником, полупроводником и изолятором и т.д. Биотранзистор настолько мал, что его введение в живую клетку не нарушает режима её функционирования. Это открывает путь к сенсорным измерениям в живых клетках организма и использованию нанороботов в лечебных целях. Биокомпьютер (ДНК-компьютер) функционирует как живой организм и может непосредственно вступать во взаимодействие с внешней живой средой. В области вычислительной техники и нанороботов удалось получить опытные образцы процессоров размером 22 нм. Э.Дрекслер (США) из лаборатории искусственного интеллекта Массачусетского технологического института выдвинул концепцию универсальных молекулярных роботов, работающих по заданной программе по сборке объектов из молекул. Ожидается, что нанороботы будут эффективны при лечении раковых заболеваний и создании новых моделей роботов.

Наиболее продвинутой прикладной базой для построения нанотранзисторов являются углеродные нанотрубки (УНТ). Они характеризуются проводимостью около 1000 раз выше меди и имеют диаметр от 1,2 до 1,5 нм с размерами в длину до 360 нм. В нанодиапазоне электроны демонстрируют одновременно свойства волн и частиц. Заряд можно измерять только порциями (квантами). Половину электрона прибавить нельзя. При приложении небольшого электрического поля вдоль оси нанотрубки с её концов происходит интенсивная эмиссия электронов (полевая эмиссия). Это позволяет создавать полевые транзисторы в качестве переключательных элементов. Одновременно появляется возможность на несколько порядков увеличить ёмкость и быстродействие памяти .

Очень большое влияние эти достижения оказали на развитие цифровой мобильной связи. Мобильные средства связи сейчас позволяют обеспечить надёжную беспроводную передачу информации в больших объёмах благодаря встроенных в них миниатюрных модемов и компьютеров с возможностями для стационарных ЭВМ. Цифровая связь одновременно в состоянии обеспечить цифровую подпись информации и её секретность.

К мобильным средствам обычно относят устройства, с помощью которых можно поддерживать связь с возможными абонентами, находясь в движении (с отрывом от ра-

бочего стола), которые также могут перемещаться. К таким устройствам обычно относят различные виды мобильных телефонов, пейджеры, радиостанции в автомобиле и т.п.

Мобильная связь порождает дополнительные возможности в управлении предприятием сферы услуг и при выполнении и учёте заказов.

Ремонтные предприятия бытовой техники, таксопарки, мастера, ведущие несколько строительных объектов или один большой объект, могут управлять своими работниками непосредственно на рабочих местах. Это даёт значительный экономический эффект, так как на перемещение работника с одного места выполнения работы на другое требуется меньше затрат и оказываемая услуга или работа в целом выполняется быстрее. Ярким примером являются таксопарки, службы эвакуации сломавшихся машин, предприятия по ремонту бытовой техники на дому и установки различных измерительных приборов в квартирах и т.п. Наличие мобильной связи с менеджерами различного ранга позволяет оперативно принимать решения диспетчером или другим работником, когда на производстве возникает нештатная ситуация. Хорошо зарекомендовала себя мобильная связь в приёме и обработке различных заказов.

Особые преимущества даёт цифровая мобильная связь, так как в этом случае может обеспечиться передача информации непосредственно в компьютерную сеть, а наиболее ответственная информация может подписываться электронной цифровой подписью, например, распоряжения банку или важные решения, влекущие юридическую ответственность.

Сейчас сети мобильной связи переживают очень бурное развитие благодаря расширению международной торговли и ряду специфических коммерческих услуг по передаче информации. Сетевые услуги мобильной связи постепенно преобразуются в «трубопроводы знаний» благодаря их сопряжению с бытовыми приборами, регистрационными и измерительными устройствами со встроенными микропроцессорами и приборами радиосвязи. Благодаря выходу на спутниковую связь появилась возможность оказывать услуги мультимедийной связи и позиционирования подвижных объектов.

В экономике это уже приносит большие экономические эффекты в управлении грузоперевозками и перемещением движущихся объектов по заданным маршрутам. Решение вообще любых логистических задач стало давать значительную прибыль (отслеживание за движением грузов, предсказание времени пересечения границ, контроль расхода горючего и т. д.).

Абоненты цифровых телефонных сетей в огромных масштабах ощутили возможности выполнения платежных поручений, связанных с различными типами оплаты бытовых и информационных услуг. Возросла эффективность выполнения сетевых коммерческих операций по приобретению бытовой техники и вещей. Последние годы наметился переход к увеличению услуг, связанных со скоростной передачей данных и выполнением аутсортинга авторитетными специалистами из других организаций (перекладывание рисков фирмы на более подготовленных специалистов).

Наблюдается постепенное расширение ИНТЕРНЕТ-экономики и увеличение масштабов производства и рынков сбыта. Это объясняется ещё и тем, что стоимость использования традиционных (бумажных) технологий при хранении и применении данных стала дороже по сравнению с компьютерными технологиями.

Идет значительный прогресс и в предоставлении и оплате услуг со стороны мобильных операторов. В частности, предприятие, использующее широко в своей деятельности мобильную связь начали ощущать расширение льгот и услуг от мобильных операторов, связанных с введением корпоративных льготных тарифных планов и учету расходов фирм на мобильную связь (льготные звонки между сотрудниками фирмы, безлимитные тарифы и др.).

С точки зрения перспектив применения мобильной связи следует обратить внимание на более широкое ее использование в сочетании с сетью ИНТЕРНЕТ и увеличение количества операций, связанных с передачей данных, развитию специфической рекламы и организации производств и продаж.

7.2. Интеллектуализация принятия решений в информационных технологиях

Интеллектуализация современных информационных компьютерных технологий становится главным элементом в повышении творческой роли человека в цепочке принятия решений, чтобы оттеснить его действия на уровни, которые трудно поддаются автоматизации (неопределённость ситуации, недостаток информации и т.п.). Поэтому принятие решений с повышением уровня их сложности требует участия ЛПР и несущего за них ответственность. Пока о полной интеллектуализации принятия решений на машинном уровне задача не ставится. Речь идёт о создании гибкой системы поддержки принятия решений, которая на формализуемых этапах оставляет их выбор за компьютером, а в сложных ситуациях для человека имеющаяся информация приводится в удобный вид для принятия окончательного решения или выбора пути продолжения процесса. Смысл использования системы поддержки принятия решений заключается в облегчении обработки больших объёмов разнородной информации, включающей и такую, где необходимы советы специалистов различных профессий.

Особенно сложными стали управленческие решения при использовании управляющих и технических систем в едином комплексе, содержащих людей и программно-управляемое оборудование (станки с ЧПУ, роботы и т. п.). Сложность повышается ещё и от того, что принятое решение надо преобразовать в набор последовательностей сигналов для оборудования. Если же имеется ещё и ряд критериев, среди которых есть противоречащие друг другу, то проблема дополнительно осложняется в выборе компромиссного варианта, который частично учитывает некоторые критерии, например, выбор расписания преподавателя с минимумом ожидания, когда освободится специализированный кабинет, когда их недостаточно. При желании сохранить высокий уровень автоматизации принятия эффективных решений прибегают к использованию заранее выработанной системы приоритетов в конкретных ситуациях.

В современных технологиях процессов принятия решений считается важным использование сетецентрических (GGG) технологий в системах мониторинга и анализа с целью эффективного сбора нужной информации из разнородных средств (операционных систем, компьютеров, датчиков и т. п.) в реальном масштабе времени. В частности, такие системы оправдали себя при управлении военными операциями.

Близкие к ним задачи решаются при использовании громадных объёмов данных на основе витрин, в которых на мировом уровне содержится краткое описание поисковых объектов, чтобы по адресу выйти на их нужную часть.

Хранилища и витрины данных в системах поддержки принятия решений также важны при необходимости быстрого подбора целевой информации из хранилищ, рассредоточенных по всему миру для конкретного предприятия.

Концепция витрины данных – перечень множества тематических баз данных, содержащих информацию, относящуюся к различным аспектам деятельности организаций.

Такой подход полезен для выбора данных, которые реально нужны за счёт их целенаправленного подбора из очень больших баз данных (по нескольким странам и т.п.).

Процесс принятия решений. Решение – это выбор альтернативы в различных сферах деятельности.

Выделяют особо организационные решения, смысл которых обеспечение движения к цели, т.е. создание условий для выполнения поставленной задачи. Число возможных альтернатив для рассмотрения, исходя из ресурсных возможностей (финансовых, технических, кадровых и т.п.). Обычно ограничено и часто удаётся назвать шаги, которые надо выполнить и их порядок для реализации поставленной задачи. Такого типа решения можно запрограммировать и поручать их выработку компьютеру. Принятию таких решений способствуют часто повторяющиеся ситуации.

Обычно выделяют типичные группы решений по видам, выполняемых функций планирования хода различных процессов, контроль.

Однако есть и ряд таких решений, некоторые этапы которых не поддаются программированию или допускают частичное программирование из-за наличия неизвестных и случайных факторов, которые трудно предвидеть. К таким объектам обычно относятся стратегические решения. Наиболее трудным являются те из них, в которых нужно преодолеть различного рода компромиссы (проблема занятости и переучивания работников из-за необходимости сохранить прибыльность предприятия, достижение оптимального решения за заданное ограниченное время и т.п.).

Полной автоматизации в первую очередь поддаются решения, основанные на суждениях. Они представляют собой знания и/или накопленный ответ, приводивший к успеху в прошлом, что побуждает их считать полезными в настоящем.

Принятие рациональных решений отличается тем, что они должны учитывать предшествующую историю развития технологии и/или предприятия и прогнозы развития событий. Поэтому информация для принятия решений базируется на результатах диагностики, которая позволяет сформулировать поле поиска альтернативных решений. Одновременно вырабатываются ограничения и критерии для принятия решений. К поиску рациональных решений побуждает и ограничение времени на их принятие, когда процесс выбора может прекращаться по критерию удовлетворения ограничений.

Риск в принятии решения должен опираться на оценки принимаемых решений по степени определенности их исхода, степень неопределенности может выражаться через вероятностную оценку. Ситуация значительной неопределенности наступает, когда невозможно оценить вероятность результатов. В частности, при изменении среды функционирования объекта в будущем, большими издержками времени и средств на получение информации, когда выгоды от ее получения сравнимы с затратами на ее добывание и обработку, трудность учета влияния принимаемых решений на технологию и условия труда и кадры.

Современные стандарты в области информационной безопасности, использующие концепцию управления рисками, очень важны в построении прикладных систем поддержки принятия решений.

Общие критерии предусматривают наличие двух типов требований безопасности: функциональных и доверия. Функциональные относятся к сервисам безопасности (идентификация, аутентификация, управление доступом). Требования доверия относятся к технологии разработки, тестированию, сопровождению, эксплуатационной документации и т.д. Работы должны производиться в выделенной вычислительной среде.

Интеллектуальная система – решатель творческих задач, который в своём составе имеет блоки: базу знаний, решатель и интеллектуальный интерфейс.

В технологии принятия решений часто рассматриваются интеллектуальные системы с вариантами принятия решений без участия человека и с участием ЛПР.

7.3. Рост сферы услуг в современном информационном обществе и его влияние на экономические и социальные процессы

Бизнес направляет усилия на борьбу за постоянного потребителя за счёт индивидуального учёта его запросов. Аналогично и государство в целом стало ориентироваться на потребности конкретного человека. Электронное правительство по своему замыслу должно предоставлять набор государственных информационных услуг гражданам, бизнесу, другим ветвям власти на основе готовых процедур их оказания. Тем самым оно становится частью электронной экономики. Например, реализация принципа «одного окна», предоставление общественных услуг гражданам (электронная почта, доступ к учебным и образовательным материалам, информация о продуктах и товарах, спорт и досуг, свободное распространение ПО, информация о банковском и страховом обслуживании и т.п.). Поэтому они в перспективе должны в какой-то мере контролироваться государством, чтобы защитить интересы граждан.

Информационные технологии сегодня рассматриваются как неотъемлемый компонент технологий управленческих, так как координация человеческой деятельности на современных предприятиях строится на основе новейших телекоммуникационных систем и компьютеров.

Традиционно под технологией понимается – формализованное описание деятельности, включающее набор ресурсов, инструментов, приёмов их использования и способов организации производства – необходимое и достаточное для воспроизводства процесса получения определённых продуктов, услуг и т.п. с заранее заданными параметрами. Глобальные перемены в сфере традиционных технологий произошли с появлением новой среды коммуникаций, приведшие к появлению особых форм взаимоотношений, названных сетевыми, которые реализуются на основе информационных технологий.

Информационная технология (ИТ) – совокупность методов, приёмов и средств, реализующих информационный процесс в соответствии с заданными требованиями. Они стали основным инструментом. Среди них выделяют базовые ИТ (реализуемые на уровне взаимодействия вычислительных систем, таких как ОС, ИНТЕРНЕТ, СУБД, криптозащита и т.п.) и прикладные ИТ, реализующие типовые процедуры обработки информации в конкретных предметных областях (автоматизация проектирования, управления).

Выделяют три вида информационного менеджмента: управление предприятием, внутренней документацией и публикациями.

В круг задач менеджмента входят также разработка, внедрение, эксплуатация и развитие автоматизированных информационных систем и сетей, обеспечивающих деятельность предприятия, взаимодействия с внешним информационным миром (сети, БД, издательства и т.п.).

Информационный менеджмент превращается в базовую технологию организации управленческой деятельности во всех сферах информационного общества.

Важнейшим аспектом в информационном обществе является организация труда отдельного работника, которая позволяет благодаря ИТ значительно повысить комфортность и производительность труда, по новому организовать его взаимодействие с фирмой.

К дистанционной работе относят такую, которая выполняется работником предприятия с использованием информационно-вычислительной техники и средств коммуникации всегда или время от времени вне пределов предприятия. Основным фактором является наличие связи работника и его рабочего места с предприятием.

Такие новые формы труда дают значительные экономические выгоды предприятию и повышают комфортность труда сотрудника и гибкость графика его работы, а также по исследованиям западных специалистов повышается производительность труда на 45 % и более.

Однако порождается и ряд новых проблем, связанных с контролем деятельности работника, сохранением конфиденциальности информации, оплатой труда, возмещением расходов работника по организации рабочих мест и за средства коммуникации. Такого рода работа имеет место и в условиях РБ. Например, сервисное обслуживание малыми предприятиями клиентов на дому, выполнение различных заявок, вручение подарков, подготовка информации по заданному кругу вопросов, программирование по заданию.

Интерес к дистанционной работе обычно проявляют высококвалифицированные и инициативные специалисты (программисты, инженеры, экономисты, переводчики, появляются свои особенности и в налогообложении, так как личная собственность может использоваться и для нужд предприятия и др.).

Объём современной мировой электронной торговли (*E*-коммерции) оценивается в несколько триллионов долларов. Её основой являются дистанционные сделки, осуществляемые, как правило, через Интернет на основе транзакций. Под транзакцией понимается любая сделка между хозяйствующими субъектами по циклу «запрос – выполнение задания – ответ», обладающая свойствами неделимости, согласованно-

сти, надёжности, изолированности. Неделимость означает, что операция должна выполняться полностью, либо немедленно при наличии каких-то причин и обстоятельств отменяться (принцип «всё или ничего»). Свойство согласованности подразумевает взаимную целостность данных в базах обоих партнёров, свойство надёжности означает, что при успешном выполнении транзакции ни при каких обстоятельствах данные не теряются. Изолированность означает, что доступ других транзакций до завершения выполняемой к используемым ею данным запрещён.

На основе понятия «транзакция» реализуется современная сетевая экономика, использующая бизнес-процессы, основой которых является обязательное взаимодействие субъектов через сети различного ранга (Интернет, Интранет, Экстранет и др.).

Исходной точкой для совершения сделок обычно является сетевая реклама, простейшей формой которой являются различные каталоги и специальные сайты «электронных» магазинов и предприятий-производителей продукции или услуг. Для ряда продаваемых продуктов реклама может носить специфический характер, который невозможно отразить за пределами электронных сетей. Специфика рекламы на базе сетевых операций содержит такой элемент, как использование медиасредств с одновременным воспроизведением объекта в действии, показом его различных возможностей с сопровождением демонстрации звуком, таблицами, текстом и т.д. Ещё более широкие возможности открываются в продаже программных и информационных продуктов, когда пользователь по сети может получить демонстрационную версию продукта и испытать её для своих условий, а также получить консультации и за умеренную оплату более совершенные версии продукта. По такого рода продуктам можно делать и заказы на их доработку под условия заказчика.

Особенность продаж предполагает наличие банковских счетов у партнёров, которыми они могут управлять дистанционно, а также наличие специальных средств фиксации оплаты сделки и необходимых перечислений за электронными подписями партнёров или банков, представляющих их интересы.

Под системой безналичных электронных расчётов будем понимать корпоративные сети для дистанционных коммерческих сделок, касающихся выполнения различных платёжных операций, как на основе пластиковых карт, так и путём управления счетами в банках по каналам связи непосредственно.

Основой таких операций являются счета в банках участников сделки. Они могут быть в разных банках одной страны и даже в зарубежных банках тоже. Корпоративная сеть подразумевает наличие ряда соглашений между её участниками, чтобы придать правовой статус любой сделке и должна обеспечивать конфиденциальность и надёжность сделок, а также проверку наличия денег на счетах и выполнения их перечислений в нужные адреса с выдачей подтверждений участникам сделки.

Отдельно рассмотрим платежи непосредственно с помощью пластиковых карт и аппаратуры и программных средств для управления счетами.

В первую очередь отметим общегосударственные эффекты от безналичных расчётов: уменьшение массы наличных денег и экономия на их печатании; экономия на процессах инкассации: уменьшение количества сделок без уплаты налогов, упрощение сделок, упрощение борьбы с преступностью типа грабежей касс и магазинов; краткосрочное кредитование за счёт хранения средств в банках.

Дистанционное управление процессами постепенно выходит на ведущие позиции. Например, БПС-Сбербанк предложил своим клиентам доступ к их банковскому счёту в течение 24-х часов на основе интернет-банкинга (оплата за квартиру, получение и использование пенсий через карт-счёт, оплату мобильных телефонов и ряд других платежей). В Норвегии к 2016 году планируется полностью отказаться от использования денежных знаков в платежах, за счёт применения систем безналичных расчётов.

Беларусбанк проявил себя как высокотехнологичная финансовая структура, выведя на рынок новый для отечественной банковской системы продукт с возможностью дистанционного управления счетом.

Внедряя новый продукт – «Удаленный депозит», он расширил линейку электронного обслуживания своих клиентов, помогая оперативно размещать в депозит временно свободные средства и дистанционно оперативно ими управлять. Дистанционное управление начали успешно использовать и в других сферах деятельности. Начали создаваться своеобразные виртуальные предприятия, ориентированные на электронную коммерцию.

Такое виртуальное предприятие представляет собой объединение географически разделенных экономических субъектов, взаимодействующих в процессе совместной деятельности на основе электронных средств коммуникаций для решения общих задач в рамках оговоренных сроков и полномочий. Смысл его создания состоит в повышении эффективности операций, связанных с электронной коммерцией:

– устранение влияния географического фактора для снижения затрат на связь с контрагентами;

– образование общего информационного пространства предприятия (доступ к ресурсам, активизация сотрудников в формировании и использовании информационных ресурсов);

– улучшение внутрифирменной координации при принятии решений.

Кроме того, повышается возможность сокращения сроков и улучшения качества выполнения заказа потребителя не за счёт организации нового производства, а поиска соответствующего нового партнёра. Поэтому на межгосударственном уровне потребуется правовое обеспечение такой деятельности, например, как при пересечении границ товарами, так и интеллектуальными продуктами.

Литература

[6, 9, 11, 13, 14, 22, 24, 26, 27, 29, 37, 41, 42]

Контрольные вопросы

1. Назовите особенности нанотехнологий, отличающие их от традиционных.
2. Что характерно для нанотранзисторов из различных материалов и биологических элементов?
3. Охарактеризуйте перспективы использования нанокомпьютеров и нанороботов.
4. Какие новые возможности открываются при применении нанотехнологий в мобильной связи?
5. В чём заключаются преимущества мобильной связи через спутники?
6. Почему возможны сенсорные измерения на уровне живой клетки наноприборами?
7. Какие новые функции стала обеспечивать мобильная цифровая связь в автоматизации быта и банковских платежей?
8. Какова цель использования сетецентрических технологий (GGG)?
9. Почему необходимо накапливать типовые решения для интеллектуализации технологий?
10. Назовите причины повышения реализации индивидуальных требований заказчиков.
11. Почему спутниковая мобильная связь стала использоваться в управлении на Земле?
12. Какие преимущества даёт переход к системе дистанционных безналичных платежей?

Глава 8. ПРАКТИЧЕСКИЕ ЗАДАНИЯ И РЕКОМЕНДАЦИИ

8.1. Общие указания

Содержание заданий практических занятий и лабораторных работ ориентировано на усвоение теоретических основ алгоритмизации управления на основе типовых задач дискретной математики и защиты информации в прикладной деятельности при создании важных документов для охраны интеллектуальной собственности в виде проектной документации и ведения систем безбумажного документооборота.

Учебное пособие построено с ориентацией на активную самостоятельную работу студентов за счет индивидуализации заданий. Для их выполнения приводятся необходимые минимальные теоретические сведения и схемы. Творческая часть состоит в создании цепочки конкретных вычислительных моделей на основе средств высокоинтеллектуальной системы Mathcad и формулировании выводов по результатам вычислительного эксперимента. Эта система позволяет быстро и эффективно реализовывать различные модели с участием экспериментатора.

Все задания можно условно разбить на три категории: освоение общего математического аппарата и программных систем, используемых в разработке информационных компьютерных технологий; получение навыков по применению базовых элементов математического и программного аппарата при моделировании простейших объектов; освоение методов защиты информации и их использования в создании документов и баз данных с электронной цифровой подписью. Особо следует выделить операции по обеспечению надежности коммерческих операций в сетях и идентификации (аутентификации) пользователей. Ключевыми элементами являются вопросы описания и реализации процессов. Система Mathcad (и её аналоги) хорошо подходит для первичного освоения навыков моделирования объектов и процессов, так как способствуют лучшему пониманию организации хода моделирования без его реализации в виде полной программы. Однако наличие развитой системы процедур позволяет реализовывать моделирование в смешанном человеко-машинном варианте, когда исследователь управляет процессом переходов между процедурами, описанными средствами Mathcad. По всем заданиям необходимо сделать краткое описание этого процесса и выводы о наблюдаемых его особенностях (регулирование количества итераций и точности, поведении заданной функции, причины остановки процесса или отсутствия решения).

Кратко охарактеризуем особенности ключевых элементов математического аппарата криптографии. В первую очередь – это аппарат, развиваемый в теории чисел, теории вероятностей и математической статистики.

Во многих операциях используется операция сравнения по модулю: $x = a \bmod p$, где a и p целые положительные числа, p – простое число, x при делении на p дает остаток a. На основе этого элемента легко показать реализацию односторонних функций, когда $y =$ (остаток) легко вычисляется при знании конкретного x, но обратная задача по y (остатку) вычислить x является многозначной, и только знание дополнительной секретной информации позволяет однозначно найти x. В том и другом случае решение прямой и обратной задач требует времени t как функции от некоторого полинома.

Вторым важным механизмом являются процедуры, основанные на применении подстановок и перестановок, чтобы менять исходный текст с использованием случайных чисел так, чтобы создать максимальные трудности криптоаналитику (злоумышленнику) при расшифровке закодированных сообщений.

В создании практических механизмов схем цифровой подписи и оплаты коммерческих сделок, а также банковских операций играют криптографические протоколы. Они применяются при обмене шифрованными сообщениями удаленных абонентов по открытым каналам связи. Часто бывает, что информация не является секретной, а для принимающей стороны (например, банка) важно убедиться, что информация действительно от его клиента, а клиенту важно убедиться, чтобы никто не изменил сумму в платежном поручении или кто-то послал поддельное поручение. Задача конфиденци-

альности информации решается ее шифрованием, а ее целостность и аутентификация участников при обмене сообщениями обеспечивается протоколами (совокупностью алгоритмов, которые скрупулезно должны выполнять участники обмена информацией).

Особую роль в протоколе для идентификации подписавшего сообщение лица играют хэш-функции для преобразования сообщений произвольной длины в более краткие хэш-значения требуемой длины.

Для создания дополнительных трудностей в расшифровке сообщений и аутентификации их авторов широко используются и генераторы случайных чисел, например, при наложении гаммы (случайных битов).

Варианты всех заданий подбираются в зависимости от индивидуальных характеристик студентов: Ф, И и О – количество букв в фамилии, имени и отчестве соответственно, № – численное значение номера в группе (подгруппе).

8.2. Задания для практических занятий

1. Составление численных и логических алгоритмов с демонстрацией их основных свойств.
2. Выбор машинно-ориентированных численных методов решения задач. Связность и восстанавливаемость алгоритмов
3. Разработка программы сложения двух чисел на машине Поста.
4. Задание с помощью конечного автомата движения робота по контуру.
5. Оптимизация многомерных функций.
6-8. Построение логистического алгоритма доставки груза на основе решения задачи коммивояжера методом ветвей и границ для трёх модификаций алгоритма с целью получения навыков адаптации теоретической модели к решению прикладной задачи.

Выполнение заданий №№ 1-8 предполагает обязательное описание студентом алгоритма любым известным ему методом (например, из [14]). Теоретическая основа для выполнения заданий №№ 1-5 известна из лекционного курса и достаточно полно описана в [14, 27]. Поэтому для этих заданий подчеркнём лишь ключевые моменты, которые желательно выделить при оформлении результата:

– № 1. Главное отличие численных и логических алгоритмов, обеспечение конечности, однозначности и результативности процесса вычислений;

– № 2. Особенности ЭВМ как вычислительного инструмента (учёт погрешностей из-за ограниченной разрядной сетки и перехода к двоичной форме представления чисел, необходимость экономии памяти при экспоненциальном росте промежуточных результатов, желательность автоматизации ввода больших массивов информации, возможность сбоев в работе, необходимость предусматривать эффективные пути восстановления в случае чрезвычайных ситуаций и т.д.).

– № 3. В этой задаче в отчёте надо отметить сходство в программировании и реализации программ на абстрактной машине Поста и реальной ЭВМ.

– № 4. Отметить особенности использования конечного автомата как универсального устройства для управления заданным классом процессов (движение по выбранному типу контуров) на базе описания переходов состояний автомата в виде своеобразной программы.

– № 5. Необходимо усвоить диалоговые механизмы выработки рациональных решений.

– №№ 6-8. В этом (самом сложном) задании надо описать собственные предложения по построению деталей механизмов вычислений в пределах общей схемы реализации метода ветвей и границ в рамках каждого из этапов (построение и улучшение рекорда, выбор принципов оценки вершин, модификация метода под простейшую прикладную ситуацию).

Формулировки вариантов заданий с пояснениями даны ниже.

№ 1. Описать алгоритм Евклида и проверить его работоспособность на примере для положительных чисел a = 4Ф и b = 2И. В соответствии с рис. 1.1.1 найти по алго-

ритму поиска пути в лабиринте при их существовании между вершинами А и Ф, А и (Ф + О). Для использования номеров А, Ф, (Ф + О) приведём цифровые кодировки вершин графов (лабиринта): $A = 1$, $B = 2$, $L = 3$, $H = 4$, $K = 5$, $C = 6$, $D = 7$, $N = 8$, $R = 9$, $F = 10$, $E = 11$, $M = 12$ (при Ф + О более 12 номер вершины выбирается 11 при нечётном Ф + О и 12 – в противном случае.

№ 2. Описать алгоритмы вычисления квадратного корня из $x > 0$ итерационным методом и малосвязанный алгоритм получения значения полинома, а также проверить их работоспособность для заданных условий: итерационный метод реализуется на основе формул ($y_0 = x$, $E = 0.1$, $y_i = 0.5(y_{i-1} + x / y_{i-1})$, $|y_i - y_{i-1}| \le E$; $x(test)$ = Ф + И + О, №4 – шаг для сбоя, чтобы продемонстрировать устойчивость алгоритма к сбоям); для второй задачи использовать полином ($y = a_n x^n + a_{n-1} x^{n-1} + ... + ax + a$) и известную схему для понижения связности алгоритма: $T_i = T_{i-1} x$, $y_i = y_{i-1} + a_i T_i$, $T_0 = 1$, $y_0 = a_0$ (данные для реализации и тестирования алгоритма: x = О, $n = 4$, a_0 = Ф, a_1 = И, a_2 = О, $a_3 = 2$, $a_4 = 1$).

№ 3. Запрограммировать алгоритмы для методов перекатывания и переноса меток при сложении двух расположенных подряд чисел a и b [27]. Исходные данные для проверки работоспособности программ: головка машины Поста находится над последней меткой первого числа, a = Ф, b = И.

№ 4. Запрограммировать движение робота по контуру со следующими координатами угловых точек: (0,1); (0,И); (1,И); (1,И/2); (2,И/2); (2,1). Точка загрузки детали (0,И) и точка её выгрузки (1,И/2). Исходные данные и пример программы для управления роботом описаны в разделе 1.5.

№ 5. Найти максимальное при Ф-четном (или минимальное при Ф-нечетном) значение $F(x,y,z) = xz^2 - xy + (y - 5)^5$; когда $2 \le x \le$ Ф, $3 \le y \le$ И, $3 \le$ Z $\le$ О методом покоординатного спуска (параграф 4.2).

№№ 6-8. Освоить реализацию метода ветвей и границ с прикладной ориентацией, используя в качестве исходной для выбора своего варианта таблицы, которая приводится далее по тексту.

Метод ветвей и границ в задаче коммивояжера используется во многих модификациях. Его относят к сложным дискретным задачам, которые в худшем случае точно решаются при полном переборе всех вариантов решений. Такого типа задачи называют NP полными. Их пытаются решать различными методами и используют также в экспериментах и для решений нейросетевыми методами. Таковой является и задача коммивояжера. Ее постановка: имеется n городов, между которыми заданы расстояния. Требуется из заданного города объехать все города по кратчайшему замкнутому маршруту, побывав в каждом городе только 1 раз и возвратиться в исходный город.

Суть метода ветвей границ заключается в развитии дерева решений по основному закону, когда следующая вершина дерева всегда строится в перспективном направлении на основании лучшего значения избранной функции оценки F, которая состоит из двух частей: $F = F_{фактич} + F_{оптимистич}$.

Трудность решения задачи состоит в подборе функции быстровычислимой $F_{оптимистич}$ и получении любым авторским методом одного начального конечного решения $F = F_{фактич} + F_{оптимистич} (=0)$.

Название метода связано с тем, что из одной избранной вершины строятся все продолжения и получаются их оценки (ветвление) и после получения оценок F вычеркиваются те вершины, которые имеют оценку $F \ge F_{TR}$ (текущий рекорд – лучший результат решения задачи на данный момент).

Типовая структура описания графа: в вершине (кружок, прямоугольник) располагается три цифры: вверху номер достигнутой вершины, слева $F_ф$, а справа F_0, а на ветви – номер вершины, из которой ветвь идет. Конечная вершина отмечается прямоугольником, на котором записано только $F_ф$, так как $F_0 = 0$.

Покажем решение задачи этим методом, когда F_{TR} получено по принципу FIFO (первым пришел – первый обсуживается), а F_0 равно сумме всех минимальных путей для конкретной вершины с учётом уже пройденного пути.

Разработка авторских вариантов алгоритмов решения задачи коммивояжера методом ветвей и границ преследует несколько целей:

1. Освоить подходы к выработке собственных принципов разработки построения вспомогательных функций для повышения эффективности вычислений (прекращение решения по времени, сокращение числа вариантов, выявление нерешаемости из-за жестких требований и др.). Смысл выполнения задания заключается в описании элементов алгоритма и проведении практических расчетов на основе авторских оценочных функций и методов расчетов F_{TR}.

2. Предложить модификацию алгоритма, когда появляется некоторое ограничение, например, время прибытия не позже заданного для ограниченного количества точек (до начала обеда в школе, после дойки коров, ремонтные работы на дорогах в заданные интервалы времени и т.п.).

Отчетность по работе:

1. граф-схема с отображением всего процесса вычислений по алгоритму до заключительного шага. В записях отобразить списки вершин на момент перехода с яруса на ярус, а так же значение оценочных функций;

2. записать оптимальный маршрут (порядок прохождения точек);

3. описать свои модификации алгоритма и привести результаты вычислений, критически оценить эффективность алгоритма.

Таблица № 8.2.1 – Данные для решения задачи

j \ i	1	2	3	4	5	Min
1	///	1	2	6	И	1
2	1	///	3	5	3	1
3	3	4	///	Ф	4	3
4	4	4	5	///	4	4
5	3	4	5	3	///	3

Примечание: Ф – количество букв в фамилии
И – количество букв в имени

8.3. Задания для лабораторных работ

Этот раздел включает задания для 11-ти лабораторных работ, которые необходимы для усвоения типовых элементов теории моделирования на ЭВМ базовых механизмов при реализации интеллектуальных компьютерных технологий. Основным средством выполнения работ является система Mathcad, так как она содержит ряд полезных процедур для построения и адаптации моделей: символьное и численное решение большого класса задач прикладной математики, аппарат для реализации случайных процессов, механизмы округления чисел до целых с избытком и недостатком, возможность гибкого построения графиков в нужном масштабе и с одновременным построением графиков нескольких функций на одном чертеже. Наличие этих возможностей полезно и для построения криптографических процедур защиты информации, аутентификации пользователей, генерировании электронной цифровой подписи и т.д., и для понимания механизмов защиты передаваемой информации в сетях ЭВМ.

Лабораторная работа № 1. Построение графиков полиномов $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ на отрезке $[p, r]$ в системе Mathcad.

Выполнение задания опирается на типовую структуру системы Mathcad для построения графиков в декартовой системе координат, которая, будучи однажды задана, позволяет управлять процессом вычислений для получения графика заданного полинома в изменённом масштабе для любого подотрезка, вызывающего интерес исследователя.

Исходя из этой возможности, следует построить необходимые графики для ответов на следующие вопросы:

1. Найти графическим методом отрезки убывания и возрастания полинома.

2. Найти приближённо, с точностью до ε = 0.1, корни полинома.

3. Описать своими словами схему решения задачи и сделать выводы об особенностях исследований в вашем варианте функции.

4. Вычислить полином в корневых точках.

Одна из возможных схем построения декартова графика полинома приведена ниже.

Задаем количество точек на данном отрезке.

$n := 50$ (при необходимости его можно в дальнейшем изменять).

$i := 0..50$ (две точки вводятся нажатием клавиши «ж» (в MATCAD), запись обеспечивает цикл для вычислений 50 точек графика).

Далее задаём свой вариант

$a :=$ $b :=$ $c :=$ $d :=$

и координаты начала и конца отрезка, на котором исследуется функция:

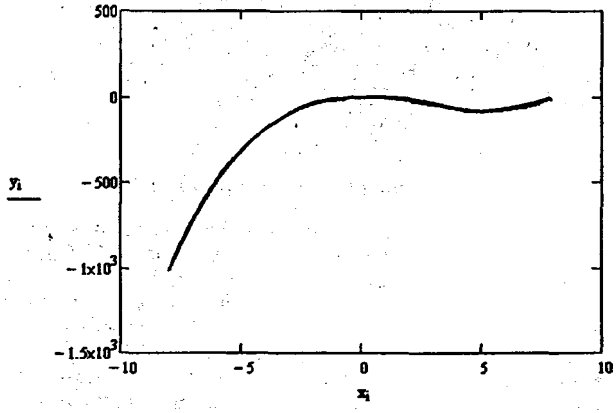
$p := -8, r := 8.$

Определяем текущие значения x и y в рассматриваемой точке:

$$x_i := p + \frac{r-p}{n} \cdot i$$

$$y_i := a \cdot x_i^3 + b \cdot x_i^2 + c \cdot x_i + d.$$

После этого строим график. Для этого на панели инструментов выбираем «График» и в появившемся окне выбираем «График *X-Y*». После появления его шаблона на чёрных прямоугольниках заполняем x_i и y_i и щёлкаем левой клавишей мыши за пределами графика, что дает возможность получить первый график функции с целью определения последующих отрезков для более тщательного её исследования.



Новый исследуемый отрезок задается изменением p и r, и результат отображения функции получается после щелчка рядом с графиком.

Варианты для выполнения лабораторной работы даны в таблице 8.3.1.

Таблица 8.3.1 – Выбор варианта

№ варианта	*a*	*b*	*c*	*d*
1	**2**	**3**	**4**	**5**
1	1	−8	−1	1
2	1	−8	1	1
3	2	−16	1	1
4	1	−15	1	1
5	1	−9	1	15
6	2	−20	1	15

Продолжение таблицы 8.3.1

1	2	3	4	5
7	2	15	1	1
8	1	5	1	1
9	1	4	1	1
10	1	2	−1	1
11	1	−3	−4	−4
12	1	−2	0	−4
13	1	-2	1	-5

Оформление и предъявление отчета возможно в двух вариантах: в печатном, что лучше, или на компьютере на сайте студента, где будут храниться все лабораторные работы, которые дублируются на флешке студента с целью обеспечения их сохранности.

Лабораторная работа № 2. Решение уравнений *n*-й степени от одной переменной, содержащих простейшие трансцендентные функции (тригонометрические, логарифмические, показательные) на заданном отрезке $[p, r]$ типа $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 - k\sin(tx)$; $\ln tx$; k^{α}.

Требуется решить задачу для своего варианта тремя путями:

1. Графическим методом с точностью до 0,1 с одновременным построением графиков на заданном отрезке для двух функций (полинома и трансцендентной) найти координаты их точек пересечения. В качестве корней записываются все точки пересечения двух графиков, построенные по общей процедуре в Mathcad:

$n := 50$

$i := 0..50$

Далее для своего варианта указываются конкретные значения коэффициентов и координаты начала и конца отрезка, на котором исследуется функция:

$a :=$ $b :=$ $c :=$ $d :=$ $k :=$ $t :=$

$p := -8$

$r := 8$

Определяется текущее значение x_i, общее для двух графиков, и значения функций y_i и z_i:

$$x_i := p + \frac{r - p}{n} \cdot i,$$

$$y_i := a \cdot x_i^3 + b \cdot x_i^2 + c \cdot x_i + d,$$

$z_i := k\sin(tx_i)$.

После этого строим декартов график. Для этого на панели инструментов выбираем «График» и в появившемся окне выбираем «График *X-Y*». После появления его шаблона на чёрных прямоугольниках внизу, заполняем x_i, а слева посередине записываем y_i, ставим запятую и далее z_i. Щёлкаем мышью рядом с графиком, что дает возможность получить первый график для определения отрезков для более тщательного исследования расположения корней уравнения. При необходимости путем изменения длины исследуемого отрезка получают более точные значения корней и записывают их с нужной точностью.

2. Каждый корень находится вновь с помощью процедуры root для отыскания одного корня. В качестве его начальной величины подставляются в порядке следования корни, найденные графическим путем и соответствующие отрезки для них. Последовательно фиксируются результаты, полученные с помощью процедуры root. Типовая структура для решения задачи (исходные данные используются те же: коэффициенты и длины отрезков), так как они фиксировались ранее и их можно использовать снова. Перед процедурой root задается соответствующее значение x, равное значению корня, и отрезок $[p, r]$, на котором этот корень был найден, т.е.:

$x :=$

$f(x) := a \cdot x^3 + b \cdot x^2 + c \cdot x + d - k\sin(x)$

root ($f(x)$, x, p, r), где p и r – границы найденного корня, и щелчком получается результат.

3. Регулирование точности вычислений при применении процедуры root и любых других по количеству знаков делается так: открывается меню «формат», затем подменю результат и указывается требуемое число десятичных знаков в открывшемся окне (5 пять! 5 набирается с клавиатуры). После чего пересчитываются все корни с применением процедуры root.

4. Вычисляются все значения функции для всех найденных корней. Делаются выводы о степени приближения результатов к нулю в зависимости от точности. Для отчета по работе описывается ход эксперимента, и делаются выводы.

Варианты для выполнения лабораторной работы даны в таблице 8.3.2.

Таблица 8.3.2 – Выбор варианта

№ варианта	a	b	c	d	k	t
1	1	−8	−1	1	5	2
2	1	−8	1	1	6	2
3	2	−16	1	1	7	2
4	1	−15	1	1	5	3
5	1	−9	1	15	6	3
6	2	−20	1	15	7	3
7	2	15	1	1	5	4
8	1	5	1	1	6	4
9	1	4	1	1	7	4
10	1	2	−1	1	5	5
11	1	−3	−4	−4	5	6
12	1	−2	0	−4	5	7
13	1	−3	1	1	5	8

Лабораторная работа № 3. Символьные и численные решения уравнений и неравенств, содержащих функции алгебраического типа.

Требуется освоить применение символьных вычислений с последующим получением конкретных результатов численными методами. Задача решается для функций в виде полиномов $P(x)$ и алгебраических дробей $P(x) / Q(x)$ в соответствии с заданными вариантами

Mathcad является полезной системой для изучения в рамках специальности «Искусственный интеллект», так как содержит процедуры символьных вычислений, когда результаты отображаются в символьной форме: вычисления производных, интегралов, уравнений, неравенств и др. Это позволяет решать ряд задач в общей алгебраической форме, а потом подбирать значения нужных параметров и вычислять конкретные значения для найденных решений в символьной форме. Для этой цели используются палитры символы, вставить функцию, матрица, математика и другие.

Отбор основных функций для изучения диктуется тем, что решение неравенства часто связано с необходимостью решить уравнение, упростить функцию в ее символьной записи или сделать преобразование для толкования результатов.

К таким процедурам для более глубокого изучения отнесем: simpflay (упростить), solve (решить уравнение или неравенство), factor (представление выражений в виде сомножителей), expand (разложение рациональных дробей $P(x) / Q(x)$) и некоторые другие.

Отбор и освоение этих инструментов диктуется необходимостью освоения полезной информации и навыков в решении неравенств и уравнений, содержащих в своем составе элементарные алгебраические функции. Главным объектом исследования яв-

ляются полиномы, так как для них наиболее развит аппарат символьных вычислений. В частности, получение решения уравнений в символьной форме и различные преобразования алгебраических выражений. Поэтому основными исходными объектами для исследования выбираются алгебраические уравнения и неравенства типа:

$P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 (= 0$, или > 0, или $< 0)$, а так же алгебраические дроби типа $P(x) / Q(x)$, содержащие полиномы в числителе и знаменателе.

Конкретный вид объекта определяется вариантом лабораторной работы, который предусматривает получение символьного решения, а затем по полученному объекту с подстановкой данных варианта конкретного решения. Желательно проанализировать возможные границы их колебаний в рамках существования полученного решения и сделать выводы.

Задания для изучения имеют следующую структуру:

1. $P(x) = ax^3 + bx^2 + cx + d$ ($=0$, или > 0, или <0).
2. $P(x) / Q(x) = (ax^3 + bx + c) / (ax^2 + bx + c) > 0$.

По первому заданию получить символьное и численное решение, воспользовавшись процедурой solve с представлением d в буквенной форме при получении символьного решения. Далее против строки слева с символьным решением присваивается конкретное значение d, вычисляются корни, и после этого записывается решение неравенства $P(x) > 0$ по известным из средней школы методам. Далее при конкретных a, b, c, d для полинома $P(x)$ применяется процедура solve и получается решение неравенства в символьной форме. Требуется сравнить оба результата.

По третьему заданию вычисляется предел дроби $P(x) / Q(x)$ с помощью оператора lim, когда x стремится к n, а затем вычисляется лимит при $n = d$.

Описать ход работы и сделать выводы.

Таблица № 8.3.3 – Выбор варианта задания:

№ варианта	a	b	c	d
1	1	2	−1	−2
2	1	0	−3	−2
3	1	−3	−1	3
4	1	−4	−1	4
5	1	−1	−4	4
6	1	−3	−4	12
7	1	−4	−4	16
8	1	−6	3	10
9	1	−6	11	−6
10	1	−7	14	−8
11	1	−8	17	−10
12	1	−9	20	−12
13	1	−8	19	−12

Лабораторная работа № 4. Символьное дифференцирование функций в системе Mathcad и его применение (частные производные в градиентных методах оптимизации).

Требуется вычислить определитель 3-го порядка в символьной форме, содержащей цифровые и буквенные элементы (x, y, z). От полученной функции $F(x, y, z)$ найти частные производные F'_x, F'_y, F'_z как будущие компоненты градиента (вектора, указывающего своим направлением наискорейшее возрастание функции: $F'_x \cdot \vec{i} + F'_y \cdot \vec{j} + F'_z \cdot \vec{k}$). Далее вычислим их нужное количество раз в требуемых точках. Найти локальные минимум и максимум $F(x, y, z)$, когда $a-\varepsilon \le x \le a+\varepsilon$, $b-\varepsilon \le y \le b+\varepsilon$, $c-\varepsilon \le z \le c+\varepsilon$. Начальная точка для вычисления частных производных равна количеству букв в Ф, И, О, т.е. $x = \Phi = a$, $y = И = b$, $z = О = c$, $\varepsilon = 0{,}1$. Первая строка определителя – (1, x, 2), вторая – (3, 4, y), третья – (z, 5, 6). Сделать вывод о возможном существовании опре-

делителя, равного нулю, так как смысл решения задачи состоит в установлении факта возможности отсутствия решения системы линейных уравнений из-за того, что переменные x, y, z замерялись приборами с ошибкой ε, и в скрытой форме может существовать определитель, равный нулю.

Опишем процедуры отыскания локального минимума (максимума): в полученной текущей точке вычисляются знаки производных по переменным x, y, z для построения следующей точки. Учитывая, что структура исследуемой функции $F(x, y, z)$ такова, что ее максимум (минимум) достигается на концах отрезков области изменения переменных, то допустимо применение метода наискорейшего спуска, когда по знаку частной производной для данной переменной больше нуля при поиске минимума выбирается левая сторона отрезка, а для максимума правая и, наоборот, в противном случае. Процесс поиска обрывается при зацикливании, т. е. при получении одной и той же точки (x, y, z) второй раз. Одновременно ведется вычисление значения функции в каждой полученной точке. Зацикливание соответствует прекращению нарастания (убывания) функции. Все шаги протоколируются по итоговым результатам. Процедура вычисления определителя представляет собой символьные действия над векторами и матрицами (вычисление определителей, транспонирование и др.). Будем рассматривать все операции над матрицами, так как вектор можно считать однострочной или одностолбцовой матрицей. Матрица может задаваться следующим образом: на панели управления набирается «вставка» и в ее меню отмечается матрица и на экране можно задать количество строк и столбцов. Аналогичный результат получим сразу, если щелкнем в меню по изображению незаполненной матрицы. Заполним конкретными данными матрицу $n \times n (3 \times 3)$. Через панель управления (символика) можно вызвать подменю «матрица» и в нем выбрать нужную операцию (транспозиция, инвертирование, определитель).

Обратившись к меню символика, выполним операцию по вычислению определителя и в итоге получим $F(x, y, z) = 54 - 5y - 18x + zxy - 8z$, соответствующее значению $\det A$ в символьной форме. После этого вычислим в символьной форме общий вид частных производных F'_x, F'_y, F'_z и, подставляя в них конкретные значения текущих точек, приступим к процедуре выполнения оптимизации.

Описать процесс вычислений и сделать выводы. Дополнительно для связи с задачами, решаемыми в курсе высшей математики, вычислить обратную и транспонируемую матрицу для вашего задания. Для контроля результата лабораторной работы можно применить процедуру Minimize (Maximize) по следующей схеме:

$F(x, y, z)$:= значение символьного определителя.

$x := 2; y := 5; z := 8$ (любые числа по возможности близкие к min, max)

$a - \varepsilon \le x \le a + \varepsilon$

$b - \varepsilon \le y \le b + \varepsilon$

$c - \varepsilon \le z \le c + \varepsilon$

P := (Minimize) (f, x, y, z), P = (обыч. равно) $f(P_0, P_1, P_2)$=(обыч. равно = ставится в момент чтения).

Лабораторная работа № 5. Отыскание экстремумов функций многих переменных в системе Mathcad.

Требуется вычислить max и min $F(x, y, z) = x^3y^2z - 2x^2yz + 3xy + c$. От полученной функции $F(x, y, z)$ найти частные производные как будущие компоненты градиента (вектора, указывающего своим направлением наискорейшее возрастание функции: $F'_x \cdot \vec{i} + F'_y \cdot \vec{j} + F'_z \cdot \vec{k}$). Далее вычислим их нужное количество раз в требуемых точках. Найти локальные минимум и максимум $F(x, y, z)$, когда $\Phi - 2 \le x \le \Phi + 2$, $И - 2 \le y \le И + 2$, $О - 3 \le z \le О + 3$. Начальная точка для вычисления частных производных равна $(b - a)\gamma$ (вычисляется как $x(y,z) = a + + rnd(b - a)$, где $(b - a)$ – величина отрезка для соответствующей переменной; Ф, И, О – количество букв в фамилии, имени, отчестве.

Опишем процедуры отыскания локального минимума (максимума): в полученной текущей точке вычисляются знаки производных по переменным x, y, z для построения следующей точки. Учитывая, что структура экстремумов исследуемой функции $F(x, y, z)$ неизвестна, требуется испытать разный шаг h по каждой переменной, по знаку частной производной для данной переменной больше нуля при поиске минимума выбирается движение к левой стороне ($x - h$) отрезка, а для максимума – к правой ($x + h$) и, наоборот, в противном случае. Процесс поиска обрывается при зацикливании, т.е. при получении одной и той же точки (x, y, z) второй раз (при выходе за пределы отрезка в качестве значения переменной выбирается пересекаемая граница). Одновременно ведется вычисление значения функции в каждой полученной точке. Зацикливание соответствует прекращению нарастания (убывания) функции. Все шаги протоколируются по итоговым результатам. Процедура вычисления F служит дополнительным критерием движения к цели.

Вычислим в символьной форме общий вид частных производных F'_x, F'_y, F'_z и, подставляя в них конкретные значения текущих точек, приступим к процедуре выполнения оптимизации. Рекомендуем начать с шага h=1, понижая его в процессе после зацикливания постепенно до уровня 0.1.

Описать процесс вычислений и сделать выводы. Для контроля результата лабораторной работы можно применить процедуру Minimize (Maximize) по следующей схеме:

$f(x, y, z) := x^3y^2z - 2x^2yz + 3xy + c$

x:=; y:=; z:= (любые числа по возможности близкие к точкам min, max)

$\Phi - 2 \le x \le \Phi + 2,$

$И - 3 \le y \le И + 3,$

$O - 3 \le z \le O + 3$

P:= Minimize (f, x, y, z) P= (обычное равно) $f(P_0, P_1, P_2)$ = (обыч. равно = ставится в момент чтения результата).

Вторая попытка делается из новой случайной точки ($a + rnd(b - a)$) при повторном решении первым методом.

Лабораторная работа № 6. Моделирование информационных процессов со случайными компонентами.

Освоить механизм моделирования независимого от исследователя выбора хода процесса методом статистических испытаний, сделать выводы об особенностях таких вычислений с учётом излагаемой далее теории. Модель – представляемая мысленно или материально реализованная система, которая в процессе познания, анализа замещает реальный объект, сохраняя важные для исследования его черты. По сути дела модель является упрощенным образом реального объекта для более глубокого изучения действительности. Метод исследований, базирующийся на разработке и использовании моделей, называется моделированием. Модель всегда предполагает участие в ее создании (конструировании) исследователя, так как на нем лежит ответственность за адекватность модели объекту с позиции изучаемых его свойств.

Смысл использования моделей состоит в упрощении и удешевлении процессов изучения, а иногда их использование практически представляет почти единственный путь к получению необходимых результатов (угроза взрыва, дороговизна натурного эксперимента и т.п.). Обычно различают методы материального (существует связь с материальными объектами модели) и идеального моделирования (рассматриваются мысленные связи между объектом и моделью).

Среди формализованных моделей выделяют знаковые модели, в которых используются системы знаков с правилами их преобразования и интерпретации. Его частным случаем является математическое моделирование, которое характерно тем, что разные процессы (объекты) могут описываться одинаковыми формулами и уравнениями. Например, законы о течении жидкостей и газов, расширении твердых тел и т.п.

Во всех случаях и видах моделирования соблюдается главный принцип: модель должна соответствовать оригиналу в главном только в том, что интересует исследователя, а несущественное можно отбросить.

Математическое моделирование наиболее важно при использовании вычислительной техники. Это связано с тем, что с его помощью можно получить доказательство существования решения задачи и оценить время получения результата, диапазон изменения переменных, возможную точность результата и т.д.

При математическом моделировании на ЭВМ возможно формировать базы данных для других задач, из типовых элементов – строить прочие модели. Корректное построение всех видов моделей невозможно без участия специалистов: выбор переменных, формирование ограничений.

Среди многочисленных видов моделей важнейшая роль отводится моделям принятия решений и их оптимизации. Наибольшую трудность обычно вызывают задачи многокритериальной оптимизации, которые поддаются получению приемлемых решений с применением ЭВМ, а иногда и с участием человека, который формирует путь поиска решений.

Особая роль отводится вероятностным моделям и реализации на их основе вероятностных процессов. Эти модели используются в имитационном моделировании, когда требуется изучить характеристики процесса, на который влияют случайные факторы, путем его многократного воспроизведения по элементарным шагам и фиксирования различных средних характеристик в целом путем подсчета количества воспроизводимых явлений в изучаемом процессе (например, отыскание емкости резервного бункера заготовок для обработки на автоматической линии по причине поломок или затупления инструмента, поиска показателей реальных случайных процессов для статистических оценок разброса измерений и т.п.).

Вероятностное моделирование на ЭВМ обычно опирается на использование генератора случайных чисел равномерно распределенных в диапазоне (0, 1) или (0, a), который содержится в большинстве вычислительных систем в виде стандартной процедуры. На его основе обычно строятся случайные процессы с использованием стандартных процедур для воспроизведения элементов моделей в виде конкретных случайных законов (экспоненциальный, нормальный и др.)

Второй путь использования генератора случайных чисел состоит в выборе случайных условий или путей продолжения процессов, когда требуется исключить влияние исследователя на выбор текущего продолжения процесса, например, стартовых точек для различных итерационных процессов вычислений для многомерных функций и т.п.

Информационные модели – знаковые модели, описывающие процессы получения, передачи, преобразования и использования информации. Их основой во многих случаях являются вычислительные модели, на базе которых разрабатываются алгоритмы и программы. При машинном моделировании всегда реализуется знаковые модели. Иногда говорят, что компьютерное моделирование – это моделирование теории изучаемого объекта. В этом смысле нейронная сеть является теорией для грубого представления функционирования мозга. Имитационные модели воспроизводят (имитируют) процессы функционирования объекта. Программа для ЭВМ – формализованная информационная модель деятельности. Фотография – статическая информационная модель.

В данной работе предлагается повторить лабораторную работу № 5 многократно при случайном выборе координат 5-ти начальных точек (в вашем варианте функция $F(x, y, z) = = x^3y^2 + z + 3x^2 y - yz\,c$ областью определения $Ф - 2 \le x \le Ф$, $И - 3 \le y \le И$, $О - 1 \le z \le О$), градиентным методом найти минимум и максимум функции (лучшие результаты из 5 попыток, их же вычислить с помощью процедуры Maximize (Minimize). Шаг изменения каждой координаты может быть разным в диапазоне от 1 до 0.1. Протокол одной попытки поиска результата можно представлять в укрупнённой форме в виде логической цепочки векторов, каждый из которых должен содержать значения компонент: координаты точки, значение функции и частных производных, рекомендуемый следующий шаг по каждой координате(0 или $-h$ или $+h$). При обобщении шагов координаты одного знака суммируются. Для эффективной работы полезно построить такую структуру: первая строка – переменные с присвоением текущих значе-

ний координат, вторая строка – функция в алгебраической форме и в 3-5 строках аналогично записываются частные производные. Такая структура позволяет сразу вычислить весь вектор за исключением шагов, подбираемых исследователем. Оформить выводы, обобщив результаты всех попыток и проверок.

Лабораторная работа № 7. Моделирование решения задач методом Монте-Карло.

Освоить метод вычисления площадей и сложных функций на основе моделирования случайных процессов, осветить проблемы достижения точности результатов и провести их сравнительный анализ с обычными численными методами для различных типов функций.

Метод Монте-Карло (*статистических испытаний*) – численный метод решения математических задач при помощи моделирования случайных процессов и событий. Его название происходит от города Монте-Карло, известного своими игорными домами.

Для реализации случайной величины используют датчики, генерирующие случайную последовательность чисел, равномерно распределенных на интервале (0, 1). Различают чаще всего два типа генераторов случайных чисел: физические датчики (источник – реальные физические процессы типа шумов электронных приборов, радиоактивные излучения и т.п.); программные датчики *псевдослучайных чисел (эти числа по своим свойствам практически неотличимы от полученных физическими датчиками).*

Чтобы обеспечить получение цепочки случайных событий, обычно их воспроизводят, используя набор стандартных программ для моделирования известных случайных событий с фиксированными параметрами (математическое ожидание, дисперсия и т.п.).

Это порождает проблему доказательности доверия к результату, полученному методом Монте-Карло. Поэтому в серьезных исследованиях проверяются источники моделирования на соответствие теоретическим параметрам при выполнении заданного количества вычислительных опытов (обычно от 100 до 10 000). Это связано с тем, что ошибка вычислений обычно пропорциональна величине $1/\sqrt{N}$, где N – количество опытов, например, при 400 опытах можно ожидать ошибку до 0,05 от конечного результата.

В данном случае ставится задача проверить на пригодность модели для получения чисел, распределенных по равномерному закону, который моделируется для базовых параметров в интервале (0, 1) и математического ожидания 0,5 и среднего квадратичного отклонения 0,083.

Практическое применение в различных математических задачах метод нашел при вычислении площадей, ограниченных сложными контурами, когда обычным интегрированием решить задачу очень сложно, а иногда отсутствуют и готовые методы ее решения. Идея предлагаемого метода заключается в следующем: сложную фигуру неизвестной площади помещают в квадрат (круг, прямоугольник), площадь которого известна. После этого моделируют вбрасывание N случайных точек в квадрат, и фиксируют количество точек n, попавших в искомую фигуру. При большом числе точек справедливо равенство: $S_{\text{фигуры}}/S_{\text{квадрата}} \approx n/N$. Этот способ можно использовать для проверки качества генератора случайных чисел из интервала (0, 1).

Для этих целей возьмем квадрат со стороной 1 и впишем в него четверть круга единичного радиуса. Тогда $S_{\text{кв}}=1$; $\frac{1}{4}S_{\text{кр}}=\frac{1}{4}\pi R^2=\frac{1}{4}\pi$. Тогда при получении n/N приближенно равно 0,78 ($\pi/4$) можно сделать вывод о работоспособности генератора. С целью выполнения такой проверки вбрасывают N случайных точек. Если хотят обеспечить точность 0,01, то для доверия к результату необходимо вбросить около 10 000 точек.

Проверить на практике, при скольких точках может быть достигнута заданная точность E и построить график зависимости точности от N (N = 100, 200, ... 1000). Один из возможных вариантов реализации этой проверки можно осуществить по следующей схеме, позволяющей получить для любого N декартов график и n/N:

N:= T_0:=0

i:=1..N

t_i:=(rnd(1))^2+(rnd(1))^2
T_i:=T_{i-1}+(2 − ceil(t_i))
K_i:=i
R_i:=T_i/K_i

Замечания: N − задаваемое количество точек, t_i − сумма квадратов координат текущей точки, T_i − попавшие в четверть круга точки, R_i − результат, на графике по оси x записывается K_i, а по оси y − R_i. Для вычисления интегралов материал подбирается самостоятельно из курса высшей математики. Для $f(x) > 0$ на отрезке (a,b) можно при $a > 0$, max $f(x) = c$ воспользоваться формулой $S = c(b - a)n/N$. В этом случае в качестве x вбрасывается координата (a + rnd (b − a)), а в качестве y − координата rnd(c). Считается, что испытание прошло успешно (случайная точка попала под кривую или на неё: $f(a$ + rnd(b− −a)) − rnd(c) ≥ 0), а в противном случае − она выпала над кривой в площади окаймляющего прямоугольника.

Лабораторная работа № 8. Одноключевая система шифрования Диффи и Хеллмана.

Построить систему шифрования Диффи и Хеллмана для a = (количество согласных букв в фамилии студента), $p \geq$ (количество всех букв в фамилии). Подобрать a и p самостоятельно методом проб и ошибок, выбрать два секретных числа X_i и X_j и для связи пользователей сети i и j вычислить числа Z_{ij} и Z_{ji}.

В методе Диффи и Хеллман реализована идея использования функций с лазейкой для построения криптосистемы в сети с открытым распределением ключей. Для решения этой задачи они предложили использовать функцию $F(x) = a^x \bmod p$, где p − большое простое число, x − произвольное натуральное число из множества $\{1, 2, ..., (p-1)\}$, a – целое число из множества $\{2, 3, ..., p\}$, для которого выполняется требование, чтобы все степени a от 1 до $(p-1)$ в произвольном порядке по модулю p дали все числа из множества $\{1, 2, ..., (p-1)\}$.

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

$x_i = 3$ (секрет i), $y_i = 3^3 \bmod 7 = 6$.

$x_j = 4$ (секрет j), $y_j = 3^4 \bmod 7 = 4$.

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

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

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. Вычисление остатков x при делении целых чисел на модуль y можно выполнять с помощью функции mod(x,y).

Лабораторная работа № 9. Двухключевая система RSA.

Построить двухключевую систему с использованием алгоритма RSA и выполнить в ней операцию шифрования и дешифрования трех первых букв фамилии студента (при количестве букв меньше 3, недостающие буквы берутся из имени). Пара простых чисел P и Q выбирается из диапазона ближайших к количеству букв в фамилии и имени студента.

Например, Петров Владимир, P (5 или 7), Q (7 или 11). Методом испытаний подбирается также ближайшая пара чисел E и D.

В нашем случае это могут быть $P = 5$, $Q = 7$.

В случае неудачных сочетаний из названного диапазона берутся рядом другие ближайшие простые числа, например, $P = 5$, $Q = 13$.

В системе RSA каждый пользователь имеет свой ключ шифрования. Ключи дешифрования известны всем, а шифрующий ключ держится в секрете. Криптографические системы типа RSA подходят для реализации цифровой подписи, применяемой в системах электронных платежей и при передаче сообщений с помощью устройств телесвязи.

К недостаткам системы RSA и аналогичных ей относят ее существенно более низкое быстродействие и потребность в более длинных ключах. Наиболее эффективные реализации RSA характеризуются скоростью шифрования порядка нескольких тысяч бит в секунду. Тогда как аналогичные реализации более простых систем шифруют несколько миллионов бит в секунду. В связи с этим наиболее целесообразным применением RSA считается организация обмена секретными ключами, необходимыми для обеспечения безопасности в сетях связи.

Основная проблема для системы RSA – генерация соответствующей пары ключей. Для генерации используется следующая процедура:

1. Выбрать 2 простых числа P и Q;
2. Найти произведение $N = PQ$ и число $L = (P - 1)(Q - 1)$;
3. Выбрать случайное число D такое, что оно должно быть взаимно простым с числом L (числа называются взаимно простыми, если они не имеют общего делителя);
4. Определяют другое число E такое, что $(ED) \bmod L = 1$;
5. Как только все числа найдены, мы имеем: секретный ключ – E; открытый ключ – пара чисел D и N;

Тогда при шифровании сообщения его разбивают на блоки M. В результате шифрования для каждого блока M получим число

$C = (M^E) \bmod N$.

При дешифрации получаем:

$M^* = (C^D) \bmod N$.

Рассмотрим это на примере алфавита из букв {А, О, Я} = {1, 2, 3} для передачи текста «ОЛЯ» (или 2, 1, 3). Цифровые обозначения букв или блоков обязательны, так как метод основывается на обработке натуральных чисел.

1. Выберем $P = 3$ и $Q = 11$;
2. Найдем $N = PQ$, $N = 33$; $L = (P - 1)(Q - 1)$; L=20;
3. Выберем D, взаимно простое с L: D = 3;
4. Выберем E такое, что $(ED) \bmod L = 1$: $E = 7$, действительно,

$(7 \cdot 3) \bmod 20 = 1$;

5. Тогда открытый ключ $\left.\begin{matrix} D = 3 \\ N = 33 \end{matrix}\right\}$ секретный $E = 7$.

Производим шифрацию своим закрытым ключом 7:

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

$$C1 = (M1^9) \bmod N: \; C2 = (1^7) \bmod 33 = 1,$$

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

Зашифрованный текст получается (29,1,9).

Расшифровка текста открытым ключом 3 и33:

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

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

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

В результате мы получили исходный текст.

Остается только добавить, что для получения достаточно стойкой шифровки необходимо брать очень большие простые числа.

Выполнение соотношения $(ED) \bmod L = 1$ позволяет использовать этот факт для проверки подлинности подписи без знания секретного ключа E с помощью аппарата ХЭШ-функций.

В практической работе необходимо идентифицировать автора электронного документа и предприятие не по особенностям подписи и печати (например, по образцам подписей и печатей в банковской карточке клиента), а по наличию у него электронного ключа для подписывания документов. В этом случае конкретное число-подпись под данным документом в фиксированное время, может сделать только законный обладатель ключа (E).

Процедура электронной подписи включает в себя два этапа: первый – подписывание (вычисление параметров подписи, зависящих от текста конкретного документа, один из которых (E) хранится в секрете); второй – проверка получателем с помощью несекретных параметров (D, N) подлинности сообщения (подписи)

Сообщение шифруется по алгоритму RSA, где E подбирается и известно только отправителю, а D, N знает и получатель. Получатель должен иметь возможность с помощью открытого ключа проверить подлинность сообщения. Для этой цели в сообщение добавляется еще одно число, которое является результатом вычисления хэш-функции $h(T)$, зависящей от текста T.

Лабораторная работа № 10. Обеспечение подлинности сообщений.

Подобрать хэш-функции $h(T)$ или применить из теоретического материала и используя секретный ключ E из предыдущего задания и зашифрованное сообщение (три буквы) вычислить $m = h(T)$ и $S = (m^E) \bmod N$. Далее, пользуясь открытым ключом D вычислить m из соотношения $(S^D) = m \bmod N$ и убедиться в его совпадении с m владельца секретного ключа. В конечном виде передаваемое $m < N$.

К хэш-функции предъявляется ряд требований:

– невозможность (или за очень длительное время) найти по значению $h(T)$ само T (т.е. требуется построить практически необратимую функцию);

– для заданного T нельзя найти такое T', чтобы $h(T) = h(T')$;

– вообще нельзя найти пару различных слов T и T' такую, что $h(T) = h(T')$;

– сообщение T (например, текст договора, платежного поручения и т.п.) по заданной функции сжимается в целое число $m = h(T)$, причем $1 < h(T) < N$.

Число m позволяет с помощью открытого ключа констатировать подлинность документа.

С этой целью автор документа с помощью своего секретного ключа E получает второй параметр подписи $S = (m^E) \bmod N$. Параметры m и S вставляются в текст сообщения на место подписи и печати. Все сообщение по телекоммуникационным каналам передается получателю. Он проверяет правильность цифровых параметров m и s, исходя из знания функции $h(T)$, полученного символьного объема зашифрованного сообщения ($T1$) и «лазейки» для вычисления $h(T)$, $h(T1)$.

Проверка параметра S производится путем идентификации условия:
$(S^D) = m \bmod N$.

Математически доказано, что результат проверки m и S будет положительным в том случае, когда в их формировании использовался секретный ключ E, соответствующий открытому ключу D. Вероятность расшифровки секретного ключа E по открытым параметрам S, m, D и N считается ничтожно малой из-за затрат времени на решение задачи взлома системы велико по сравнению со временем полезного действия сообщения.

Покажем в упрощенном варианте проверку подписи сообщения T = (ОЛЯ), с дополнением его параметрами m и S. В качестве функции хеширования $h(T)$ возьмем произведение П сумм из двух элементов для каждой буквы: ее цифрового кода и простого числа p из ряда (2, 3, 5, 7...) с позиции, соответствующей расположению буквы в тексте. Например, для слова ОЛЯ, коды букв О = 2, Л = 1, Я = 3, а их позиции в слове 1, 2, 3, тогда T = П mod 33 = (2 + 2)(1 + 3)(3 + 5) mod 33 = 29. Можно убедиться, что эта функция удовлетворяет требованиям к ней по крайней мере для любого сообщения из 3-х разных букв. Это обнаруживается при вычислении всех возможных шести разнобуквенных сообщений T: (ОЛЯ) – 29, (ОЯЛ) – 12, (ЛОЯ) – 21, (ЛЯО) – 27, (ЯОЛ) – 18, (ЯЛО) – 8, т.е. нет равных $h(T)$.

Следует заметить, что с ростом длины сообщения может оказаться, что такая функция не удовлетворяет требованию, когда два сообщения разной длины имеют одинаковое значение, т.е. $h(T) = h(T')$. Чтобы избежать такого явления можно разбивать сообщения на блоки заранее ограниченной длины. Поэтому выбор хорошей функции $h(T)$ является очень трудной задачей и ее решением занимаются специалисты, которые разрабатывают стандарты шифрования. В описанном нами примере ограничимся лишь демонстрацией процедуры признания подписи.

Текст T, который получает партнер позволяет проверить подлинность подписи по параметрам m, S и D = 3 (известно как открытая часть ключа), S и m приходят с текстом отправителя ($S = m^E \bmod N = 29^7 \bmod 33 = 17$; $m = 17$, т.е. меньше 33). Проверка подлинности сообщения: $S^D = m \bmod N$; $17^3 \bmod 33 = 29$, т.е. результат проверки положителен и 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 П34, 101.25-2008 для электронной подписи, в котором алгоритм RSA один из трех рекомендуемых для применения.

Лабораторная работа № 11. Методы борьбы с контрафактной продукцией на основе БД.

Проверить штрих-код любого предприятия по описанному далее алгоритму и отметить его полезные функции в автоматизированных системах управления, логистике, создании баз данных. Предложить скрытую часть кода(пломбируемую или защищенную краской), например, какую-то функцию от серийного номера изделия, гаммированный передаваемый в двоичной форме серийный номер изделия. Гаммирование можно выполнить с помощью части таблицы случайных двоичных чисел: 01011000101110101011100010101110. В этом случае передающая и принимающая стороны знают начальную строку и позицию цифры в ней, с которой начинать отсчет. Пусть требуется гаммировать блок 10101111010011100110 с помощью таблицы с пятой цифры в ней. Тогда, например, запись 7624 переводится в двоично-десятичный код каждой из цифр: 0111011000100100 и получим гаммированный блок:1000001011000010.

Особенности использования штрих-кодов в борьбе с контрафактной продукцией (подделки от имени предприятия –производителя) можно использовать также особого рода протоколы.

В основе борьбы от подделок лежат скрытые и открытые маркировки производителя. Обычно такие маркировки состоят из двух и боле 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). Всей этой информации достаточно, чтобы обратиться к производителю. Кодирование скрытой части индивидуально для каждого изделия и представляет собой набор случайных генерируемых программой цифр и машинно-читаемых знаков. Скрытую часть невозможно прочитать без нарушения целостности изделия или упаковки или без удаления стирающегося слоя краски (подобно защите лотерейных билетов спортлото).

Производитель создаёт уникальную электронную базу данных с неповторяющимися идентификаторами тождественными скрытой маркировке на изделии. При первом запросе покупателя на предприятие для подтверждения подлинности кода изделия, этот код удаляется в другую базу данных, т. е. вторая и последующие авторизации невозможны, а при несоответствии первого предъявления уникального кода или его повторном предъявлении, можно приступить к борьбе с авторами подделок.

Эта же система исключает и продажу неучтённой продукции, выпущенной на самом предприятии. Кроме того, исключаются и грубые подделки штрих-кода. Существует простой алгоритм проверки кода EAN-13.

Алгоритм проверки кода:

1. Сложить цифры, стоящие на четных местах *Sr*.
2. Получить $S1 = Sr \times 3$.
3. Сложить цифры на нечетных местах *S*н (без контрольной).
4. Получить $S = S1 + Sн$.
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. $Sr = 5 + 0 + 2 + 0 + 1 + 0 = 8$.
2. $S1 = 8 \times 3 = 24$.
3. $Sн = 8 + 9 + 7 + 1 + 0 + 2 = 27$.
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. $Sн = 8 + 9 + 5 + 1 + 0 + 2 = 25$.
4. $S = 24 + 25 = 79$.
5. $t = 9$.
6. $P = 10 - 9 = 1$, т.е. контрольная цифра говорит о чувствительности кода (она не совпала).

Важность этого типа кода определяется еще и тем, что ряд банков Республики Беларусь используют для счетов предприятий 13-разрядные коды типа EAN-13. Ошибка персонала предприятия при заполнении документов, тогда легко вскрывается по контрольному разряду и банк не выполняет перевод денег.

Идея скрыть некоторые элементы передаваемой информации или сам факт ее передачи и хранения оказалась плодотворной в создании сложных многоступенчатых систем защиты информации.

Многоступенчатость защиты информации выражается как в 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 – псевдослучайный последовательности ($D'_г$). Процесс шифрования тогда содержит процедуру генерации гаммы шифра и ее наложении 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 $D'_о$ одинаковый длины (чаще всего 64 бита). Каждый открытый блок $D'_о$ путем сложения с гаммой преобразуется в гаммированный блок $D'_ш$ аналогичной длины готовый для шифрования. Уравнение гаммирования для каждого блока i из набора k блоков, тогда записывается так $D'_ш = D'_о \oplus D'_г$. Обратная процедура на приемном конце сводится к повторной генерации гаммы по известному для принимающего закону. Тогда расшифрованный текст легко получается по формуле: $D'_о = D'_ш \oplus D'_г$. Такой метод позволяет изменять гамму для каждого шифруемого блока случайным образом за счет генерации псевдослучайных чисел.

Для компьютерных продуктов характерна защита интеллектуальной собственности в форме ноу-хау, так как эти объекты пока не патентуются.

Частью системы правовой охраны промышленной собственности в Республике Беларусь является защита от недобросовестной конкуренции.

«Недобросовестная конкуренция – любые направленные на приобретение преимуществ в предпринимательской деятельности действия хозяйствующих субъектов, которые противоречат требованиям добросовестности и разумности и могут причинить или причинили убытки другим хозяйствующим субъектам-конкурентам либо нанести ущерб их деловой репутации».

Любые действия, направленные на ограничение или устранение конкуренции путем нарушения прав других хозяйствующих субъектов на свободную конкуренцию, а также нарушающие права и законные интересы потребителей, не допускаются (незаконное использование фирменного наименования, товарного знака, копирование внешнего вида товара другого хозяйствующего субъекта, введение в гражданский оборот товаров другого хозяйствующего субъекта с использованием собственных средств индивидуализации товара, неправомерные утверждения при осуществлении предпринимательской деятельности, способные дискредитировать хозяйствующий субъект, товары или предпринимательскую деятельность конкурента).

Объекты промышленной собственности охватывают результаты интеллектуальной деятельности, имеющие производственную направленность. К объектам промышленной собственности относят изобретения, полезные модели, промышленные образцы, селекционные достижения, топологии интегральных микросхем, нераскрытую информацию.

Заключительные общие рекомендации. При необходимости использования результатов в конкретных целях (в выполнении курсовой, дипломной, магистерской или другой работы, носящей исследовательский характер) полезно прибегнуть к дополнительному изучению литературы из списка источников [4, 13, 36, 37]. Смысл их использования состоит в необходимости системной увязки вопросов защиты информации и интеллектуальной собственности с экономических и юридических позиций. Это необходимо при подготовке объектов для выхода на международные рынки, так как здесь решающим фактором становятся экономические и правовые вопросы: первые – используется для оценки затрат на рыночное продвижение продукта, а вторые – для его защищенности (патентование объектов промышленной собственности, регистра торые ция товарного знака, патентная чистота объекта и т.д.). Вторым важным объектом являются различные дистанционные операции в сетях ЭВМ (электронная коммерция и платежи между субъектами хозяйствования, выполнение дистанционных совместных работ и заказов.

8.4. Пример заданий для письменной работы

Учреждение образования
«Брестский государственный технический университет

Письменная работа студента (Ф.И.О.)_____
группы ИИ-11 факультета ЭИС по дисциплине:
«Традиционные и интеллектуальные информационные технологии».

Задание № 1. Описать схему итерационного алгоритма вычисления $y = \sqrt{x}$ на базе метода, использующего формулы $y_i = 0.5(y_{i-1} + x / y_{i-1})$ с критерием по точности вычислений $|y_i - y_{i-1}| \le \varepsilon = 0{,}1$, где x = И × rnd(И) + Ф rnd(Ф) – данное для проверки алгоритма, для доказательства его устойчивости к сбоям сделать ошибку на шаге 3 и сравнить результаты, полученной без сбоя с результатом со сбоем, при вычислениях сбой сделать в целой части y на 1. (И.Ф – количество букв в имени и фамилии студента).

Задание № 2. Методом ветвей и границ решить задачу коммивояжера для 5 городов, расстояния a_{ij} между которыми заданы матрицей $\|a_{ij}\|$, где $a_{ii} = 0$, $a_{ij} = 1 + \text{ceil}(\text{rnd}(\Phi))$.

В ответе указать найденный маршрут в виде вектора ($x1, x2, x3, x4, x5, x1$) и длину кольцевого маршрута L, где xi – номер города

Задание № 3. Решить трансцендентное уравнение $a_3x^3 + a_2x^2 + a_1x + a_0 - k\sin(tx) = 0$ на заданном отрезке [p, r] графическим методом в системе Mathcad с точностью 0,1.

Перечислить все действительные корни и выполнить проверку полученных результатов с помощью процедуры root. Исходные данные: p = И rnd(И); $r = p$ + Ф rnd (Ф); $a_3 = 1$; $a_2 = 1$ + rnd(2); $a_1 = 2$ rnd (3); $a_0 = -$И, где И, Ф – количество букв в имени и фамилии.

Сделать вывод о разнице результатов, полученных по двум методам.

Задание № 4. Построить систему шифрования Диффи и Хелмана для a = 1 + + ceil(rnd(И / 2)); p – ближайшее простое число к сумме (a + Ф), удовлетворяющее правилу построения системы. Выбрать два секретных числа x_i и x_j и для связи пользователя i и j вычислить числа z_{ij} и z_{ji}.

Задание № 5. Методом Монте-Карло вычислить интеграл $\int_{\Phi*\text{rnd}(1)}^{\Phi} \sqrt[4]{(x^3 - x^2 + u)/(x^2 + \Phi)}\,dx$ путем вбрасывания $N_1 = 1000 + $ И и $N_2 = 5000 + \Phi$ точек. Этот же интеграл найти численным методом, используя систему Mathcad и сранить оба результата. Оценить величину ошибки моделирования в % для этих двух вычислений и сделать выводы (Ф – количество букв в фамилии, И – количество букв в имени). Записать в системе Mathcad ход вычисления $n = T_1$ (количество удачных попыток в испытаниях для N_1 и N_2.

Задание № 6. Построить двухключевую систему шифрования с использованием алгоритма RSA и выполнить в ней операцию шифрования и дешифрования осмысленного слова из четырех букв в алфавите {А, Б, В, Л, О. Я}. P и Q выбрать из пар (3, 11), (5, 7), (5, 11), (5, 13), (5, 19) – пара определяется по номеру, равном величине ceil(rnd(4)).

Заведующий кафедрой_____В.А.Головко Преподаватель_____Л.П.Матюшков

Дата утверждения: . .2014, протокол № заседания кафедры.

ЗАКЛЮЧЕНИЕ

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

– стремительное нарастание количества «умной» бытовой техники с программным управлением и возможностями принятия отдельных решений без непосредственных контактов с хозяевами;

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

– развитие систем совместного взаимодействия с мобильными средствами цифровой связи;

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

– управление из космоса движением на Земле;

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

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

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

– создание виртуальных предприятий.

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

– внедрение дистанционных систем обучения с индивидуальный скоростью подготовки для каждого учащегося;

– использование баз знаний во многих областях деятельности на уровне «экспертных» систем, когда их пользователь не будет иметь очень высокой квалификации в данной области, но обеспечит получение хороших общих решений в рамках поставленных ему задач;

– применение в преподавании вспомогательных систем, ускоряющих процесс обучения и позволяющих сосредоточиться учащимся на главных элементах (например, автоматическая запись исполняемой музыки на ноты, различные подсобные вычисления).

В последние годы усилились разработки по организации прямого поиска и передачи информации в системах из разнородных технических и программных компонентов (GGG-системы). Эти исследования открывают пути к глобальному получению информации в реальном масштабе времени работы пользователя из любого уголка Земли.

Нанотехнологии становятся также одним из важных элементов бурного развития компьютерных технологий: совершенствование компьютерной базы (биокомпьютерные, квантовые и т.п.), прямое биоуправление роботами,создание роботов для синтеза себеподобных объектов и т.д.

Полезность предложенного нами подхода в методическом плане состоит в лёгкой адаптируемости использования материалов для ряда специальностей ,широко применяющих описанные методы.Построение системы обучения включает такие важные моменты:развитие исследовательских навыков во всех практических и лабораторных работах у студентов младших курсов,выдача индивидуальных заданий для каждого студента,возможность проверки результатов другими методами(косвенное подтверждение правильности вполнения задания для студента и преподавателя),рекомендуемая письменная работа по сути является комплексным закрытым тестом по всему учебному курсу.

ЛИТЕРАТУРА

1. Логика. Автоматы. Алгоритмы / М.А. Айзерман [и др.]. – М. : Физматгиз, 1963. – 556 с.

2. Таненбаум, А. Архитектура компьютера: пер. с англ. / А. Таненбаум. – СПб : Питер, 2002. – 704 с.

3. Бибило, П.Н. Основы языка VHDL / П.Н. Бибило. – М. : Солон, 2002. – 224 с.

4. Баричев, С.Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. – М. : Горячая линия, Телеком, 2001. – 120 с.

5. Информационные системы и технологии в экономике / Т.П. Барановская [и др.]. – М. : ФиС, 2003. – 416 с.

6. Витязь, П.А. Основы нанотехнологий и наноматериалов: уч.пособие // П.А. Витязь, Н.А. Свидунович. – Минск : Выш.шк., 2010. – 302 с.

7. Бирюков, Б.В. Машина и творчество: Результаты, проблемы, перспективы / Б.В. Бирюков, И.Б. Гутчин. – М. : Радио и связь. 1982. – 152 с.

8. Бондаенко, В.Ф. MATLAB. Основы программирования, компьютерная математика. Учебный курс / В.Ф. Бондаренко, В.Д. Дубовец. – Минск : Харвест, 2010. – 256 с.

9. Глушков, В.М. Основы безбумажной информатики / В.М. Глушков. – М. : Наука. Ф.М., 1982. – 552 с.

10. Головко, В.А. Нейронные сети: обучение, организация и применение. Кн. 4. Учебное пособие для вузов / В.А. Головко. – М. : ИПРЖР, 2001. – 256 с.

11. Норенков, И.П. Основы автоматизированного проектирования / И.П. Норенков. – Издательство МГТУ имени Н.Э. Баумана, 2000. – 360 с.

12. Татур, М.М. Классификаторы в системах распознавания: прикладные авпекты синтеза и анализа / М.М. Татур, Д.Н. Одинец. – Минск : Бестпринт, 2008. – 165 с.

13. Головко, В.А. Основы защиты информации и управления интеллектуальной собственностью:учебно-методический комплекс / В.А. Головко, Л.П. Матюшков. – Брест : Изд-во УО «БрГТУ», 2011. – 76 с.

14. Головко, В.А. Основы вычислительных систем: методическое пособие // В.А. Головко и [др.]. – Брест : Изд-во УО «БрГТУ», 2013. – 148 с.

15. Дудкин, А.А. Обработка изображений в проектировании и производстве интегральных схем / А.А. Дудкин, Р.Х. Садыхов. – Минск : ОИПИ НАН Беларуси, 2008. – 270 с.

16. Дэо, Н. Комбинаторные алгоритмы / Н. Дэо, Ю. Нивергельт, Э. Рейнгольд // Пер. с англ. – М. : «Мир» 1980. – 476 с.

17. Искусственный интеллект. Кн.1: Системы общения и экспертные системы / Под ред. В. Попова. – М. : Радио и связь, 1990. – 420 с.

18. Карпов, Б. Microsoft Access 2000: справочник / Б. Карпов. – СПб. : Питер, 2001. – 416 с.

19. Кемени, Дж. Кибернетическое моделирование / Дж. Кемени, Дж. Снелл. – М. : «Советсеое радио», 1972. – 192 с.

20. Криницкий, Н.А. Алгоритмы и роботы / Н.А. Криницкий. – М. : Радио и связь, 1983. – 168 с.

21. Программирование и алгоритмические языки / Н.А. Криницкий [и др.]. – М. : Наука, 1979. – 509 с.

22. Комарцова, Л.Г. Нейрокомпьютеры: учебное пособие для вузов / Л.Г. Комарцова, А.В. Максимов – М. : Изд. МГТУ им. Н.Э. Баумана, 2004. – 400 с.

23. Базы данных. Интеллектуальная обработка информации // В.Г.Корнеев [и др.]. – М. : Нолидж 2001. – 496 с.

24. Лисьев, Г.А. Технология поддержки принятия решений // Г.А. Лисьев, И.В. Попова. – М. : изд. Флинта, 2011. – 133 с.

25. Макаров, Е.Г. Инженерные расчеты в Matcad-14 / Е.Г. Макаров. – СПб. : Питер, 2007. – 592 с.

26. Мацукевич, В.В. Основы управления интеллектуальной собственностью. Учебно-методический комплекс: учеб. пособие / В.В. Мацукевич, Л.П. Матюшков. – 2-е изд. – Минск : Выш. шк., 2013. – 224 с.

27. Матюшков, Л.П. Традиционные и интеллектуальные информационные технологии. Методические указания к выполнению практических занятий и лабораторных работ / Л.П. Матюшков, В.А. Головко. – Брест : БрГТУ, 2014. – 30 с.

28. Матюшков, Л.П. Основы машинной математики / Л.П. Матюшков, А.А. Лихтарович. – Минск : Нар. Асвета, 1988. – 240 с.

29. Матюшков, Л.П. Перспективы информационных технологий в развитии дистанционных рабочих мест и коммерческих операций / Л.П. Матюшков // «Вучоныя запіскі БрДУ». – Брест : БрДУ, 2006. – Т.2. – Ч.1. – 107-115 с.

30. Пархоменко, П.П. Основы технической диагностики / П.П. Пархоменко, В.С. Сагомонян. – М. : Энергоиздат, 1981. – 349 с.

31. Пасхин, Е.Н. Автоматизированная система обучения ЭКСТЕРН / Е.Н. Пасхин, А.И. Митин. – М. : Изд-во МГУ, 1985. – 144 с.

32. Попов, Е.П. Роботы и человек / Е.П. Попов, А.С. Ющенко. – М. : Наука, 1984. – 112 с.

33. Попов, Э.В. Общение с ЭВМ на естественном языке / Э.В. Попов. – М. : Наука, 1982.– 360 с.

34. Поспелов, Г.С. Искусственный интеллект – прикладные системы / Г.С. Поспелов, Д.А. Поспелов. – М. : Знание, 1985. – 46 с.

35. Положение о коммерческой тайне (утв. СМ РБ 06.11.1992 №-670).

36. Предварительный государственный стандарт Республики Беларусь СТБ П 34.101.25-2008. Информационные технологии. Стандарт электронной цифровой подписи.

37. Об электронном документе и электронной цифровой подписи. Закон Республики Беларусь (28 декабря, 2009 года, №113-3). – 10 с.

38. Тихонов, А.Н. Вводные лекции по прикладной математике / А.Н. Тихонов, Д.П. Костомаров. – М. : Наука, 1984. – 192 с.

39. Успенский, В.А. Машина Поста / В.А. Успенский. – М. : Наука, 1979. – 95 с.

40. Гэри, М. Вычислительные машины и труднорешаемые задачи: Пер. с англ. / М. Гэри, Д. Джонсон. – М.: Мир, 1982.– 416 с.

41. Якимахо, А.П. Управление объектами интеллектуальной собственности в РБ / А.П. Якимахо. – Минск : Амалфея, 2005. – 472 с.

42. Янсен, Ф. Эпоха инноваций / Ф. Янсен. – М. : ИНФРА-М, 2002. – 308 с.

ГЛОССАРИЙ

Аутентификация	– проверка соответствия пользователя по его особым характеристикам (отпечаток ладони, пальца,части сугубо личной информации и т.п.)
Схема алгоритма	– функционально-ориентированное графическое изображение, с помощью которого, используя текст и специальные символы, описывают последовательность шагов процесса во времени, связи рассматриваемой системы с внешней средой и ветвление процесса.
Дисплей	– устройство, обеспечивающее визуальное представление цифровой, алфавитно-цифровой и (или) графической информации на экране электронно-лучевой трубки, в плазменных панелях, на жидких кристаллах, светодиодах и т. п. в форме, удобной для оператора.
Идентификация	– опознание пользователя на право доступа к системе(пароль и т.п.)
Интеллект	– способность мышления, рационального познания.
Интеллектуальный видеотерминал	– видеотерминал, включающий в свой состав процессор и память, доступную для программирования пользователем.

Интерпретатор	– обслуживающая программа, осуществляющая пооператорную трансляцию и выполнение исходной программы.
Информатика	– отрасль науки, изучающая вопросы, связанные с поиском, сбором, хранением, преобразованием и использованием информации в различных сферах человеческой деятельности.
Информационная технология	– технология, использующая в качестве своего носителя ЭВМ как инструмент управления орудиями производства.
Интеллектуальная информационная технология	– информационная технология, использующая при своей реализации творческие решения или их элементы для управления орудиями производства.
Информация	– сведения, совокупность каких-либо данных, знаний и т. п.
Исчисление	– способ задания множества путем указания исходных элементов и правил вывода для построения новых элементов из исходных и уже построенных.
Итерация	– результат многократного применения какой-либо математической операции.
Ключ	– информация, необходимая для беспрепятственного шифрования и расшифрования текстов.
Команда машинная	– элементарное предписание для ЭВМ, определяющее действия машины в течение некоторого отрезка времени (содержит указание операции, адреса операндов и другие служебные признаки).
Компилятор	– обслуживающая программа, выполняющая трансляцию на машинный язык программы, записанной на исходном языке программирования.
Локальная сеть микро – ЭВМ	– сеть микро-ЭВМ, сосредоточенная на ограниченной территории (в пределах одного или нескольких зданий) и не использующая средств связи общего назначения.
Маркер (курсор)	– специальный знак на экране дисплея для указания определенных позиций или элементов.
Манипулятор	– приспособление для выполнения вспомогательных операций по захватыванию и перемещению предметов с управлением по командам.
Меню	– список альтернатив, появляющихся на экране дисплея и предлагаемых для выбора пользователю.
Моделирование	– исследование объектов на их моделях.
Модель	– описание объекта (или системы), используемое в качестве его заменителя.
Нейрокомпьютер	– вычислительная система, состоящая из множества нейронных элементов и базирующаяся на нейросетевых принципах функционирования.
Нейронная сеть	– совокупность нейронных элементов и связей между ними. Лежит в основе организации как биологических объектов, так и искусственных вычислительных систем.
Обучающая машина	– устройство, предназначенное для реализации обучающих программ (учебного материала, в котором описываются подлежащие усвоению знания, умения и навыки, а также способы их формирования).
Оператор	– допустимая в языке программирования синтаксическая конструкция, отражающая определенное действие в программе (присвоение значения, передачу управления и т. п.).
Операционная система	– комплекс взаимосвязанных управляющих и обслуживающих программ, обеспечивающих автоматическое управление вычислительными процессами и ресурсами ЭВМ при решении задач.
Параметр	– величина, входящая в формулы и выражения, значение которой является постоянным в пределах одной задачи.
Персональный компьютер	– небольшая по размерам и стоимости универсальная микро-ЭВМ, предназначенная для индивидуального пользователя.

Подпрограмма	– часть программы, допускающая многократное обращение к ней из различных точек программы.
Пользователь	– лицо, использующее данное вычислительное устройство для выполнения необходимых ему работ.
Программа	– последовательность инструкций, реализующих алгоритм. Программы обычно могут быть написаны: а) в машинном коде, который непосредственно выполняется процессором; б) на языке типа ассемблер; в) на языке высокого уровня.
Промышленный робот	– программируемый многофункциональный манипулятор, предназначенный для перемещения материалов, деталей, инструмента или специализированных устройств по переменным программируемым траекториям для выполнения широкого круга задач.
Процедура	– порядок выполнения ряда последовательных действий.
Процедура рекурсивная	– процедура, в описании которой содержится обращение к ней самой.
Распределенная система	– система обработки данных, в которой отдельные функции обработки выполняются независимыми устройствами.
Речевой ввод	– ввод информации в ЭВМ с помощью голоса. В этом случае ЭВМ реализует процесс распознавания речи (либо отдельных слов, либо фраз).
Робот	– способная действовать целенаправленная система управления и переработки информации, оборудованная датчиками восприятия информации о внешней среде и исполнительными механизмами.
Световое перо	– устройство, используемое для указания элемента изображения на экране графического дисплея.
Семантика алгоритмического языка	– толкование содержательного смысла единиц алгоритмического языка.
Сеть ЭВМ	– система соединенных между собой и обменивающихся информацией ЭВМ.
Синтаксис алгоритмического языка	– совокупность правил однозначного описания содержания алгоритмов в алгоритмическом языке.
Система	– совокупность частей, связанных общей функцией.
Система команд	– полный набор всех инструкций, допустимых в машинном языке данной ЭВМ.
Системное программное обеспечение	– набор обслуживающих программ, предназначенных для трансляции, редактирования, отладки и загрузки прикладных программ пользователя.
Структура данных	– упорядоченное множество элементов данных (вектор, матрица и др.).
Технология	– наука о способах воздействия на сырье, материалы, полуфабрикаты и информацию соответствующими орудиями производства.
Транслятор	– программа, предназначенная для перевода описаний алгоритмов с одного формального языка на другой (обычно с алгоритмического на машинный).
Файл	– последовательность размещаемых на внешних ЗУ записей, рассматриваемая в процессе пересылки и обработки как единое целое.
Шифрование	– процесс преобразования исходного текста, который носит также название открытого текста, в шифрованный текст.
Численный метод	– метод приближенного или точного решения математических задач, основанный на построении конечной последовательности действий над конечным множеством чисел.
Электронная цифровая подпись	– присоединяемая к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения.
Язык алгоритмический	– средство точного формулирования вычислительных процессов для их последующей реализации на автоматических вычислительных машинах.

УЧЕБНОЕ ИЗДАНИЕ

Головко Владимир Адамович
Дудкин Александр Арсентьевич
Матюшков Леонид Петрович

ОСНОВЫ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

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

Текст печатается в авторской редакции

Ответственный за выпуск: Матюшков Л.П.
Редактор: Боровикова Е.А.
Компьютерная вёрстка: Боровикова Е.А.

ISBN 978-985-493-324-5

Издательство БрГТУ.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/235 от 24.03.2014 г.
Подписано к печати 17.03.2015 г. Формат 60×84 1/16.
Бумага «Снегурочка». Гарнитура «Times New Roman».
Усл. п. л. 10,5. Уч.-изд. л. 11,25. Тираж 70 экз.
Заказ № 199. Отпечатано на ризографе учреждения образования «Брестский государственный технический университет». 224017, Брест, ул. Московская, 267.