Министерство просвещения Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Южно-Уральский государственный
гуманитарно-педагогический университет»

Г.Б. ПОДНЕБЕСОВА

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Учебное пособие

Челябинск
2022

УДК 001.8(021)
ББК 73я73
П 44

Поднебесова, Г.Б. Теоретические основы информатики: учебное пособие / Г.Б. Поднебесова. – Челябинск: Изд-во Южно-Урал. гос. гуман.-пед. ун-та, 2022. – 196 с. – Текст: непосредственный.

ISBN 978-5-907611-09-2

Учебное пособие содержит материал для изучения курсов «Теоретические основы информатики» и «Теория информации, данные, знания». Пособие предназначено для организации аудиторной и самостоятельной работы студентов, обучающихся по направлениям «Педагогическое образование» и «Информационные системы и технологии».

Учебное пособие адресовано преподавателям и учителям, которым интересна данная предметная область.

Рецензенты:
С.А. Загребина, д-р физ.-мат. наук, профессор ЮУрГУ
А.А. Рузаков, канд. пед. наук, доцент ЮУрГГПУ

ISBN 978-5-907611-09-2

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Учебный курс вводит студентов в современные проблемы теоретической информатики. Основной акцент в курсе делается на методологические аспекты и математический аппарат информатики, составляющие ядро широкого спектра научно-технических и социально-экономических информационных технологий, которые реально используются современным мировым профессиональным сообществом в теоретических исследованиях и практической деятельности.

Учебные дисциплины «Теоретические основы информатики», «Теория информации, данные, знания» базируются на материале предшествующих ей дисциплин «Математика» (Математический анализ, Алгебра и теория чисел), курсов «Абстрактная и компьютерная алгебра» и «Теории алгоритмов».

В результате изучения дисциплины студент должен:

– иметь представление об общих проблемах и задачах теоретической информатики;

– иметь представление об основных принципах и этапах информационных процессов;

– знать наиболее широко используемые классы информационных моделей и основные математические методы получения, хранения, обработки, передачи и использования информации;

– уметь применять математический аппарат анализа и синтеза информационных систем;

– уметь применять методы программирования и навыки работы с математическими пакетами для решения практических задач хранения и обработки информации.

Изучение дисциплины предусмотрено вариативной частью блока обязательных дисциплин ОПОП ФГОС ВО.

Компетенции, формируемые в результате освоения дисциплины:

✓ ОПК-8 способен осуществлять педагогическую деятельность на основе специальных научных знаний:

ОПК.8.1 Знать историю, теорию, закономерности и принципы построения научного знания для осуществления педагогической деятельности.

ОПК.8.2 Уметь проектировать и осуществлять педагогическую деятельность с опорой на специальные научные знания.

ОПК.8.3 Владеть технологиями осуществления педагогической деятельности на основе научных знаний.

✓ ПК-1 способен осваивать и использовать базовые научно-теоретические знания и практические умения по преподаваемому предмету в профессиональной деятельности:

ПК.1.1 Знает содержание, особенности и современное состояние, понятия и категории, тенденции развития соответствующей профилю научной (предметной) области; закономерности, определяющие место соответствующей науки в общей картине мира; принципы проектирования и реализации общего и (или) дополнительного образования по предмету в соответствии с профилем обучения.

ПК.1.2 Умеет применять базовые научно-теоретические знания по предмету и методы исследования в предметной области; осуществляет отбор содержания, методов и технологий обучения предмету (предметной области) в различных формах организации образовательного процесса.

ПК.1.3 Владеет практическими навыками в предметной области, методами базовых научно-теоретических представлений для решения профессиональных задач.

ГЛАВА 1. ТЕОРИЯ ИНФОРМАЦИИ

1.1. Понятие об информации и её природе

Демонстрировать информационные процессы лучше всего в управлении. Информация передаётся от управляющего органа к управляемому объекту и от последнего к управляемому органу, то есть при осуществлении управления имеют место процессы сбора, сортировки, преобразования, передачи, хранения информации.

Термин «информация» (*information* – разъяснение, изложение) широко используется и в науке и обычной жизни, точное его определение оказывается весьма затруднительным. Информацию мы понимаем, как сообщение о состоянии и свойствах объекта, явления, процесса. Но что собой представляют эти сведения? Это информация об интересующих нас объектах.

Выделяют 3 категории свойств информации:

1) атрибутивные – это свойства, без которых информация не может существовать;

2) прагматические – это свойства, которые показывают полезность информации для практики;

3) динамические – это свойства, которые показывают изменение информации во времени.

К атрибутивным свойствам относятся такие свойства, как языковая природа, неотрывность от материального носителя и независимость от него.

К прагматическим свойствам – наличие смысла, новизна, ценность (полезность).

К динамическим свойствам – свойства роста, старения, рассеивания.

Информация – основное понятие кибернетики. Н. Винер дал следующее определение информации: «Информация есть информация, а не материя и не энергия», тем самым отделив понятие информации от материальных объектов.

Попробуем дать определение термина информация с помощью рисунка 1.1:

Рис. 1.1. Пример управляющей и управляемой системы

Исходя из общих положений материалистического понимания мира, совершенно очевидно, что в качестве источника информации может выступать лишь некоторый материальный объект. Этот объект (1) может излучать сигналы различного вида (электромагнитные, световые, звуковые, и так далее), которые приходят к приёмнику (2). Однако приемник должен быть способен использовать содержимое сигналов для целей управления. Определим, таким образом, информацию как содержание сигналов, поступающих в кибернетическую систему из окружающей среды, которое может быть (раньше или позже) использовано системой для целей управления. Это определение подчёркивает основные черты информации.

1. Понятие «информация» имеет смысл лишь в сочетании с понятием «управление». Информационные процес-

сы имеют место лишь в кибернетических системах, осуществляющих функции целенаправленного управления.

2. Информация неразрывно связана с сигналами как переносчиками информации и реализуется в результате взаимодействия двух систем – источника и кибернетического приёмника информации (при отсутствии такого взаимодействия само понятие информации становится беспредметным).

Сказанным подчёркивается неприменимость понятия «информация» к неорганическому миру, в котором происходят процессы, объяснимые определёнными физическими закономерностями. Живая природа обладает способностью к передаче и получению информации (от клетки к клетке, а также для целей управления (цветы)). Возникает вопрос о правомерности применения термина «информация» к процессам в неорганических кибернетических системах и электронных машинах. Эти устройства и системы нельзя рассматривать как независимые от человека. Даже в условиях автономности работы, будучи спроектированы, построены и запрограммированы человеком, они выполняют функции, необходимые для достижения цели, поставленной человеком.

Таким образом, мы имеем полное право говорить об информационных процессах в кибернетических машинах и системах, об обмене информацией между центрами её переработки в сети вычислительных центров и так далее, полагая, что это не автономные от человека неорганические системы, а устройства, выполняющие и продолжающие функции целенаправленного управления.

Однако следует рассмотреть вопрос о том, является ли информация материальной или идеальной категори-

ей. Источниками информации могут быть только материальные объекты, и её переносчиками могут служить сигналы. Информация – не материя, а свойство организованной материи (нет такого свойства, как масса, она не подчиняется законам сохранения массы и энергии). Сама информация не является материальной категорией, а является свойством организованной материи.

1.1.1. Общие представления о канале связи

Понятие «информация» неразрывно связано с динамическими процессами взаимодействия её источника и приёмника, которое может быть реализовано лишь при наличии между ними системы связи, которая называется каналом связи.

Рис. 1.2. Общая схема передачи информации по каналу связи

Сообщение, вырабатываемое источником и подлежащее передаче (человеческая речь, музыкальное произведение, письменный текст, изображение и так далее), нужно предварительно превратить в сигнал, вид которого удобен для передачи по каналу связи. Чаще всего в современных системах связи таким видом являются колебания тока (напряжение) при передаче по проводам или электромагнитные поля при радиопередаче. Это превращение осуществляется в передатчиках различных типов (звуковая передача – микрофон, телеграфная передача – телеграфный ключ, передача изображения –

телевизионная трубка). Кроме того, передатчики включают, как правило, специальные генераторы электрических колебаний, усилители, устройства модуляции и другие. С выхода передатчика сигнал поступает в линию связи (воздушные провода, радиолинию) и распространяется по ней до приёмника. В приёмнике происходит обратное преобразование, то есть приёмник должен обеспечить восстановление из поступившего на его вход сигнала электрической природы человеческую речь, музыку, буквы текста.

Передатчик, линия и приёмник в совокупности образуют канал связи (имеет место ослабления или затухания сигналов за счёт различных помех: электрических разрядов атмосферного электричества, индустриальных помех).

Описанный процесс передачи сообщений по каналу связи можно записать с помощью математической символики. Любое сообщение, которое содержит определённую последовательность букв, цифр, элементов изображения можно представить в виде множества этих элементов $\{V1\}$, поступивших на вход передатчика. На выходе передатчика получается множество сигналов $\{U1\} = A[\{V1\}]$, где A – некоторый оператор, описывающий преобразование множества $\{V1\}$ в $\{U1\}$. В конце линии связи (на входе приёмника) сигнал можно представить в виде множества $\{U2\} = B[\{U1\}] + \{W\}$, где B – оператор, описывающий преобразование сигнала при распространении его по линии связи (искажение и затухание), а $\{W\}$ – множество помех различной природы, примешивающееся к сигналу при его распространении. После преобразования в приёмнике на выходе его получаем сообщение

$\{V2\} = C[\{U2\}] = C(B[\{U1\}] + \{W\})$, где С – оператор преобразования в приёмнике, восстанавливающий сообщение из сигнала, поступившего с линии. Помехи могут быть сведены к минимуму путём тщательного настраивания аппаратуры, кроме случайных помех.

Естественно, что в результате всех перечисленных преобразований, затуханий, помех, оказывается, что $\{V2\} \neq \{V1\}$, то есть полученное сообщение отличается от отправленного сообщения (голос, шрифт, изображение). Условием передачи информации по каналу связи без искажений будет, очевидно, соблюдение равенства $I[\{V1\}] = I[\{V2\}]$, где I – смысловое содержание информации.

Ещё одной характеристикой, свойственной любому каналу связи, является задержка сообщений на некоторую величину τ. Причиной её является конечная скорость распространения сигналов в любой линии связи. Эта скорость равна $3 \cdot 10^5$ км/с в радиолиниях, $2 \cdot 10^5$ м/с в проводных линиях связи, около 0,35 км/с – при распространении звука в атмосфере. Для почтовых каналов связи – 100, 10, единиц км/ч. Это может являться причиной искажений (фазовые искажения). Всё это можно отнести и к другим каналам связи: акустическим, оптическим, почтовым и другим.

К важнейшим проблемам, исследуемым и решаемым теорией информации, относится проблема наиболее эффективного использования каналов связи (проблема эффективного использования каналов связи (скорость) и проблема надёжности).

Объём сигнала и ёмкость сигнала связи

Каналы связи предназначаются для транспортирования информации от источника сообщений к их полу-

чателю. К свойствам сигнала, характеризующим условия его движения по каналу, относятся такие свойства, как интенсивность (мощность) и частота (диапазон частот сигнала). Этими параметрами определяются технические требования к каналу связи – его способность пропускать сигнал такой мощности и частоты.

В то же время смысл сообщения не играет никакой роли с точки зрения техники передачи.

Важнейшее значение имеет интенсивность сигнала, выражаемая количественно мощностью. Однако новейшая техника усиления сигналов позволяет увеличивать их мощность в любое число раз. Успешно осуществляется усиление до заданной величины даже слабых сигналов, таких, как сигнал, поступающий от космических кораблей, летящих к Венере, Марсу, Юпитеру. Но если одновременно с полезным сигналом к приёмнику поступят помехи, то в процессе усиления будет пропорционально возрастать и интенсивность помех. Поэтому свойства сигнала определяются не абсолютной его величиной, а превышением уровня сигнала над уровнем помех. Количественным мерилом интенсивности сигнала принимают отношение мощности сигнала $P_{\text{с}}$ автоматизированной системы управления к мощности действующей в канале связи помехи $P_{\text{п}}$. При этом, учитывая огромный диапазон мощности сигналов, пользуются не абсолютной величиной этого отношения, а его логарифмом. Таким образом, логарифмическая мера $H_{\text{с}}$ превышения сигнала над помехой, то есть относительный логарифмический уровень сигнала над помехой, будет иметь вид

$$H_c = \log \frac{P_{\text{с}}}{P_{\text{п}}} = \log P_{\text{с}} - \log P_{\text{п}} \ . \qquad (1.1)$$

Следующим параметром сигнала является ширина его полосы или частотного диапазона F_c, представляющий разность между частотами, максимальной – $f_{макс}$ и минимальной – $f_{мин}$, которые содержатся в спектре сигнала:

$$F_c = f_{макс} - f_{мин} \ . \qquad (1.2)$$

Сигнал также характеризуется длительностью T_c, равной разности между окончанием сигнала $t_к$ и временем его начала $t_н$:

$$T_c = t_к - t_н \ . \qquad (1.3)$$

Все перечисленные характеристики сигнала можно представить в виде отрезков прямых, выраженных в определённом масштабе, и отложить эти отрезки параллельно трём взаимно перпендикулярным осям координат: оси относительных уровней H, оси частот F и оси времени T, то можно изобразить сигнал геометрически, в виде параллелепипеда с рёбрами H_c, F_c, T_c.

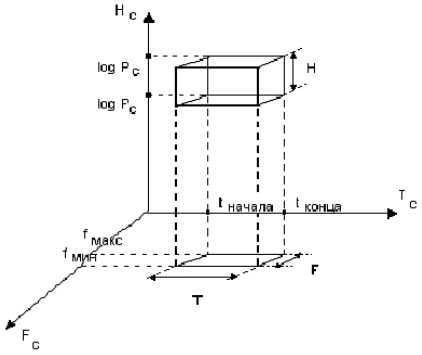


Рис. 1.3. Геометрическое изображение сигнала

Аналогичными параметрами можно охарактеризовать и канал связи.

Обозначим через $H_к$ допустимый диапазон изменения мощности в канале, $F_к$ – ширину спектра частот, пропускаемых каналом и через $T_к$ – время занятости канала. Тогда свойства канала можно охарактеризовать произведением $V_к = H_к \cdot F_к \cdot T_к$, которое называется ёмкостью канала.

Передача сигнала по каналу связи возможна лишь при соблюдении условий:

$$H_k >= H_c,\ F_k >= H_c,\ T_k >= T_c .$$

Из этого следует, что

$$H_к \cdot F_к \cdot T_к >= H_c \cdot F_c \cdot T_c$$

и, следовательно,

$$V_к >= V_c .$$

Последнее ограничение означает, что сигнал можно передать по каналу связи без искажений только в том случае, если ёмкость канала больше или равна V сигнала является абсолютным и не может быть преодолено (отдельно частоты, мощность и время являются относительными и могут быть преодолены).

Пример: 1) Контейнер. Груз не помещается по высоте. Груз можно переупаковать, так же можно поступить и с сигналом.

2) Предположим, необходимо передать речевое сообщение длинной $T_{с1}$ = 10 мин. Спектр частот $f_{н1}$ = 300 Гц, $f_{в1}$ = 3400 Гц, что соответствует ширине полосы $F_{с1}$ = 3100 Гц. Предположим, что канал для передачи может быть предоставлен на $T_{с2}$ = 2 мин. Следовательно, речевое сообщение можно записать на магнитофон и при передаче пропустить магнитофонную ленту со скоростью в 5 раз большей, то есть за 2 мин. Естественно, что частоты

также возрастут в 5 раз: $f_{н2} = 5 \cdot f_{н1} = 5 \cdot 300$ Гц = 1500 Гц, а $f_{в1} = 5 \cdot f_{в1} = 17\,000$ Гц. Новая ширина полосы $F_{с2} = 15\,500$ Гц, то есть в 5 раз шире, чем $F_{с1} = 3\,100$ Гц. Если интенсивность сигнала остаётся неизменной $H_{с2} = H_{с1}$, то очевидно $V_{с2} = H_{с2} \cdot F_{с2} \cdot T_{с2} = H_{с1} \cdot F_{с1} \cdot T_{с1} = V_{с1}$. Значит, объём сигнала не изменится, хотя длительность его изменилась во столько раз, во сколько увеличилась ширина полосы.

На принимающем конце сигнал вновь нужно записать на магнитофонную ленту, чтобы восстановить естественное звучание и длительность сообщения.

Ёмкости действующих каналов связи обычно бывают больше объёма передаваемых по ним сигналов, и для повышения эффективности использования каналов необходимо добиваться возможно большего приближения к равенству $V_к = V_с$. Но нас в конечном итоге интересует не объём переданных сигналов, а объём сообщений. Сигналы являются лишь удобной для передачи формой, в которую заключено передаваемое сообщение, и одним из важных является вопрос измерения количества информации, содержащейся в сообщении.

1.1.2. Формы представления информации и её преобразования

Информация о каких-либо событиях и состоянии материальных объектов и, соответственно, сигналы, в которых содержится эта информация, могут быть представлены в непрерывной (аналоговой) и дискретной форме.

Понятие «непрерывности сигнала» включает его непрерывность во времени и по уровню. Первое свойство означает, что в любой произвольно взятый момент вре-

мени значение сигнала соответствует некоторому сообщению. Второе свойство указывает на то, что величина сигнала может принимать бесконечное множество различных значений, которые могут отличаться друг от друга сколько угодно малыми приращениями. Существуют также непрерывные сигналы, обладающие свойством непрерывности только во времени или только по уровню.

Дискретные сигналы можно разделить на сигналы дискретные во времени и дискретные по величине. Дискретность во времени означает, что сигнал содержит информацию лишь в некоторый фиксированный момент времени. В промежутках между этими моментами сигнал может либо вообще отсутствовать, либо его значение может не содержать полезной информации. Дискретность по величине означает, что амплитуда или величина, уровень сигнала может принимать лишь определённое конечное количество значений.

Подобно непрерывному сигналу, дискретный сигнал может обладать свойством дискретности только во времени или только по величине. В этих случаях форму сигналов можно определить как непрерывно-дискретную. Для автоматического преобразования электрических информационных сигналов из одной формы в другую разработано большое количество типов преобразователей, основанных на различных принципах и имеющих различную схемную реализацию. В зависимости от их назначения все эти преобразования можно разбить на два больших класса. Одни из них предназначаются для преобразования информации из непрерывной формы в дискретную, и называются аналого-цифровыми; другие –

из дискретной формы в непрерывную, и называются цифро-аналоговыми.

Рассмотрим один из весьма важных для техники передачи информации по каналам связи вопрос о квантовании непрерывных сигналов. Процесс непрерывно-дискретного преобразования сводится при квантовании во времени к определению ряда значений непрерывного сигнала $u(t)$, взятых последовательно в определённые фиксированные моменты времени $t_0, t_1, t_2, \ldots, t_n$.

Естественно, что для абсолютно точного представления непрерывной функции на некотором конечном отрезке времени T необходимо при квантовании взять бесчисленное множество отсчётов её значений. При этом очевидно: шаг квантования $\Delta t \to 0$. Практически, конечно, такое аналого-дискретное преобразование реализовать невозможно. Но соблюдение этого условия было бы необходимо только для идеальных непрерывных функций с бесконечным частотным спектром. Как известно, непрерывную функцию времени u(t) можно представить, по Фурье, в виде суммы некоторой постоянной составляющей, основной гармонической функции и бесконечного ряда её гармоник, то есть гармонических колебаний удвоенных, утроенных и так далее частот.

Однако все реальные функции, которые необходимо передавать по каналам связи, можно считать ограниченными по своему спектральному составу. Например, при телефонной передаче не имеет смысла передавать частотные составляющие выше 20 кГц, ибо их человеческое ухо не воспринимает. Более того, для хорошей разборчивости телефонных сообщений достаточно воспроизводить диапазон частот лишь в пределах от 300 до 3400 Гц (не-

сколько нарушается тембр голоса, но воспроизведение не нарушается).

Для подобных функций применительно к передаче информации по каналам связи В.А. Котельниковым доказана очень важная теорема, гласящая, что любой сигнал, имеющий ограниченный спектр частот, полностью определяется последовательностью своих мгновенных значений, отсчитанных через интервалы времени

$$\Delta t = 1/2f_c\,, \qquad (1.4)$$

где f_c – верхняя граничная частота спектра непрерывного сигнала.

Теорема Котельникова является теоретической основы осуществления передачи непрерывных сигналов в виде последовательностей их дискретных значений, что находит самое широкое применение при различных видах импульсной модуляции. А это в свою очередь позволяет реализовать методы импульсно-временного уплотнения каналов связи, при котором по одному каналу можно одновременно передавать без помех несколько непрерывных сообщений, например, телефонных разговоров. Большое значение эта теорема имеет также для обоснования надёжных реализаций различных аналого-цифровых преобразователей, предназначенных для ввода аналоговой информации в компьютер.

1.2. Измерение информации

Понятие о количестве информации и возможности его измерения является основным в теории информации. Этот вопрос был освещён американским инженером К. Шенноном в 1948 году в статье «Математическая теория связи». С этого времени и началось интенсивное раз-

витие теории информации вообще и углубленное изучение вопроса об измерении её количества, в частности.

Ранее понятие о количественном соотношении между информацией и степенью упорядоченности (энтропией) в физике обосновал Л. Больцман (1872); в математической статистике – Р. Фишер (1921); в применении к проблеме хранения информации и передаче её по каналам связи задачами определения количества информации занимались Х. Найквист (1924) и Р. Хартли (1928).

Общая черта, свойственная всем перечисленным методам, – отвлечение от содержания, смысла и семантики информации.

Пример: выбор одной из 4-х возможностей (например, перекрёсток). Количество полученной информации при выборе одного из символов (А, Б, В, Г) будет одинаковой.

Всю совокупность символов, которыми закодированы возможные события и сообщения о них, называют алфавитом сообщений.

Р. Хартли предложил определять информационную ёмкость системы, передающей или накапливающей информацию, и количества информации в сообщении. С именем Хартли связано структурное направление измерения количества информации, в основу которого положено понятие о дискретном строении массивов информации и их разнообразии без учёта условий использования источников информации. При структурном подходе принимается во внимание возможность получения от источника информации некоторого дискретного множества сообщений, которое в свою очередь обусловлено количеством информационных элементов и связей между ними. Если количество возможных сообщений в этом множе-

стве равно N, то, по Хартли, информация I_1, приходящаяся на одно сообщение, определяется логарифмом общего числа возможных сообщений

$$I_1 = \log N \, . \qquad (1.5)$$

Если сообщение поступает из двух источников, которое содержит N_1 и N_2 возможных сообщений, то общее число этих сообщений $N = N_1 \cdot N_2$, так как возможно объединение этих сообщений. Поэтому количество информации I_1, приходящееся на одно сообщение, будет равно

$$I_1 = \log N = \log N_1 + \log N_2 \, . \qquad (6)$$

Таким образом, выявлено одно из важнейших свойств применения логарифмической меры количества информации – свойство аддитивности. В самом деле, количество информации на сообщение равно сумме количеств информации, которые были получены от двух независимых источников (основание логарифма может быть любым, но удобно брать основание 2, то есть логарифм будет двоичным). Недостаток структурного подхода в том, что здесь не учитывается вероятность поступления сообщений от источника.

Пример: в первой урне 999 чёрных шаров и 1 белый, во второй – 500 белых и 500 чёрных шаров. Опыт состоит в извлечении шаров. Любая информация приводит к снятию информации. В первом случае – меньшее; во втором случае – значительно большее количество информации для получателя.

Необходимо учитывать при определении количества информации не только количество разнообразных сообщений, но и вероятность получения тех или иных сообщений, то есть вероятностные характеристики источника и положены в основу статистического подхода к определению количества информации. Такой подход был пред-

ложен К. Шенноном в 1948 году и получил широкое распространение при определении среднего количества информации, которое содержится в сообщениях от источников самой разной природы.

1.3. Энтропия

1.3.1. Энтропия как мера степени неопределенности

Большинство источников информации характеризуются неодинаковой вероятностью происходящих в них событий, следовательно, неравной вероятностью появления сообщений об этих событиях. Выполнение тех или иных опытов, связанных с этими событиями, всегда несёт черты большей или меньшей неопределённости. Степень этой неопределённости определяется вероятностными характеристиками источника сообщений (как с шарами).

Для сравнения подобных вероятностных источников сообщений необходима численная оценка степени неопределённости соответствующих опытов.

Рассмотрим источники, в которых опыты имеют некоторое количество n равновероятных исходов. Естественно предположить, что неопределённость H исхода опыта зависит от количества возможных исходов, то есть

$$H = f(n) \ .$$

Сформулируем логические предпосылки, которые целесообразно положить в основу выбора характера этой функциональной зависимости:

1) с увеличением числа возможных исходов неопределенность опыта должна возрастать.

2) при $n = 1$, когда возможен только один исход опыта, одно событие, опыт приобретает априорную определённость и его неопределённость должна обращаться в нуль.

3) неопределённость исхода сложного опыта, заключающегося в одновременном выполнении двух опытов в двух независящих друг от друга источниках, должна быть больше, чем неопределённость каждого из этих опытов, так как к неопределённости одного из них добавляется неопределённости другого опыта. Удобно предположить, что при этом неопределённости составных опытов суммируются.

Всем этим требованиям удовлетворяет логарифмическая функциональная зависимость:

$$H = f(n) = \log n . \qquad (1.7)$$

Действительно, при $n = 1$ величина $H = 0$, а при росте n неопределённость H возрастает. Если осуществляется два независимых опыта α и β, которые могут иметь соответственно m и n равновероятных исходов, то сложный опыт γ, заключающийся в одновременном выполнении опытов α и β, имеет $m \cdot n$ равновероятных исходов и его неопределённость согласно формуле (1.7) выражается функцией

$$H_\gamma = \log(m \cdot n) = \log m + \log n = H_\alpha + H_\beta . \qquad (1.8)$$

Это удовлетворяет предположению о равенстве неопределённости сложного опыта сумме неопределённостей составляющих его опытов. Если в приведённых выражениях брать логарифм при основании 2, то, принимая

$n = 2$, приходим к общепринятой единице измерения не-

определенности получившей название двоичная единица или бит (*binary unit* – двоичная единица).

$$H_1 = \log_2 2 = 1$$

Итак, двоичная единица, или бит, есть единица измерения степени неопределенности, представляющая неопределенность, которая содержится в одном опыте, имеющем два равновероятных исхода (например, подбрасывание монеты).

Рассмотрим опыт, который может иметь n равновероятных исходов. Такую неопределённость мы условились считать равной $H = \log n$. Это выражение можно представить в виде:

$$H = n \cdot \frac{1}{n} \cdot \log n = n \cdot \left(-\frac{1}{n} \cdot \log \frac{1}{n}\right). \qquad (1.9)$$

Так как 1/n = p, где p – вероятность любого из n равновероятных исходов опыта, то выражение (1.9) можно представить в виде:

$$H = -n \cdot p \cdot \log p \cdot \qquad (1.10)$$

Если некоторый опыт α характеризуется таблицей вероятностей (см. табл. 1), то по аналогии с формулой (1.10) меру неопределённости или, что одно и то же, меру количества разнообразия такого опыта можно записать в виде

$$H = -p_1 \cdot \log p_1 - \ldots - p_i \cdot \log p_i - \ldots p_n \cdot \log p_n = -\sum_{i=1}^{n} p_i \cdot \log p_i \ . \qquad (1.11)$$

Полученное выражение (1.11) имеет вид, совпадающий с видом выражения для энтропии в статистической физике, причём это совпадение несёт не только формальный, но и содержательный характер. Поэтому величину H_α называют энтропией опыта α (различают физическую и информационную энтропию).

Таблица 1. Таблица вероятностей и исходов опыта α

Исход опыта	A_1	A_2	…	A_i	…	A_n
Вероятность исхода	p_1	p_2	…	p_i	…	p_n

Выводы:

1) энтропия любого события всегда положительна, то есть $H >= 0$. Энтропия опыта может быть равна 0 лишь в том случае, когда одна из вероятностей p_1, p_2, p_3, ..., p_i, ..., p_n равна 1, а все остальные 0, так как возможен лишь один исход опыта ($H = 0$);

2) наибольшей неопределенностью среди всех опытов, имеющих n исходов, характеризуется опыт, у которого все эти исходы равновероятны.

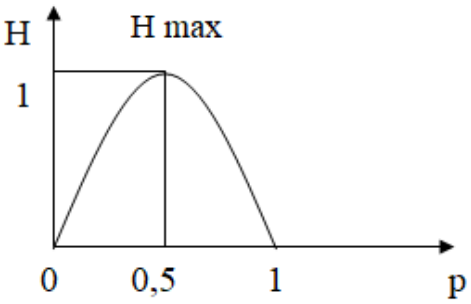


Рис. 1.4. График зависимости энтропии от вероятности

Энтропия такого опыта с равновероятными исходами будет максимальной. График зависимости энтропии опыта с двумя возможными исходами ($n = 2$), как функция вероятности одного из исходов p. Очевидно, что вероятность другого исхода (1 - p) и выражение для энтропии

$$H = -p \log p - (1-p) \log(1-p).$$

1.3.2. Энтропия сложных событий. Условная энтропия

Пусть имеют место два независимых опыта α и β, которые характеризуются следующей таблицей вероятностей (см. табл. 2 и табл. 3).

Таблица 2. Таблица вероятностей и исходов опыта α

Исход опыта	A_1	…	A_i	…	A_n
Вероятность исхода	p_{A1}	…	p_{Ai}	…	p_{An}

Таблица 3. Таблица вероятностей и исходов опыта β

Исход опыта	B_1	…	B_i	…	B_m
Вероятность исхода	p_{B1}	…	p_{Bi}	…	p_{Bm}

Рассмотрим сложный опыт γ, который заключается в одновременной реализации опытов α и β. Опыт γ может иметь $m \cdot n$ исходов, причём неопределённость этого опыта больше неопределённости каждого из рассмотренных отдельных опытов α и β. Можно доказать равенство, которое представляет правило сложения энтропий:

$$H_\gamma = H_\alpha + H_\beta, H_\gamma > H_\beta, H_\gamma > H_\alpha . \qquad (1.12)$$

Труднее дело обстоит с определением энтропии H_γ, сложного опыта γ, заключающегося также в одновременной реализации опытов α и β, которые, однако, зависят друг от друга.

Пример: последовательный вызов учеников к доске на уроке. Очевидно, после вызова некоторого ученика К* распределение вероятностей вызова других учеников изменяется, а вероятность повторного вызова того же ученика К* очень сильно уменьшается, хотя и не становится равной нулю. Энтропия H_γ такого сложного опыта,

как правило, больше энтропии входящих в него опытов, однако обычно меньше простой суммы энтропий этих опытов, хотя и может быть равна ей.

При этом энтропия определяется по следующим формулам:

$$H_\gamma = H_\alpha + H_{\beta/\alpha} \quad . \qquad (1.13\ а)$$

$$H_\gamma = H_\beta + H_{\alpha/\beta} \ . \qquad (1.13\ б)$$

В выражениях (1.13) H_α и H_β соответственно энтропии отдельно взятых опытов α и β:

$$H_\alpha = -\sum_{i=1}^{n} p_{A_i} \log\ p_{A_i} \quad . \qquad (1.14\ а)$$

$$H_\beta = -\sum_{j=1}^{m} p_{B_j} \log\ p_{B_j} \quad . \qquad (1.14\ б)$$

где $H_{\beta/\alpha}$ – средняя условная энтропия опыта β при условии выполнения опыта α, $H_{\alpha/\beta}$ – средняя условная энтропия опыта α при условии выполнения опыта β.

Условные энтропии определяются по формулам:

$$H_{\beta/\alpha} = -\sum_{i=1}^{n} p_{A_i} H_{\beta/A_i}, \qquad (1.15\ а)$$

$$H_{\alpha/\beta} = -\sum_{j=1}^{m} p_{B_j} H_{\alpha/B_j}, \qquad (1.15\ б)$$

где

$$H_{\beta/A_i} = -\sum_{j=1}^{m} p_{B_j/A_i} \log p_{B_j/A_i} \ . \qquad (1.16\ а)$$

$$H_{\alpha/B_j} = -\sum_{i=1}^{n} p_{A_i/B_j} \log p_{A_i/B_j} \ .. \qquad (1.16\ б)$$

Объединяя выражения (1.15) и (1.16), получаем

$$H_{\beta/\alpha} = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{A_i}\, p_{B_j/A_i} \log p_{B_j/A_i} \qquad (1.17а)$$

где $p_{Bj/Ai}$ – условная энтропия события B_j, если имело место событие A_i,

$$H_{\alpha/\beta} = -\sum_{j=1}^{m}\sum_{i=1}^{n} p_{B_j}\, p_{A_i/B_j} \log p_{A_i/B_j} \qquad (1.17\ б)$$

и $p_{Ai/Bj}$ – условная энтропия события A_i, если имело место событие B_j.

Во всех случаях имеет место соблюдение условий:

$$0 <= H_{\beta/\alpha} <= H_{\beta}\,, \qquad (1.18\ а)$$

$$0 <= H_{\alpha/\beta} <= H_{\alpha}\,, \qquad (1.18\ б)$$

то есть условная энтропия $H_{в/а}$ заключается между 0 и безусловной энтропией $H_в$ опыта β (то же и для $H_{а/в}$).

Таким образом, предварительное выполнение одного из опытов может лишь уменьшить степень неопределенности другого опыта (даже свести к 0) или, в крайнем случае, при независимости опытов не изменить эту неопределенность, но никак не может увеличить её.

Соответственно, сравнивая выражения (1.13) и (1.15), можно сделать вывод, что всегда

$$H_{\gamma} >= H_{а}\,, \qquad (1.19\ а)$$

$$H_{\gamma} >= H_{\beta}\,, \qquad (1.19\ б)$$

$$H_{\gamma} <= H_{\alpha} + H_{\beta}\ . \qquad (1.19\ в)$$

Пример: Сложный опыт заключается в последовательном осуществлении двух зависимых опытов α и в, представляющих извлечение шаров из урны, в которую были опущены два шара – чёрный и белый. Каждый из этих опытов выполняется независимо друг от друга и может иметь два равновероятных исхода. Поэтому энтропия в соответствии с функцией (1.11) будет $H_{\alpha} = H_{\beta} = 1$.

С другой стороны, так как после осуществления первого опыта, например, α, исходом которого является извлечение из урны шара одного цвета, нам определенно известно, что при последующем опыте будет извлечён шар другого цвета, то есть после первого опыта α опыт β уже не будет содержать никакой неопределённости ($H_{\beta/\alpha} = 0$). Таким образом, в соответствии с формулами (1.13) энтропия сложного опыта будет равна:

$$H_\gamma = H_\alpha = H_\beta .$$

1.4. Количество информации

Ранее установлено, что для двух зависимых опытов α и β средняя условная энтропия опыта β при условии осуществления опыта α, как правило, меньше безусловной энтропии опыта β, то есть $H_{\beta/\alpha} <= H_\beta$, причем разность $H_\beta - H_{\beta/\alpha}$ убывает по мере ослабления связи между этими опытами и в пределе становится равной нулю, т.е. $H_\beta = H_{\beta/\alpha}$, когда опыты становятся независимыми.

Таким образом, разность $H_\beta - H_{\beta/\alpha}$, количественно выражает уменьшение энтропии или неопределенности опыта β после осуществления опыта α. Или, можно сказать, увеличивает наши представления, знания о возможном исходе опыта β. Это даёт нам логическое основание для того, чтобы назвать эту разность информацией об опыте β, содержащемся в опыте α, то есть

$$I_{\alpha,\beta} = H_\beta - H_{\beta/\alpha} . \qquad (1.20)$$

Эти соображения закладывают основы количественного статистического подхода к измерению количества информации в соответствии с концепцией, предложенной К. Шенноном. На рисунке 1.5 показано, что энтропия

H_β опыта β после осуществления опыта α уменьшается на величину $I_{\alpha,\beta} = H_\beta - H_{\beta/\alpha}$.

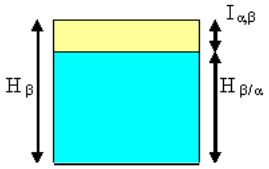


Рис. 1.5. Влияние условной и безусловной энтропии зависимых опытов на количество информации об этих опытах

Если бы опыты β и α были независимы, то $H_{\beta/\alpha} = H_\beta$ и $I_{\alpha,\beta} = 0$, и результат опыта α полностью снимает неопределённость опыта β. Таким образом, если $H_{\beta/\alpha} = 0$, то $I_{\alpha,\beta} = H_\beta$, энтропия H_β опыта β представляет информацию об этом опыте, которая содержится в самом опыте, так как осуществление опыта полностью определяет его исход и, следовательно, $H_{\beta/\beta} = 0$, то есть неопределённость опыта после его осуществления становится равной нулю. Можно также сказать, что энтропия H_β опыта β равна средней информации, которая содержится в одном исходе опыта β.

Если ввести для энтропии, взятой со знаком минус (отрицательной энтропии) определение негэнтропии, то приходим к выводу, что процессы получения информации, связанные с уменьшением энтропии, приводят к увеличению негэнтропии. Эти соображения легли в основу так называемого негэнтропийного принципа информации, разработанного известным французским физиком Л. Бриллюэном, сформулировавшим, в частности, возможность превращения негэнтропии в информацию и обратно.

Рассматривая сообщение как информацию об исходе некоторого опыта, можно утверждать, что чем более неопределенным являлось это сообщение до его получения, тем оно содержит большее количество информации. Поэтому количество информации I можно определить через отношение вероятностей $I = \log p'/p$, где p' – вероятность события, после поступления сообщения о нем, p – вероятность данного события до поступления сообщения о нём.

Так как числитель дроби – вероятность события после поступления сообщения о том, что оно произошло, очевидно, равен 1, то

$$I = \log 1/p = -\log p .$$

После того как мы рассмотрели логарифмическую сущность понятия количества информации, перейдём к определению среднего количества информации, которая приходится на одно сообщение, рассматриваемое как исход опыта, заключающегося в передаче сообщений.

Пусть мы имеем алфавит, который состоит из n некоторых элементов или символов (букв, цифр, математических знаков) $h_1, h_2, \ldots, h_n$. Пусть вероятности появления этих элементов в тексте сообщения равны: $p_1, p_2, \ldots, p_n$. Составим из этих элементов ряд сообщений, каждое из которых содержит m элементов, причём m – достаточно большое (теоретически – бесконечно большое) количество, позволяющее при рассуждениях использовать аппарат теории вероятностей. Среди этих m элементов в сообщении будет m_1 элементов h_1; m_2 элементов h_2; …; m_n элементов h_n, причём

$$m_1 = p_i \cdot m . \qquad (1.21)$$

Предполагая, что появление каждого из элементов есть независимое событие (что справедливо для цифро-

вых и, вообще говоря, несправедливо для буквенных алфавитов), можно считать, что вероятность появления каждой комбинации (сообщения) длиной m элементов определяется, как произведение вероятностей появления отдельных элементов. Учитывая многократную повторяемость одних и тех же элементов в сообщениях, определим вероятность p появления некоторого сообщения:

$$p = p_1 m_1 \cdot p_2 m_2 \cdot \ldots \cdot p_n m_n = \prod_1^m p_i m_i \qquad (1.22)$$

где p – вероятность появления некоторого сообщения. Считаем, что все N возможных сообщений (перестановок) равновероятны, следовательно,

$$p = 1 / N \ . \qquad (1.23)$$

Из (1.21), (1.22, (1.23) следует:

$$N = \frac{1}{\prod_1^n p_i m^{p_i}} \ . \qquad (1.24)$$

Прологарифмируем это выражение, в результате чего получим энтропию, или количество информации, которое приходится на одно сообщение длинной m элементов при неравно вероятности этих элементов.

$$I = \log N = -m \sum_{i=1}^{n} p_i \log \ p_i \, . \qquad (25)$$

Учитывая свойство аддитивности логарифмической меры энтропии и информации, можно найти среднее количество информации, которое приходится на один символ сообщения.

$$I_1 = I / m = -\sum_{i=1}^{n} p_i \log \ p_i \ . \qquad (1.26)$$

Эта формула полностью совпадает с формулой для расчёта неопределённости опыта с неравновероятным исходом (1.11).

Выражение (1.26) справедливо для алфавита, который содержит n неравновероятных символов с вероятностями их появления $p_1, p_2, \ldots, p_n$.

Если алфавит источника сообщения содержит всего два символа, появление которых характеризуется равными вероятностями $p = 0{,}5$, то выражение (1.26) примет вид

$$I_1 = -2 \cdot 0{,}5 \cdot \log 0{,}5 = 1 \text{ бит}.$$

Таким образом, подобно неопределённости и энтропии, за единицу количества информации принят 1 бит.

Бит можно определить как количество информации, которое содержится в результате одиночного выбора из двух равновероятных возможностей. Так как в двоичной системе счисления каждый разряд числа с равной вероятностью может принимать значения 0 или 1, то соответственно и количество информации, приходящейся на один двоичный разряд (двоичную цифру) оказывается равна 1 биту. При обработке информации в машинах ради удобства представляем слова в виде совокупных неделимых частей некоторой стандартной длины. В качестве таких частей выбраны восьмиразрядные порции, поэтому наряду с битом используется большая единица – байт (1 байт = 8 бит).

Пример: рассмотрим способ определения количества информации I в некотором отрывке русского текста. Для этого нужно знать вероятности (p_i) появления в тексте всех 32 букв русского алфавита (е, ё – одна буква, ъ, ь – считают вместе, прибавляем пробел). Тогда

$$I = m \cdot I_1$$

где I_1 – количество информации на одну букву в тексте, m – количество букв в тексте.

Для анализа лучше использовать достаточно большие тексты, до десятков тысяч символов. В результате анализа текста подсчитываем вероятности появления всех букв и пробела. Средняя информация на одну букву русского алфавита равна:

$$I_1 = -\sum_{i=1}^{n} p_i \log \; p_i = -(0{,}062 \log 0{,}062 + \ldots + 0{,}174 \log 0{,}174) = 4{,}35$$

бит.

Если бы все 32 буквы русского алфавита были бы равновероятны, то количество информации на одну букву было бы равно

$$I_1 = \log 32 = 5 \text{ бит.}$$

Таким образом, неравновероятность использования букв алфавита приводит к уменьшению информации, которая содержится в одной букве русского алфавита приблизительно на 0,65 бит.

Вероятность использования букв русского алфавита в тексте представлена в таблице 4.

Таблица 4. Таблица вероятностей букв русского алфавита

Буква	Вероятность	Буква	Вероятность	Буква	Вероятность
а	0,062	л	0,035	ц	0,004
б	0,014	м	0,026	ч	0,012
в	0,038	н	0,053	ш	0,006
г	0,013	о	0,090	щ	0,003
д	0,025	п	0,023	ы	0,016
е	0,072	р	0,040	ъ, ь	0,014
ж	0,007	с	0,045	э	0,003
з	0,016	т	0,053	ю	0,006
и	0,061	у	0,021	я	0,018
й	0,010	ф	0,002	пробел	0,174
к	0,028	х	0,009		

1.5. Общие представления об избыточности

Изложение вопроса избыточности начнём с некоторых общих соображений, касающихся структуры русского языка. Для удобства вычислений будем считать, что русский алфавит содержит 32 буквы (см. параграф 1.4). Рассмотрим вопрос о количестве слов разной длины, которые можно было бы составить.

Эта задача комбинаторики сводится к определению количества N возможных различных размещений с повторением из n элементов по k, то есть

$$N = n^k, \qquad (27)$$

где N – возможное количество различных слов, k – длина слов. Следовательно, пользуясь 32-буквенным алфавитом, можно составить:

$N_1 = 32^1 = 32$ однобуквенных слова ($k = 1$);

$N_2 = 32^2 = 1024$ двухбуквенных слова ($k = 2$);

$N_3 = 32^3 = 32768$ трёхбуквенных слова ($k = 3$);

$N_4 = 32^4 > 1$ миллиона четырёхбуквенных слов ($k = 4$);

$N_5 = 32^5 > 30$ миллионов пятибуквенных слов ($k = 5$).

В словарях обычно содержится от десятков до сотен тысяч слов. Однако большинство слов не употребляются в повседневной речи. Для автоматических переводчиков научных текстов на компьютере достаточно задать 5 тысяч слов общего пользования и столько же из соответствующей области науки. Естественно, иногда будут встречаться слова, которые машина «не поймёт», но то же имеет место и при работе даже высококвалифицированного переводчика.

Приведённые соображения позволяют сделать вывод, что, пользуясь 32-буквенным алфавитом, можно будет построить некий гипотетический язык с 32 768

трёхбуквенными словами, такими как ааб, аба, абб, баа, бба, аав и другими. Можно использовать четырёхбуквенные слова (более 1 миллиона слов), больше, чем в любом живом языке. Есть в русском языке и слова длинной более 20 букв, например, слово «малоквалифицированный», а средняя длинна слов – 6 букв.

Таким образом, можно было бы сократить объём книг, длительность лекций и другое, то есть получить огромную экономию. Но это невозможно по трём причинам:

1) любой живой язык является стойкой исторически сложившейся категорией, сохраняющей особый строй лексики и грамматики (например, эсперанто – искусственный язык);

2) слова искомого языка были бы неудобопроизносимыми и трудночитаемыми;

3) такой язык являлся бы малонадёжным средством общения.

Пример: ааа – вперёд, ааб – назад. Одна ошибка может привести к аварии, а в живом языке слово «здравствуйте» в худшем случаем может стать малопонятным.

Если средняя длина слова 6 букв, то таких слов можно составить более 1 миллиарда, а мы используем до 50–100 тысяч (с учетом пассивного словаря).

В любом реальном языке, таким образом, наблюдается избыточность – свойство, характеризующее возможность представления той же информации в более экономной форме, то есть более короткими кодами.

Пример: «корона». При замене одной буквы можно получить «борона», «ворона», «корова» и так далее.

Кроме избыточности длины слов, естественные языки характеризуются избыточностью употребления слов («значит», «типа» и так далее).

Всё это относится к информационной избыточности или избыточности кодирования.

Существует также структурная, временная, функциональная избыточность.

Структурная избыточность заключается в дублировании и многократном резервировании оборудования.

Временная избыточность – совокупность методов повышения надёжности системы за счёт увеличения времени решения тех или иных проблем (повторение, передача, перерасчёт).

Функциональная избыточность – совокупность мер, применяемых для сохранения работоспособности системы при выходе за пределы допусков (напряжение, температура, давление).

1.6. О содержательности и полезности информации

1.6.1. Понятие о семиотике

Приведённый выше метод измерения количества информации полностью игнорирует её смысл (содержание) и полезность для достижения цели (ценность, целесообразность). Количество информации, определённое формулой

$$I_1 = -\sum_{i=1}^{n} p_i \log \quad p_i \ ,$$

является лишь усреднённой мерой неопределённости появления этого символа и общее количество информации, рассчитанной по формуле (1.25), никак не связано с

содержательностью и полезностью этой информации для получателя.

Пример: рассмотрим три сообщения:

1) количество информации растёт год от года;
2) американские астрономы обнаружили звезду;
3) срочно отправьте сорок шесть компьютеров.

Каждое из этих сообщений содержит по 41 букве. Можно подсчитать общее количество информации в сообщении. Если среднее количество информации, приходящееся на одну букву алфавита, $I_1 = 4{,}35$ бит, то оно будет одинаково во всех трёх сообщениях и будет равно

$$I_A = I_Б = I_B = 4{,}35 * 41 \approx 178.35 \text{ бит}.$$

Мы рассчитали общее количество информации безотносительно к содержанию сообщений. Для телеграфистки действительно это не важно (время передачи каждого из трёх сообщений будет одинаковым).

Если обратиться к содержанию, то первое сообщение не несёт никакой новой информации, второе интересно только для астрономов, а третье несёт важную информацию с точки зрения управления процессом доставки компьютеров.

Таким образом, мы подошли к различным аспектам оценки количества информации не только по формально-структурным признакам, но и по их содержанию и практической ценности для получателя. Во всех случаях мы сталкиваемся с любыми сообщениями. Эти сообщения выражены некоторыми знаками – условными обозначениями, изображениями; словами – совокупностью знаков, имеющих смысловое значение; языками – словарём и правилами пользования им.

Таким образом, рассуждая о количестве, содержании и ценности информации, содержащейся в сообщении, нужно исходить из возможностей соответствующего анализа знаковых структур.

Этими вопросами занимается относительно новая отрасль знаний – семиотика.

Семиотика – это комплекс научных теорий, изучающих свойства знаковых систем, то есть систем, объектов различной природы, называемых знаками, каждому из которых определённым образом сопоставлено некоторое значение. В таком понимании знаковыми системами являются естественные и искусственные языки, в том числе информационные языки и языки программирования, различные системы сигнализации, логическая, математическая, химическая символики и другие.

При рассмотрении семиотической проблематики выделяют три её основных аспекта: семантику, синтактику и прагматику.

Семантика изучает знаковые системы как средство выражения смысла, определенного содержания, то есть правила интерпретации знаков и их сочетаний, смысловую сторону языка.

Синтактика изучает синтаксис знаковых структур, то есть способы сочетания знаков и правила образования этих сочетаний и их преобразование безотносительно к их значениям и каким бы то ни было функциям знаковых систем.

Таким образом, структурный и статистический способы определения количества информации в сообщении можно отнести к синтаксическим.

Прагматика рассматривает соотношение между знаковыми системами и их пользователями, или приёмниками – интерпретаторами сообщений, значит к прагматике относится изучение практической полезности знаков, слов и, следовательно, сообщений, то есть потребительской стороны языка.

1.6.2. Семантический подход к определению количества информации

Методы точного количественного определения смыслового содержания информации в настоящее время мало разработаны. Существуют некоторые рациональные соображения и подходы к решению этой проблемы.

Р. Карнап и И. Бар-Хиллел предложили определять величину семантической информации через так называемую логическую вероятность, которая представляет собой степень подтверждения той или иной гипотезы. При этом количество семантической информации, содержащейся в сообщении, возрастает по мере уменьшения степени подтверждения априорной, доопытной гипотезы.

Мера семантической информации учитывает не только структуру и содержание самого сообщения, но и запас знаний и предположений получателя, то есть что нового несет получателю сообщение по сравнению с тем, что он уже знал.

Пример: логическая вероятность равна 1, ели гипотеза, построенная на эмпирических данных, полностью подтверждена сообщением, то есть не несёт новых знаний и, значит, семантическая информация $I_c = 0$. И наоборот, по мере уменьшения степени подтверждения

гипотезы, или запаса знаний, количество семантической информации возрастает.

Ю.А. Шрейдером предложена идея, основанная на учёте «запаса знаний» получателя или приёмника сообщений.

Пример: шофёр послан для получения легкового автомобиля, грузовика и прицепа. Он должен сообщить какой-то результат (легковой – получен, грузовик – не получен, прицеп – не получен). Можно было бы договориться об условном сокращении: «да, нет, нет» или «100».

Очевидно, что таким образом можно закодировать и более сложные и длинные сообщения, но необходимым условием их информативности, содержательности для получателя является наличие у получателя запаса знаний, словаря, при помощи которого получатель сможет истолковать полученное сообщение, извлечь смысл и понять его.

Следовательно, содержательность, семантическая информация одного и того же сообщения оказывается неодинаковой для различных получателей и зависит от их тезауруса.

Тезаурус (от греч. «сокровище») – словарь, в котором указаны не только значения отдельных слов, но и смысловые связи между ними.

Пример: проиллюстрируем наглядно количественную зависимость семантической информации, которая содержится в одном и том же сообщении, от тезауруса получателя этой информации. Пусть рассматриваемое сообщение представляет собой формулу

$$\int e^{x^2} dx .$$

Для того, чтобы извлечь из этого сообщения какую-либо семантическую информацию I_c, получатель должен обладать знаниями, причём зависимость $I_c = f(T)$ можно изобразить графически в виде кривой (рис. 1.6).

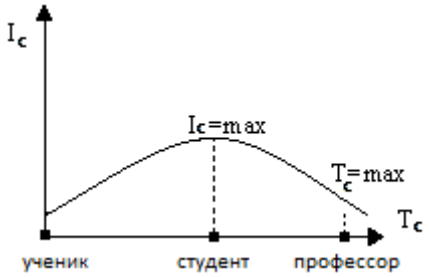


Рис. 1.6. Зависимость семантической информации от тезауруса

1.6.3. Прагматический подход к определению количества информации

Одним из непременных свойств информации является ее использование в процессе управления. А так как информация используется для управления, то и оценивать её надо не только с точки зрения структурно-статистической или семантической, но и с точки зрения её полезности, ценности, целесообразности для достижения поставленной цели управления.

Пример: управление судном (радиомаяки, многочисленные приборы и др.).

Каждое сообщение должно отправляться не с точки зрения его познавательных характеристик, а с точки зрения прагматической (от греч. «прагма» – действие, практика), то есть полезности или ценности для выполнения функций управления.

Исходя из этих соображений, А.А. Харкевич предложил определить такую меру ценности информации I_u,

как изменение вероятности достижения этой цели при получении информации:

$$I_ц = \log p_1 - \log p_0 = \log p_1/p_0, \qquad (1.28)$$

где p_0 – начальная вероятность достижения цели до получения информации, p_1 – вероятность достижения цели после получения информации.

Возможны 3 варианта:

1) $p_0 < p_1$	2) $p_0 = p_1$	3) $p_0 > p_1$
$I_ц > 0$	$I_ц = 0$	$I_ц < 0$.

Пример: указатели на дороге (три варианта достижения цели: 1) не знали, куда ехать, по указателю приехали, 2) знали, куда ехать, по указателю приехали, 3) не знали, куда ехать, по указателю не приехали).

Контрольные вопросы

1. В каких формах представляется информация?
2. Сформулировать теорему В.А. Котельникова.
3. Перечислить свойства информации.
4. Как вычислить объем сигнала и емкость канала связи?
5. Формулы для измерения количества информации.
6. Что такое энтропия? Условная энтропия?
7. Как вычислить энтропию сложного опыта?
8. Как связаны понятия неопределенности и вероятности?
9. Как влияет выбор вариантов решения на неопределенность?
10. Чему равно количество информации по Р. Хартли?
11. Как связана вероятность с информативностью?
12. Дать определение количества информации.

13. Как вычисляется количество информации по формуле К. Шеннона?
14. В каком случае формула Шеннона переходит в формулу Хартли?
15. К какому подходу относятся методы Р. Хартли и К. Шеннона?
16. Чем занимается наука семиотика?
17. С каких позиций рассматриваются знаковые системы?
18. В чем состоит основная идея семантической концепции информации?
19. В чем заключается подход Р. Карнапа и И. Бар-Хиллела?
20. Кому принадлежит идея учета «запаса знаний»?
21. В чем суть прагматического подхода к измерению количества информации?
22. Дать определение алфавита, системы счисления.
23. Какая система счисления должна использоваться в компьютерах с точки зрения минимизации?
24. Избыточность.
25. Виды избыточности.

ГЛАВА II. КОДИРОВАНИЕ

Любое сообщение, подлежащее передаче по каналу связи, записи в память или переработке должно быть представлено в виде некоторой последовательности символов или, другими словами, закодировано.

Кодирование есть преобразование сообщения в код, то есть в совокупность символов, отображающих сообщение, передаваемое по каналу связи.

Конечное множество попарно различных символов, применяемых в какой-либо системе или языке, называется алфавитом, а количество различных символов – объемом алфавита.

Примеры.

1. Описанная в главе 3 позиционная система счисления – это способ кодирования чисел. Причем алфавит и его объем зависит от выбранного основания: в десятичной системе это {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, в шестнадцатеричной – {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *F*}, в двоичной – {0, 1}.

2. Другой способ кодирования натуральных чисел – римские цифры.

3. Еще один способ кодирования натуральных чисел – замена десятичных цифр их русскими названиями: нуль, один, два и т.д.

4. Декартовы координаты – способ кодирования геометрических объектов числами.

5. Текст программы на любом языке программирования – кодирование некоторого алгоритма.

Дадим формальное определение задачи кодирования.

Пусть заданы алфавиты $A=\{a_1,\ldots,a_n\}$, $B=\{b_1,\ldots,b_n\}$, элементы алфавита называются буквами, A^*, B^* – мно-

жества слов, то есть последовательностей букв в алфавитах A и B соответственно. Пусть также задана функция $F: S \to B^*$, $S \subset A^*$. Тогда функция F называется кодированием, элементы множества S – сообщениями, а элементы $\beta = F(\alpha)$, $\alpha \in S$, $\beta \in B^*$ – кодами соответствующих сообщений. Обратная функция F^1 (если она существует) называется декодированием.

Если $|B| = m$, то F называется m-ичным кодированием.

Выделим некоторые естественные свойства, которые требуются от кодирования:

1) оптимальность алфавита кодирования B, то есть минимизация количества элементов в устройствах хранения информации, их простота и надежность;

2) существование декодирования (при всей своей естественности оно требуется не всегда: трансляция программы с языка высокого уровня в машинные коды не требует однозначного декодирования – обратного перехода);

3) помехоустойчивость или исправление ошибок, возникающих при передаче информации по каналу связи;

4) заданная сложность кодирования и декодирования с целью недопущения возможности правильного декодирования посторонним лицам.

В этой главе мы рассмотрим несколько возможных задач теории кодирования, удовлетворяющих указанным свойствам.

2.1. Выбор алфавита для хранения информации

При рассмотрении позиционных систем счисления было отмечено, что при уменьшении основания системы счисления и, следовательно, упрощении алфавита происходит удлинение числового кода. В частности, код числа, записанного в двоичной системе счисления, оказывается в среднем приблизительно в 3,5 раза длиннее десятичного кода того же числа.

Так как во всех компьютерах в течении длительного времени приходится хранить большие информационные массивы, то одним из существенных критериев для выбора алфавита кодирования числовой информации, то есть основания системы счисления, является минимизация количества элементов в устройствах хранения, а также их простота и надежность.

При определении количества элементов будем исходить из естественного предположения, что для фиксации каждого из разрядных символов требуется количество простейших элементов, равное основанию системы счисления q. Тогда для хранения в некотором устройстве n-разрядных чисел всего потребуется m элементов, где

$$m = q \cdot n \ .$$

Наибольшее количество различных чисел N, которое может быть записано в этом устройстве:

$$N = q^n \ .$$

Отсюда

$$n = \frac{\ln N}{\ln q} \ .$$

Представляя это выражение в m, получим:

$$m = q \cdot \frac{\ln N}{\ln q} \ .$$

Для того чтобы определить, при каком основании q количество элементов m будет минимальным при фиксированном N, продифференцируем последнее выражение по q и, приравняв нулю полученную производную, определим q, соответствующую минимуму функции $m = f(q)$:

$$\frac{dm}{dq} = \ln N \cdot \frac{\ln q - 1}{(\ln q)^2} = 0.$$

Так как для любого $q \in N \setminus \{0, 1\}$ выражение

$$\frac{\ln N}{(\ln q)^2} \neq 0,$$

то

$$\ln q - 1 = 0,$$

откуда $q = e = 2{,}718\ldots$.

Таким образом, функция $m = f(q)$ имеет один минимум при $q = e$. Так как основание системы счисления может быть только натуральным числом, то q следует выбирать равным 2 или 3.

Пусть, например, максимальная емкость устройства хранения $N = 10^6$ чисел.

Тогда при различных основаниях q количество элементов в устройстве будут в соответствии с выражением для m следующие (см. табл. 5).

Таблица 5. Основание системы счисления
и количество элементов в системах хранения

q	2	3	4	5	10	20
m	39,2	38,24	39,2	42,9	60	91,5

Следовательно, если исходить из минимизации количества оборудования, то наиболее выгодной окажется

троичная система счисления и почти равноценными ей – двоичная и четверичная.

Однако двоичная система обладает и другими существенными преимуществами. Учитывая, что числа в двоичной системе записываются в виде комбинаций лишь двух цифр 0 и 1 и, приписывая значение «1» наличию тока или напряжения, а значение «0» – отсутствию тока или напряжения, можно чрезвычайно просто и надежно представлять числа в виде импульсов тока или напряжения. Кроме того, преимуществом двоичной системы счисления является чрезвычайная простота арифметических операций над числами. Наконец, наиболее простыми и надежными запоминающими и логическими элементами являются элементы с двумя устойчивыми состояниями.

Все сказанное является причиной того, что подавляющее число цифровых вычислительных систем, как и прочих технических информационных устройств, предназначается для хранения и обработки дискретной информации, закодированной в двоичной системе счисления.

Величина, способная принимать лишь два различных значения, представляет собой своеобразный информационный атом, получивший специальное наименование бит (это название – *bit* – образовано из двух начальных и последней букв английского выражения *binary unit*, т. е. двоичная единица).

Для кодирования алфавитов, которыми привык пользоваться человек, употребляются последовательности бит. Легко видеть, что последовательностями из n двоичных разрядов можно закодировать 2^n различных

символов. При $n = 8$ их число равно 256, что оказывается вполне достаточным для кодирования большинства встречающихся на практике алфавитов (исключая иероглифическое письмо).

Последовательность из 8 двоичных цифр называется байтом.

Ранее в информатике употреблялись 7-, 6- и даже 5-битовые байты, но в настоящее время 8-битовый байт установился в качестве международного стандарта кодирования строчных и прописных букв латинского алфавита, знаков препинания, десятичных цифр, а также ряда наиболее употребительных математических символов (знаки арифметических и логических операций, знаки равенства и неравенства и др.). Специальный байтовый код закреплен также за знаком пробел.

Число букв байтового алфавита (256 букв!) оказывается достаточным для представления, кроме этих символов, строчных и прописных букв русского алфавита, отличных по написанию от латинских букв. Остается резерв и для кодирования других символов, например, греческих букв.

Всюду в этой главе, если это не будет оговорено особо, мы будем предполагать, что $B = \{0, 1\}$, то есть рассматривать двоичное кодирование, причем слово «двоичное» будем опускать.

2.2. Алфавитное кодирование

Основные результаты общего характера сформулируем для случая побуквенного алфавитного кодирования.

Рассматривается следующая модель канала связи:

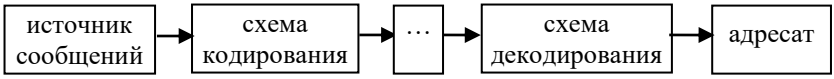


Рис. 2.1. Модель канала связи

$A = \{a_1, \ldots, a_n\}$ – алфавит языка сообщений, $B = \{0, 1\}$ – алфавит канала связи, то есть перечень сигналов, которые могут передаваться по каналу. Схема кодирования F отображает буквы A в слова B^* (B^* – моноид, порожденный B), $F(a_i) = B_i$ и слова из A^* кодируются побуквенно:. $F(a_{i_1}, a_{i_2}, \ldots, a_{i_k}) = F(a_{i_1})\, F(a_{i_2}) \ldots F(a_{i_k})$. Таким образом, отображение F полностью задается кодом $\{\beta_1, \beta_2, \ldots, \beta_n\}$, , где $\beta_i = F(a_i)$, $i = 1, \ldots, n$.. Если F взаимно однозначно, то задача схемы декодирования – восстановить переданное сообщение, реализуя отображение F^{-1}. Возникает вопрос об условиях существования однозначного декодирования.

Перейдем к изложению основных результатов. Если слово $\alpha = a_1 \ldots a_k \in A^*$, то количество букв в слове называется длиной слова:

$$|\alpha| = |a_1 \ldots a_k| = k .$$

Пустое слово обозначается $\wedge : \wedge \in A^*$, $|\wedge| = 0$, $\wedge \notin A$.

Если $\alpha = \alpha_1 \alpha_2$, то α_1 называется префиксом слова α, а α_2 –постфиксом слова α .

Алфавитное (или побуквенное) кодирование задается схемой (или таблицей кодов) δ:

$$\delta = \langle a_1 \to \beta_1, \ldots, a_n \to \beta_n \rangle,\ a_i \in A,\ \beta_i \in B^* .$$

Множество кодов букв $V = \{\beta_i\}$ называется множеством элементарных кодов.

Пример. Пусть A={0, 1, 2, 3, 4, 5, 6, 7, 8, 9,}, B={0, 1} и дана схема

$\delta = \langle 0\to 0,\ 1\to 1,\ 2\to 10,\ 3\to 11,\ 4\to 100,\ 5\to 101,\ 6\to 110,\ 7\to 111,$
$8\to 1000,\ 9\to 1001\rangle.$

Эта схема однозначна, но она не имеет однозначного декодирования:

$$F_\delta(22) = 1010 = F_\delta(50)\ .$$

Но если в качестве схемы рассмотреть двоично-десятичное представление, которое описано далее в главе 3

$\delta = \langle 0\to 0000,\ 1\to 0001,\ 2\to 0010,\ 3\to 0011,\ 4\to 0100,\ 5\to 0101,$
$6\to 0110,\ 7\to 0111,\ 8\to 1000,\ 9\to 1001\rangle,$

то она уже допускает однозначное декодирование.

Схема δ называется разделимой, если

$$\beta_{i_1}\ldots\beta_{i_k} = \beta_{j_1}\ldots\beta_{j_l} \Rightarrow k = l \& \forall \mathrm{t} \in 1,\ldots,k\ \ i_\mathrm{t} = j_\mathrm{t}\,.$$

Таким образом, в разделимой схеме любое слово, составленное из элементарных кодов, единственным образом разлагается на элементарные коды. Очевидно, что такая схема допускает однозначное декодирование.

Схема δ называется префиксной, если элементарный код одной буквы не является префиксом элементарного кода другой буквы:

$$(\forall i \neq j\ \beta_i,\ \beta_j \in V) \Rightarrow (\forall \beta \in B^*\ \beta_i \neq \beta_j\beta).$$

Лемма. Префиксная схема является разделимой.

Доказательство. Предположим противное, то есть кодирование с некоторой префиксной схемой δ не является разделимым. Тогда существует такое слово $\beta \in F_\delta(A^*)$, что

$$\beta = \beta_{i1}\ldots\beta_{ik} = \beta_{j1}\ldots\beta_{jl}\ \&\ (\exists \mathrm{t} \forall \mathrm{s}\ (\mathrm{s}<\mathrm{t} \Rightarrow \beta_{is} = \beta_{js}\ \&\ \beta_{it} \neq \beta_{jt})).$$

Так как $\beta_{it}\ldots\beta_{ik} = \beta_{jt}\ldots\beta_{jl}$, то $\exists\beta' = (\beta_{it} = \beta_{jt}\beta' \vee \beta_{jt} = \beta_{it}\beta'))$, но это противоречит тому, схема τ префиксная, ч. и т.д.

Свойство быть префиксной является достаточным, но не необходимым для разделимости схемы.

Пример. Следующая схема разделимая, но не префиксная

$$A = \{a, b\}, B = \{0, 1\}, \ \delta = \{a \to 0, b \to 01\}.$$

Следующее условие, накладываемые на длины элементарных кодов схемы кодирования, известное как неравенство Макмиллана, является необходимым и достаточным, чтобы существовало однозначное декодирование.

Теорема 1. Для того, чтобы схема $\delta = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ была разделима необходимо и достаточно, чтобы

$$\sum_{i=1}^{n} \frac{1}{2^{l_i}} \le 1, \text{ где } l_i = |\beta_i|. \qquad (2.1)$$

Доказательство: пусть схема δ разделима. Обозначим $l = max\ \{l_i\}$. Возведем левую часть неравенства (2.1) в n-ю степень и раскроем скобки:

$$\left(\sum_{i=1}^{n} 2^{-l_i}\right)^n = \sum_{(l_1 \dots i_n)} \left(2^{l_{i1}+\dots+l_{in}}\right)^{-1},$$

где $i_1, \dots, i_n$ – различные наборы элементарных кодов. Обозначим через $\mu(n, t)$ количество входящих в эту схему слагаемых вида 2^{-t}, где $t = i_{i1}, \dots, i_{in}$. Отметим, что для некоторых t может быть, что $\mu(n, t) = 0$.

Приводя подобные, получим сумму

$$\sum_{t=1}^{nl} \frac{\mu(n,t)}{2^t}.$$

Каждому слагаемому вида $(2^{l_{i1}+\dots+l_{in}})^{-1}$ можно однозначно сопоставить код вида $\beta_{i1}\dots\beta_{in}$. Это слово состоит из n элементарных кодов и имеет длину t.

Таким образом, $\mu(n,\ t)$ – это количество некоторых слов вида $\beta_{i1}\ldots\beta_{in}$, таких что $|\beta_{i1}\ldots\beta_{in}\ | = t$. В силу разделимости схемы $\mu(n,\ t) \le 2^t$, так как в противном случае заведомо существовали бы два одинаковых слова, $\beta_{i1}\ldots\beta_{in} = \beta_{j1}\ldots\beta_{jn}$ допускающих различные разложения.

Имея

$$\sum_{t=1}^{nl} \frac{\mu(n,t)}{2^t} \le \sum_{t=1}^{nl} \frac{2^t}{2^t} = nl,$$

следовательно, $\forall n (\sum_{i=1}^{n} 2^{\ l_i}) \le nl$, и, значит, $\sum_{t=1}^{n} 2^{\ l_i} \le \sqrt[n]{nl}$, откуда $\sum_{i=1}^{n} 2^{\ l_i} \le \lim_{n\to\infty} \sqrt[n]{ne} = 1$.

Пусть теперь числа l_1, ..., l_n удовлетворяют неравенству (2.1). Без ограничения общности можно считать, что $l_1 \le l_2 \le \ldots \le l_n$. Разобьем множество $\{l_1,\ldots, l_n\}$ на классы эквивалентности по отношению равенства $\{l_1, \ldots, l_m\}$, $m \le n$.

Пусть

$$\lambda_i \in l_i,\ \mu_i = |l_i|,\ \sum_{i=1}^{m} \mu_i = n,\ \lambda_1 < \lambda_2 < \ldots < \lambda_m.$$

Тогда неравенство (2.1) можно записать так:

$$\sum_{t=1}^{m} \frac{\mu_i}{2^{x_i}} \le 1.$$

Из этого неравенства следует m неравенств для частных сумм:

$$\frac{\mu_1}{2^{x_1}} \le 1 \Rightarrow \mu_1 \le 2^{\lambda_1}$$

$$\frac{\mu_1}{2^{x_1}} + \frac{\mu_2}{2^{x_2}} \le 1 \Rightarrow \mu_2 \le 2^{\lambda_2} \quad \mu_1 2^{\lambda_2 \quad \lambda_1}$$

....

$$\frac{\mu_1}{2^{x_1}} + \frac{\mu_2}{2^{x_2}} + \ldots + \frac{\mu_m}{2^{x_m}} \le 1 \Rightarrow \mu_m \le 2^{\lambda_m} \quad \mu_1 2^{\lambda_m \quad \lambda_2} \quad \ldots \quad \mu_{m\ 1} 2^{\lambda_m \quad \lambda_{m\ 1}}.$$

Рассмотрим слова длины λ_1 в алфавите B, поскольку $\mu_1 \le 2^{\lambda_1}$ из этих слов можно выбрать μ_1 различных слов $\beta_1,\ldots,\beta_{\mu 1}$ длины λ_1. Исключим из дальнейшего рассмотрения все слова из B^*, начинающихся со слов $\beta_1,\ldots,\beta_{\mu 1}$.

Теперь рассмотрим множество слов в алфавите B длиной λ_2 и не начинающихся со слов $\beta_1,\ldots,\beta_{\mu 1}$. Таких слов будет $2^{\lambda_2} - \mu_1 2^{\lambda_2-\lambda_1}$. Но $\mu_2 \le 2^{\lambda_2} - \mu_1 2^{\lambda_2-\lambda_1}$ и, значит, можно выбрать μ_2 различных слов. Обозначим их $\beta_{\mu 1+1},\ldots,\beta_{\mu 1+\mu 2}$. Исключим из дальнейшего рассмотрения все слова из B^*, начинающиеся с этих слов.

И так далее, используя неравенства для частных сумм, мы будем на i-ом шаге выбирать μ_i слов длины λ_1, $\beta_{\mu 1+\mu 2+\ldots+\mu i-1+1},\ldots,\ \beta_{\mu 1+\mu 2+\ldots+\mu i-1+\mu i}$, причем эти слова не будут начинаться с тех слов, которые были выбраны ранее. В то же время длины этих слов все время растут, так как $\lambda_1 < \lambda_2 < \ldots < \lambda_m$, и поэтому они не могут быть префиксами тех слов, которые выбраны ранее.

В конце этого процесса мы получим набор из n слов $\beta_1,\ldots,\beta_{\mu 1+\ldots+\mu m} = \beta_n$, где $|\beta_1| = l_1,\ldots,\ |\beta_n| = l_n$ и коды $\beta_1,\ldots,\beta_n$ не являются префиксами друг друга, а значит, схема $\delta = \langle \alpha_i \to \beta_i \rangle_{i=1}^{n}$ будет префиксной и, по лемме, разделимой, ч. и т.д.

Пример. Азбука Морзе – эта схема алфавитного кодирования.

⟨A→01, B→1000, C→1010, D→100, E→0, F→0010, G→110, H→0000, I→00, J→0111, K→101, L→0100, M→11, N→10, O→111, P→0110, Q→1101, R→010, S→000, T→1, V→001, W→011, X→1001, Y→1011, Z→1100⟩

Обычно 0 называют точкой, а 1 – тире. Имеем:

$$\frac{1}{4}+\frac{1}{16}+\frac{1}{16}+\frac{1}{8}+\frac{1}{2}+\frac{1}{16}+\frac{1}{8}+\frac{1}{16}+\frac{1}{4}+\frac{1}{16}+\frac{1}{8}+\frac{1}{16}+\frac{1}{4}+$$
$$+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\frac{1}{16}+\frac{1}{8}+\frac{1}{8}+\frac{1}{2}+\frac{1}{8}+\frac{1}{16}+\frac{1}{8}+\frac{1}{16}+\frac{1}{16}+$$
$$+\frac{1}{16}=3+\frac{5}{8}>1$$

Следовательно, неравенство Макмиллана для азбуки Морзе не выполнено, и эта схема не является разделимой. Поэтому в азбуке Морзе имеются дополнительные элементы – паузы между буквами и словами, которые и позволяют однозначно декодировать сообщения.

Отметим еще следующий интересный результат [Scutzenberger, 1967].

Теорема 2. Для того чтобы схема $\delta = \langle \alpha_i \to \beta_i \rangle_{i=1}^{n}$, $l_i = |\beta_i|$ обладала свойством самосинхронизации, то есть, чтобы ошибки при декодировании автоматически локализовались с вероятностью 1, необходимо и достаточно, чтобы выполнились два условия:

1) неравенство Макмиллана;
2) НОД (l_1, l_2, …, l_n) = 1.

2.3. Передача дискретных сообщений по каналу без шумов. Оптимальное кодирование

Задача оптимального кодирования – эффективная передача максимального количества информации в единицу времени.

Рассмотрим алфавит, состоящий из 0 и 1.

Пропускная способность – максимальное количество информации, которое может быть передано по каналу за единицу времени (1 – положительно, 0 – отрицательно). Передача 1 = 1 бит или 0 = 1 бит

Если максимальная длительность сигнала равна начальному времени (t_0 сек.), то число сигналов (импульсов) равно n и $n = \frac{1}{t_0}$, а пропускная способность каналов C, и $C = n = \frac{1}{t_0}\frac{\text{бит}}{\text{сек}}$.

Теорема (первая теорема Шеннона)

Если источник сообщения имеет энтропию $H\frac{\text{бит}}{\text{буква}}$, а канал обладает пропускной способностью $C\frac{\text{бит}}{\text{сек}}$, то всегда можно найти такой способ кодирования, который обеспечит передачу букв по каналу, со средней скоростью

$$V_{\text{ср}} = \left(\frac{C}{H} - E\right)\frac{\text{букв}}{\text{сек}},$$

где E – сколь угодно малая величина.

Обратное утверждение, доказанное Шенноном, гласит, что передача букв по каналу со средней скоростью больше, чем C/H невозможно. Отсюда следует, что

$$V_{\max} = \frac{C}{H}$$

Закодируем четыре символа следующим образом: $a_1 \to 0$, $a_2 \to 1$, $a_3 \to 01$, $a_4 \to 10$. Попробуем составить и закодировать слово: $a_2a_1a_2a_3$ – 10101.

Здесь не выполняется однозначность расшифровки. Можно получить последовательность $a_4a_2a_3$ или $a_2a_3a_3$.

Закодируем теперь четыре символа следующим образом: $a_1 \to 00$, $a_2 \to 01$, $a_3 \to 10$, $a_4 \to 11$. Попробуем составить и закодировать слово: $a_2a_1a_2a_3$ – 01000110. Эта последовательность раскодируется однозначно.

1. Предположим, что эти символы равновероятны и вероятности равны: $p_1 = p_2 = p_3 = p_4 = 0{,}25$. Вычислим

энтропию H и скорость V. Пропускная способность канала $C = 10000\ \frac{\text{бит}}{\text{сек}}$.

$$H = -\sum_{i=1}^{4} p_i \cdot \log p_i = -4 \cdot 0{,}25 \log 0{,}25 = 2\ \frac{\text{бит}}{\text{буква}}.$$

$$V_{max} = \frac{10000\ \frac{\text{бит}}{\text{сек}}}{2\ \frac{\text{бит}}{\text{буква}}} = 5000\ \frac{\text{букв}}{\text{сек}}.$$

2. Предположим теперь, что символы будут встречаться с разными вероятностями (см. табл. 6).

Таблица 6. Таблица вероятностей

Буква	α_1	α_2	α_3	α_4
Вероятность	0,125	0,5	0,125	0,25

$H=-(0{,}125 \log 0{,}125 + 0{,}5 \log 0{,}5 + 0{,}125 \log 0{,}125 + 0{,}25 \log 0{,}25) = 1{,}75\ \frac{\text{бит}}{\text{сек}}$

$$V_{max} = \frac{10000\ \frac{\text{бит}}{\text{сек}}}{1{,}75\ \frac{\text{бит}}{\text{буква}}} = 5714\ \frac{\text{букв}}{\text{сек}}.$$

Можно сделать вывод, что если при кодировании символы будут встречаться с разными вероятностями, то можно увеличить скорость их передачи по каналу.

Код Морзе, созданный в 1835 году, использовал эту методику (часто встречающийся символ кодировался более коротким кодом и наоборот).

Этот код является не оптимальным, так как для оптимальности необходимо два следующих условия:

1. Скорость передачи.
2. Однозначность расшифровки.

В коде Морзе выполняется только второе условие.

Пример: передается сообщение, содержащее N = 5 000 букв, при 2-х разрядном кодировании будет передано L = 10 000 символов. Однако, по теореме Шеннона, при оптимальном кодировании то же самое сообщение может быть передано с помощью меньшего количества знаков:

$$L_{опт} = N \cdot H_{опт} = 5000 \cdot 1{,}75 = 8750 \text{ символов.}$$

Отношение максимально возможной длины сообщения при оптимальном кодировании к фактической длине при неоптимальном кодировании называется коэффициентом сжатия.

$$K_{сж} = \frac{L_{опт}}{L} = \frac{H}{H_{max}}.$$

Например:

$$K_{сж} = \frac{8750}{10000} = \frac{1{,}75}{2} = 0{,}875.$$

Избыточность. В качестве количественной меры избыточности R при оптимальном кодировании принимают

$$R = 1 - K_{сж} = \frac{H}{H_{max}}.$$

Пример: Если коэффициент сжатия равен 0,875, то R = 1 – 0,875 = 0,125.

Метод Шеннона–Фено

Код строится следующим образом. После подсчета всех символов, встречающихся в файле, они вписываются в таблицу в порядке убывания частот их появления. На первом этапе она разбивается на две части так, чтобы суммы частот появления символов в них были приблизительно равны.

Таблица 7. Пример кодирования методом Шеннона-Фено

Буква	Вероятность	Цифры кода			Код
		1ая	2ая	3я	
α_2	0,5	0			0
α_4	0,25	1	0		10
α_1	0,125	1	1	0	110
α_3	0,125	1	1	1	111

Так, первое подмножество: α_2 – 0,5; второе: α_4 – 0,25; α_1–0,125; α_3 – 0,125 (в сумме 0,5). Для верхнего подмножества каждому символу приписывается 0, для нижнего – 1. Затем каждое из этих двух подмножеств делится пополам, и процедура повторяется для каждого из подмножеств. В конечном итоге в каждом подмножестве останется по одному символу.

Закодируем строку $\alpha_2\alpha_3\alpha_1\alpha_4$ полученным кодом – 011111010. Ее можно однозначно раскодировать.

В последовательности из N букв «1» встречается $L_1 = (1 \cdot 0{,}25 + 5 \cdot 0{,}125) \cdot N = 0{,}875\ N$ раз, «0» встречается $L_0 = (0{,}5 + 0{,}25 + 0{,}125) \cdot N = 0{,}875\ N$ раз. Общая длина сообщения: $L = L_0 + L_1 = 1{,}75\ N$ раз. Если $N = 5000$, то $L = 1{,}75 \cdot 5000 = 8750$ символов. То есть этот код является оптимальным за счет того, что часто встречающиеся символы кодируются более коротким кодом, а реже встречающиеся символы кодируются более длинным кодом.

Метод Хаффмена

Код строится следующим образом. Например, имеется строка символов «От топота копыт». Всего 15 букв. После подсчета всех символов, встречающихся в файле, они вписываются в таблицу в порядке убывания частот их появления. Каждый следующий столбец получаем сложением двух нижних чисел, и эту сумму помещаем на

соответствующее место, или ниже, если такое число есть в последовательности.

Таблица 8. Пример кодирования методом Хаффмена

Символ	6	5	4	3	2	1	1	2	3	4	5	6
т	4	4	4	4	7	8	0	1	00	00	00	00
о	4	4	4	4	4	7	1	00	01	01	01	01
п	2	2	3	4	4			01	10	11	100	100
« »	2	2	2	3					11	100	101	101
к	1	2	2							101	110	111
ы	1	1									111	1100
а	1											1101

В итоге на вершине должно остаться два числа. Их сумма должна быть равна общему количеству символов в строке. Верхнее число кодируется 0, нижнее кодируется 1. Подчеркнутое число выделяем. В каждом следующем столбце выделенный код записываем внизу дважды, к первому добавляем 0, ко второму – 1, остальные коды переписываем, сдвигая на одну позицию вверх.

В результате каждый из символов получает свой код (т – 00, о – 01, п – 100 и т. д.). При этом, чем чаще встречается символ в строке, тем он короче.

2.4. Сжатие данных

По мере развития компьютерных систем, увеличения объемов пользовательской информации и необходимости передачи ее по локальным и глобальным сетям появилась и потребность в сжатии различных данных.

В отличие от узкоспециализированных аппаратных средств сжатия, предназначенных чаще всего для онлайновой работы с одним конкретным типом данных,

программные системы сжатия данных используются в основном для экономии места на различных носителях информации и не пригодны для работы в онлайновом режиме.

Кратко коснемся терминологии. Принято различать архивацию и упаковку (компрессию, сжатие) данных. В первом случае речь идет о слиянии нескольких файлов и даже каталогов в единый файл – архив, во втором – о сокращении объема исходных файлов путем устранения избыточности. Как правило, современные программные архиваторы обеспечивают также сжатие данных, являясь еще и упаковщиками.

Все алгоритмы (методы) сжатия данных можно разделить на два больших типа – сжатие без потерь и с потерями. Первый тип алгоритмов используется в тех случаях, когда необходимо полное совпадение исходных данных и восстановленных после сжатия/распаковки (компрессии/декомпрессии), например, в случае двоичной информации, текстов и др. На алгоритмах этой группы основываются широко используемые программы-архиваторы и протоколы передачи данных в компьютерных сетях.

Второй тип алгоритмов сжатия используется в случаях, когда пользователю не требуется полного совпадения входной и обработанной информации. Как правило, такие алгоритмы используются для уменьшения избыточности файлов, содержащих изображения и звуки, и основываются на нечувствительности человеческих органов восприятия к небольшим искажениям в представляемой информации. Сжатие данных с потерей части информации обеспечивает наивысшую степень и ско-

рость сжатия данных. Примерами таких алгоритмов являются форматы *jpeg* и *mpeg*, обеспечивающие 20-кратную степень сжатия.

Теоретические основы сжатия информации были заложены в конце 40-х годов прошлого века Клодом Шенноном. Сформулировано положение о том, что энтропия любого блока информации равна вероятности его появления во всем массиве данных. Соответственно, наиболее часто повторяющиеся блоки являются и наиболее «избыточными» и могут быть представлены в более сжатом виде.

Общая формула Шеннона, позволяющая найти количество информации в случайном сообщении фиксированного алфавита, выглядит так:

$$H = p_1 \log_2(1/p_1) + p_2 \log_2(1/p_2) + \ldots + p_n \log_2(1/p_n),$$

где H – количество бит информации в одном символе сообщения, $p_1, \ldots, p_n$ – вероятности появления символов $x_1, \ldots, x_n$ в тексте сообщения.

На основании этой формулы рассчитывается значение, которое показывает, сколько бит необходимо для представления одного символа алфавита данного сообщения. В некоторых случаях алфавит сообщения неизвестен, тогда выдвигаются гипотезы о нем. Имея разные алфавиты, можно достичь разных значений степени сжатия информации.

Принцип работы архиваторов основан на поиске в файле избыточной информации и последующем ее кодировании (генерировании кода), чтобы получить минимальный объем хранимых данных. Целью процесса сжатия является получение более компактного выходного потока данных из некоторого изначально некомпактного входного потока с помощью соответствующего преобразо-

вания. В настоящее время известно множество различных алгоритмов сжатия информации, которые условно делятся на четыре группы (повторяющихся последовательностей, вероятностных, арифметических и словарных).

Все алгоритмы сжатия оперируют входным потоком информации, минимальной единицей которой является 1 бит. Основными техническими характеристиками процессов сжатия являются степень сжатия (отношение объемов исходного и результирующего потоков), скорость сжатия (время, затрачиваемое на сжатие некоторого объема информации) и качество сжатия (насколько сильно упакован выходной поток).

Метод повторяющихся последовательностей (*Run-Length Encoding – RLE*) является наиболее простым из известных алгоритмов сжатия информации и используется в основном для сжатия графических файлов. Классический вариант этого метода предусматривает замену последовательности повторяющихся символов на строку, содержащую сам этот символ, и число повторов этого символа.

Например, строка АААББББВВВВВГГ будет записана в виде: 3А4Б5В2Г. В рассмотренном примере - содержится 14 символов. Как известно, для представления одного символа требуется 1 байт, или 8 бит. Для представления всей последовательности требуется $14 \cdot 8 = 112$ бит, а после сжатия нужно лишь $8 \cdot 8 = 64$ бита, т. е. даже на таком маленьком массиве данных объем уменьшается почти вдвое. В то же время применение этого метода для сжатия текстовых или исполняемых (*exe*, *com*) файлов оказывается неэффективным, а общим недостатком RLE является достаточно низкая степень сжатия.

К вероятностным методам относятся алгоритмы Шеннона–Фено и Хаффмана, рассмотренные выше. В их основе лежит идея построения дерева, на ветвях которого положение каждого символа определяется частотой его появления в файле. Каждому символу присваивается код, длина которого обратно пропорциональна частоте появления символа в файле.

Метод Хаффмана очень похож на метод Шеннона–Фено, но создание дерева в нем начинается не с вершины, а с корня. Это метод сжатия, который уменьшает среднюю длину кода для символов.

Существуют две разновидности вероятностных методов сжатия данных, различающихся способом определения вероятности появления каждого символа в файле – статические и аддитивные. Первые используют таблицу частот появления символов в файле, рассчитываемую перед началом процесса сжатия. В аддитивных же методах частота появления символов меняется, и по мере считывания нового информационного блока из файла происходит перерасчет начальных значений частот.

Как и вероятностные алгоритмы, арифметические алгоритмы сжатия используют в качестве своей основы вероятность появления символа в файле, хотя сам процесс сжатия имеет принципиальные отличия. Арифметическое кодирование является методом, позволяющим упаковывать символы входного алфавита без потерь при условии, что известно распределение частот этих символов. В этих условиях оно является наиболее оптимальным.

При арифметическом кодировании текст представляется вещественными числами в интервале [0; 1]. По

мере кодирования текста, отображающий его интервал уменьшается, а количество бит для его представления возрастает. Последующие символы текста сокращают величину интервала исходя из значений их вероятностей. Более вероятные символы делают это в меньшей степени, чем менее вероятные, и, следовательно, добавляют меньше бит к результату.

В качестве примера рассмотрим кодирование двух символов **а** и **б** с вероятностями 0,666 и 0,333. Кодируя сообщение, для символа **а** выбираем интервал [0; 0,666], для символа **б** – [0,666; 1]. Для двух символов получим следующее: **аа** – [0; 0,444], **аб** – [0,444; 0,666], **ба** – [0,666; 0,888], для **бб** – [0,888; 1] и т.д.

Арифметическое кодирование позволяет обеспечить высокую степень сжатия, особенно при работе с данными, в которых частоты появления различных символов сильно отличаются друг от друга. В то же время сама процедура арифметического кодирования требует мощных вычислительных ресурсов.

Метод словарей. Этот алгоритм был впервые описан в работах Абрахама Лемпеля и Якоба Зива (двухступенчатое кодирование). На сегодняшний день этот алгоритм, известный как LZ-алгоритм, и его модификации получили наиболее широкое распространение по сравнению с другими алгоритмами сжатия. В его основе лежит идея замены наиболее часто встречающихся последовательностей символов (строк) в файле ссылками на образцы, хранящиеся в специально создаваемом словаре. Так, создав словарь, содержащий 65536 наиболее употребительных слов, можно представить текстовые файлы в виде последовательности 16-битовых ссылок на место данного слова в словаре.

Например, если программа сжатия уже имеет в словаре последовательность «абв» и обнаружит последовательность «абва», то она вначале занесет в выходной файл код из словаря для «абв», затем для символа «а», после чего последовательность «абва» будет добавлена в словарь. Если же она позже встретит «абваб», то выведет код для «абва», затем код для символа «б» и добавит «абва» в словарь. Когда программа встречает последовательность из словаря, она выдает код и добавляет новую запись, которая на один байт длиннее, то есть каждый раз при повторении последовательности словарь будет расти за счет включения продолжения этой последовательности.

На практике приходится оперировать таблицами, заполняемыми по мере сканирования файла. При этом уже просмотренная часть файла используется как словарь. Алгоритм основывается на движении по потоку данных скользящего окна, состоящего из двух частей: большей по объему, в которой содержатся уже обработанные данные, и меньшей, в которую по мере просмотра помещается вновь считанная информация. Во время считывания каждой новой порции данных происходит проверка, и если оказывается, что такая строка уже есть в словаре, то она заменяется ссылкой.

Алгоритм Лемпеля–Зива–Велча (Lempel–Ziv–Welch – LZW) разработан в 1974 году. Его отличают невысокие требования к памяти и более низкая степень сжатия по сравнению со схемой двухступенчатого кодирования. Этот алгоритм просматривает входной поток, разбивая его на подстроки и добавляя новые коды в конец словаря. Он преобразует поток символов на входе в поток индексов ячеек словаря на выходе.

Растровые изображения представляют собой двумерный массив чисел – пикселей, а изображения можно подразделить на две группы: с палитрой и без нее. У первых в пикселе хранится число – индекс в некотором одномерном векторе цветов, называемом палитрой (из 16 и 256 цветов).

Изображения без палитры бывают в какой-либо системе представления цвета и в градациях серого. При использовании некой системы представления цвета каждый пиксель является структурой, полями которой являются компоненты цвета (например, RGB и CMYK).

На заре компьютерной эры для сжатия графики применялись традиционные алгоритмы, рассмотренные выше. С появлением новых типов изображений эти алгоритмы утратили эффективность. Многие изображения практически не сжимались, хотя обладали явной избыточностью. Тогда и появились алгоритмы с потерей информации. Как правило, в них можно задавать коэффициент сжатия (т. е. степень потерь качества).

Таблица 9. Некоторые разновидности LZ-алгоритма

	Описание
LZ77	Алгоритм, в котором чередуются указатели и символы. Указатели ссылаются на подстроку, расположенную в предыдущих N символах.
LZFG	Алгоритм, в котором указатели выбирают узел в дереве поиска. Строки в дереве выбираются из текущего окна
LZH	Алгоритм, аналогичный LZSS, на втором этапе которого кодирование указателей осуществляется по методу Хаффмана
LZC	Алгоритм, реализованный программой compress в UNIX-системах

Окончание табл. 9

LZFG	Алгоритм, в котором указатели выбирают узел в дереве поиска. Строки в дереве выбираются из текущего окна
LZMW	Алгоритм, аналогичный LZT, но фразы строятся путем объединения предыдущих фраз
LZR	На выходе алгоритма чередуются указатели и символы
LZW	На выходе этого алгоритма имеются только указатели фиксированного объема, ссылающиеся на предварительно выделенные подстроки

Алгоритмы сжатия с потерями не рекомендуется использовать при сжатии изображений, которые затем будут предназначены для печати с высоким качеством или для обработки с помощью ПО распознавания образов.

2.5. Помехоустойчивое кодирование

Задача кодирования при наличии помех существенно отличаются от задачи кодирования при передаче по каналу без помех, так как важнейшую роль приобретает проблема минимизации влияния помех на достоверность передачи информации. Классификация помех и их источников представлена на рисунке 2.2.

Источники помех делятся на внешние и внутренние, помехи – на регулярные и случайные. Сложнее обнаруживать случайные помехи. Внешние источники помех, вызывают в основном импульсные помехи, внутренние флуктуационные помехи, накладываясь на видеосигнал, приводит к двум типам искажения: краевые и дробление. Флуктуационные помехи представляют собой отклонение от средних значений термодинамической параметров состояния системы. Краевые искажения связа-

ны со смещением переднего либо заднего фронта импульса. Дробление связано с дроблением единого видеосигнала на некоторое количество более коротких сигналов.

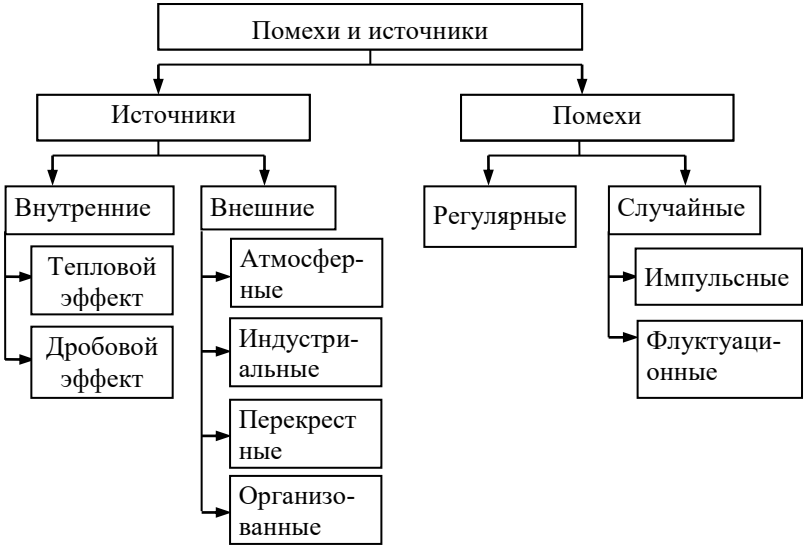


Рис 2.2. Помехи и их источники

По мере совершенствования электронных устройств возрастает их надежность, но, тем не менее, в реальной жизни возможны физические ошибки при работе. Часто говорят, что в канале связи имеют место помехи.

Достаточно очевидно, что для решения этой проблемы вместе с пересылаемой информацией мы будем вынуждены передавать какую-то дополнительную информацию, позволяющую найти и исправить (или только найти) ошибку (так называемые избыточные коды).

Для оценки кодов с точки зрения возможности обнаружения и исправления ошибок введем понятие кодового расстояния.

Кодовым расстоянием d, или числом кодовых переходов для двоичных кодов, называется число разрядов, которыми отличаются любые две кодовые комбинации.

Пример. Рассмотрим кодовые комбинации 00, 01, 10, 11. Кодовые расстояния между комбинациями 00 и 01, а также 10 и 11 будут равны единице (d = 1), так как эти пары кодовых комбинаций отличаются только своими вторыми элементами. Аналогично d = 1 для кодов 00 и 10, а также 01 и 11, которые отличаются только первыми элементами. А вот между кодами 00 и 11, а также 01 и 10 кодовое расстояние d = 2, так как коды этих пар отличаются друг от друга двумя элементами.

Очевидно, что минимальное кодовое расстояние, необходимое для различения двух кодов, $d_{\min} = 1$.

Обозначим через r и s соответственно количества обнаруживаемых и исправляемых ошибок. Легко видеть, что для обнаружения каждой единой ошибки в коде требуется один избыточный элемент, а для исправления одной ошибки – два избыточных элемента. Таким образом, в избыточных кодах имеет место соотношение:

$$d = 1 + r + s, \qquad (2.2)$$

причем $r \geq s$.

Теорема (вторая теорема Шеннона)

Пусть дискретный канал обладает пропускной способностью $C \frac{\text{бит}}{\text{сек}}$, а источник сообщения имеет энтропию $H \frac{\text{бит}}{\text{буква}}$. При $H <= C$ существует такая система кодирования, что информация от источника может быть передана по каналу с произвольно малой вероятностью ошибки.

При $H > C$ возможно так закодировать информацию, что ошибочность будет меньше $H - C + E$, где E – произ-

вольно мало. Не существует способа кодирования, дающего ошибочность меньше $H - C$. Свойства кодов по обнаружению и исправлению ошибок сведены в табл. 10.

Таблица 10. Свойства кодов

Характеристика кодов			Свойства кодов
d	r	s	
1	0	0	Отличает одну кодовую комбинацию от другой
2	1	0	Обнаруживает одну ошибку.
3	1	1	Обнаруживает и исправляет одну ошибку
	2	0	Обнаруживает две ошибки
4	2	1	Обнаруживает две ошибки и исправляет одну
	3	0	Обнаруживает три ошибки.
	2	2	Обнаруживает и исправляет две ошибки
	3	1	Обнаруживает три и исправляет одну ошибку
	4	0	Обнаруживает четыре ошибки и т.д.

Рассмотрим возможности использования избыточности в кодовых комбинациях для обнаружения и исправления ошибок на примере трехэлементных кодов. Выпишем все восемь возможных кодовых комбинаций: 000,001, 010, 011, 100, 101, 110, 111. Какую бы из этих комбинаций мы ни выбрали, на расстоянии одного кодового перехода от нее будут находиться еще три кодовые комбинации (например, если выбрать комбинацию 001, то на расстоянии одного кодового перехода от нее будут находиться комбинации 000, 011, 101). Поэтому, если все восемь комбинаций использовать как рабочие комбинации для кодирования восьми различных сообщений, то ошибка при передаче в одном элементе любой комбина-

ции, превратит ее в одну из трех других комбинаций, что в свою очередь, приведет к восприятию сообщения отличного от того, которое было передано.

Предположим теперь, что в качестве рабочих кодовых комбинаций выбрано реально только четыре из восьми возможных, а именно комбинации 010, 010, 100, 111. У этого трехмерного кода кодовое расстояние $d = 2$, так как любая комбинация может быть получена из любой комбинации только путем замены двух элементов. Если в процессе передачи возникает ошибка только в одном элементе любой комбинации, то она превращается в некоторую нерабочую комбинацию (одну из комбинаций: 001, 011, 101, 110), не соответствующую никакому сообщению. Таким образом, при приеме этой бессмысленной кодовой комбинации обнаружен факт появления одиночной ошибки, хотя и неизвестно, какой именно. Описанный код не позволяет выяснить, какая именно из переданных комбинаций была искажена, а значит, и исправить эту ошибку. Однако получение этого бессмысленного сообщения является сигналом для переспроса, для повторной передачи искаженного сообщения.

Выберем теперь в качестве рабочих комбинаций того же элементного кода только две из восьми возможных, например, комбинации 001 и 110, отличающиеся друг от друга в трех разрядах, то есть у этого кода $d = 3$.

Если в процессе передачи произойдет ошибка в одном или двух элементах, любой из двух рабочих комбинаций, то возникает одна из нерабочих комбинаций, не соответствующих никакому сообщению, то есть применение этого кода дает возможность обнаружить две ошибки ($r = 2$). Более того, если произойдет только одна ошибка,

что значительно более вероятно, чем сразу две ошибки, то с высокой степенью вероятности можно восстановить истинную рабочую комбинацию как более близкую к возникшей нерабочей. Например, если вместо рабочей комбинации 001 вследствие ошибки в одном элементе появится одна из нерабочих комбинаций 000, 011 или 101, то все эти комбинации будут находиться на расстоянии одного кодового перехода от истинной рабочей комбинации 001 и на расстоянии двух переходов от ложной рабочей комбинации 110.

Таким образом, при $d = 3$ согласно условию (1) имеем $r = 2$ и $s = 0$, или $r = 1$ и $s = 1$. Тогда, как было описано в первом случае, мы обнаруживаем две ошибки, а во втором, более вероятном, обнаруживаем и исправляем одну ошибку.

Одним из простейших методов обнаружения одиночных ошибок, получившим широкое распространение, является так называемая проверка на четность. Этот метод заключается в том, что к каждой кодовой комбинации добавляется для проверки один избыточный знак (0 или 1) так, чтобы общее количество единиц в каждой комбинации было четным. Пример построения кода с проверкой на четность иллюстрируется в таблице 11.

В первом столбце таблицы выписаны восемь возможных трехэлементных кодовых комбинаций. Второй столбец содержит знаки, которые должны быть приписаны, например, справа к этим комбинациям так, чтобы общее количество единиц в любой комбинации стало четным. В третьем столбце выписаны полученные таким образом избыточные комбинации, причем до расшифровки принятых комбинаций предварительно проводит-

ся их проверка на четность. При одиночной ошибке количество единиц в соответствующей комбинации станет нечетным, что дает возможность обнаружения ошибки и осуществление переспроса (в частности, и автоматического).

Таблица 11. Пример построения кода

Трехэлементные кодовые неизбыточные комбинации	Контрольный избыточный знак	Полные кодовые комбинации, обнаруживающие одиночную ошибку
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

Приведем еще пример так называемого двоичного корректирующего кода или, двоичного кода с защитой сдвоенными элементами. В этом случае количество элементов удваивается, причем к каждому 0 приписывается 1, а к 1 приписывается 0.

Если, например, число 22 закодировано своим двоичным представлением 10110, то двоичный корректирующий код этого числа имеет вид 1001101001.

Полученный корректирующий код содержит вдвое большее число элементов и позволяет обнаружить одиночную ошибку в каждом разряде. Для этого до расшифровки кодовой комбинации производится сложение по модулю 2 каждой пары знаков корректирующего кода,

соответствующей одному знаку исходного кода. Так как в каждой паре содержится одна единица и один нуль, то при отсутствии ошибки сумма по модулю 2 даст 1. Если же вследствие одиночного сбоя в какой-либо паре окажется 00 или 11, то сумма по модулю 2 будет 0, что является сигналом ошибки в данном разряде.

Классификация помехоустойчивых кодов

Построение помехоустойчевых кодов в основном связано с добавлением к исходной комбинации (k-символов) контрольных (r-символов).

Коды бывают блочные и непрерывные. При блочном кодировании передаваемые сообщения представляют собой последовательности отдельных блоков – кодовых комбинаций, которые кодируются знаками первичного алфавита.

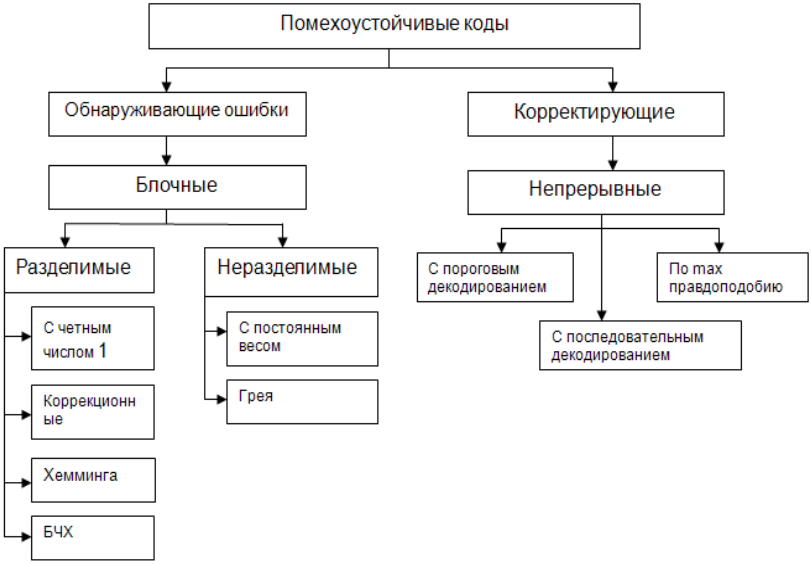


Рис. 2.3. Классификация помехоустойчивых кодов

Если все кодовые комбинации имеют одинаковую длину, код называется равномерным. При декодировании удобнее иметь дело с равномерным кодом.

Непрерывные (цепные коды) представляют собой непрерывную последовательность бит, не разделенную на блоки.

Разделимые коды – коды, в которых информационные и проверочные биты располагаются в строго определенных позициях (в неразделенных такой последовательности нет).

Блочные разделимые коды подразделяются на систематические и несистематические. В систематических кодах информационные и проверочные биты связаны между собой зависимостями описываемыми линейными уравнениями.

В несистематических кодах информационные проверочные биты либо вообще не имеют связи, либо эта связь нелинейная.

Наиболее часто в линиях связи используются систематические коды, к которым относятся циклические коды, коды Хемминга, матричные и ряд др.

Код Хемминга

Разработан в 1948 году. Позволяет построить оптимальный систематический код.

Оптимальный код – это код, который обеспечивает минимальную вероятность ошибочного декодирования среди всех иных кодов.

В коде Хемминга проверочные и информационные биты не разнесены в отдельные кодовые матрицы, а чередуются (см. рис. 2.4).

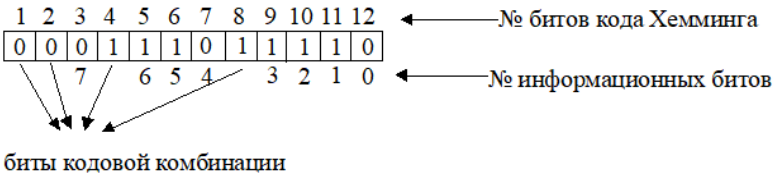


Рис. 2.4. Проверочные и информационные биты
в коде Хемминга

Так как информационная таблица содержит 8 бит, для исправления одной ошибки требуется включение в код 4 проверочных бит.

Проверочная таблица представлена на рисунке 2.5.

$$H_{12,4}=\begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{matrix} \\ \left[\begin{matrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{matrix}\right] \end{array} \begin{matrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{matrix}$$

Рис. 2.5. Проверочная таблица

2.6. Криптография

Рассмотрим основные понятия криптографии.

Шифрование – это кодирование данных с целью защитить от несанкционированного доступа (еще говорят, что сообщение зашифровано).

Обратный процесс, то есть переход от зашифрованного сообщения к общедоступному, называется расшифровыванием.

Основное требование к шифру состоит в том, чтобы расшифровка была возможна только при наличии санкции, то есть некоторой дополнительной информации (это

может быть и специальное устройство), которая называется ключом шифра. Естественно, что процесс шифрования также требует знания этого ключа.

Процесс декодирования без наличия ключа (то есть несанкционированное действие) называется дешифрованием.

Область знаний о шифрах, методах их создания и раскрытия называется криптографией (или тайнописью).

Свойство шифра противостоять раскрытию называется криптостойкостью или надежностью.

Криптография исторически зародилась из потребности передачи секретной информации. Длительное время она была связана только с разработкой специальных методов преобразования информации с целью ее представления в форме, недоступной для потенциального злоумышленника. С началом применением электронных способов передачи и обработки информации задачи криптографии начали расширятся.

В настоящее время, когда информационные и коммуникационные технологии нашли массовое применение, проблематика криптографии включает многочисленные задачи, которые не связаны непосредственно с засекречиванием информации. Современные проблемы криптографии включают разработку систем электронной цифровой подписи и тайного электронного голосования, протоколов электронной жеребьевки и идентификации указанных пользователей, методов защиты от навязывания ложных сообщений и др. Имеется большое количество различных публикаций по этим вопросам [1; 11; 12; 16].

Существует много различных классификаций шифров. Так как в этом небольшом параграфе мы излагаем только принципиальные подходы к решению задач криптографии (более подробную информацию можно найти, например, в указанных выше изданиях), то в качестве признака, по которому производится классификация шифров, используются тип ключа – закрытый или открытый.

2.6.1. Системы с закрытым ключом

Рассмотрим наиболее типичные примеры подобных шифров [1; 11].

Будем говорить, что сообщение обрабатывается блочным шифром, если открытый текст перед шифрованием разбивается на блоки, состоящие из нескольких знаков, каждый из которых обрабатывается в отдельности.

Мы будем рассматривать данные, находящиеся в компьютере, поэтому в дальнейшем предлагаются сообщения в алфавите {0, 1}. Тогда алфавитом, на котором действует блочный шифр, является множество двоичных векторов-блоков открытого текста одинаковой длины (64, 128 и т.д. бит)

К. Шеннон сформулировал общий принцип построения шифрующих преобразований – принцип «перемешивания»: если наборы открытого текста отличаются в незначительном числе позиций, то после шифрования этот результат должен быть сильно изменен. Так как шифрование – длительный процесс, то реализация этого принципа должна быть достаточно простой, что в общем случае сделать достаточно сложно. Поэтому К. Шеннон предложил реализовать сложные преобразования в виде

суперпозиции нескольких простых некоммутирующих отображений. Этот подход является магистральным в современном блочном шифровании.

В общем виде этот процесс можно описать так. Выделены так называемые базовые преобразования двух типов:

1. Перемешивающие – сложные в криптографическом отношении локальные преобразования над отдельными частями шифруемых блоков (усложняет восстановление взаимосвязи статистических и аналитических свойств открытого и шифрованного текстов).

2. Рассеивающие – простые преобразования, переставляющие между собой части шифруемых блоков (распространяет влияние одного знака открытого текста на большое число знаков шифротекста, что позволяет сгладить влияние статистических свойств открытого текста на свойства шифротекста).

Блочные шифры используют многократное повторение некоторого набора таких операций преобразований, называемые раундом шифрования. В каждом раунде используется некоторая часть ключа шифрования, называемая ***подключом***. Очередность выборки подключей из ключа шифрования называется ***расписанием*** использования ключа шифрования.

Раундовую криптосхему можно записать в виде рекуррентной формулы

$$B_i = E(B_{i-1}, K_i),$$

где E – раундовая функция шифрования; B_i, B_{i-1} – выходной и входной блоки для i-го раунда; K_i – подключ, используемый на i-ом раунде шифрования, $i = 1, 2, \ldots, N$;

N – число раундов преобразования. В таких шифрах раундовая функция должна быть обратима.

Пусть D – раундовая функция дешифрования. Тогда процедура дешифрования описывается формулой:

$$B_i = D(B_{i-1}, K_{N-i+1}).$$

Для разработки блочных шифров широко используется криптосхема Х. Фейстеля, позволяющая использовать произвольные (в том числе и необратимые) функции Е для построения обратимых шифрующих преобразований. Идея этой криптосхемы состоит в следующем.

Входной блок B разбивается на два одинаковых подблока L (левый подблок) и R (правый подблок), затем шифрование выполняется в соответствии со следующими рекурентными соотношениями:

$$L_i = R_{i-1},$$

$$R_i = L_{i-1} \otimes E(R_{i-1}, K_i),$$

где $L_0 | R_0$ – исходный блок данных; $L_N | R_N$ – преобразованный блок данных на выходе последнего раунда шифрования; $\oplus$ – операция поразрядного суммирования по модулю два; $i = 1, 2, \ldots, N$.

Важность преобразований Х. Фейстеля состоит в том, что даже если Е не является обратимой функцией, дешифрование все равно возможно и заключается в выполнении N следующий раундов:

$$R_i = L_{i-1},$$

$$L_i = R_{i-1} \otimes E(L_{i-1}, K_{N-i+1}),$$

Эти схемы шифрования и дешифрования показаны на рисунке 2.5.

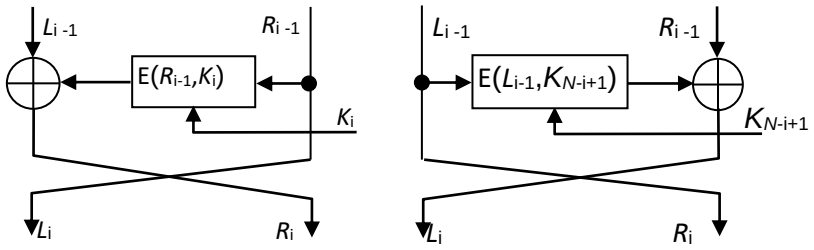


Рис. 2.5. Схемы шифрования и дешифрования

По подобной схеме построен американский стандарт шифрования данных DES (*Data Encryption Standard*), опубликованный Национальным бюро стандартов США в 1977 г. и принятый в 1980 г. в качестве стандарта шифрования данных для защиты от несанкционированного доступа к важной, но несекретной информации в государственных и коммерческих организациях США и российский стандарт шифрования данных ГОСТ 28147-89.

2.6.2. Системы с открытым ключем

В 1976 г. У. Диффи и М.Э. Хеллман опубликовали статью, которая ознаменовала собой рождение двухключевой криптографии и привела к сильному росту числа открытых исследований в области криптографии. Ее основной ошеломляющий результат: возможно построение практически стойких секретных систем, который не потребуют передачи секретного ключа.

Они ввели понятие односторонней функции с потайным ходом (лазейкой).

Под односторонней функцией *f(x)* понимается функция, которая легко вычислима для любого значения аргумента *x* из области определения, однако для данного у

из области ее значений вычислительно сложно нахождение значения аргумента x, для которого $f(x) = y$.

В 1978 г. Р. Ривест, А. Шамир и Л. Адлеман предложили пример функции f для шифрования. На ее основе была построена реально используемая система шифрования, получившая название по первым буквам имен авторов – система RSA.

Эта функция такова, что

a) существует достаточно быстрый алгоритм вычисления значений $f(x)$;

b) существует достаточно быстрый алгоритм вычисления значений обратной функции $f^{-1}(x)$;

c) функции $f(x)$ обладает некоторым «секретом», знание которого позволяет быстро вычислить значение $f^{-1}(x)$; в противном случае вычисление $f^{-1}(x)$ становится трудно разрешимой в вычислительном отношении задачей.

Изложим сначала общую концепцию криптосистемы RSA.

В качестве числа M будем брать исходное сообщение, которое необходимо подписать или зашифровать. Условие взаимной простоты чисел M и n обеспечим тем, что будем выбирать число n, равное произведению двух больших простых сомножителей. В этом случае вероятность того, что случайное сообщение не будет взаимно простым с n, является пренебрежимо малым. Возьмем $f : x \to x^e (\mathrm{mod}\ n)$, то есть будем иметь функцию шифрования $E : C = \mathrm{E}(M) = M^e (\mathrm{mod}\ n)$.

Для дешифрования необходимо выбрать значение степени d, такое, чтобы функция дешифрования $D(C) = C^d (\mathrm{mod}\ n)$ была обратной по отношению к $E(M) = M^e (\mathrm{mod}\ n)$, т.е. должно выполняться условие:

$$M = \mathrm{D}(\mathrm{E}(M) = (M^e)^d\ (\mathrm{mod}\ n) = M^{ed}\ (\mathrm{mod}\ n).$$

Из этого соотношения следует, что $e \cdot d \equiv 1(\text{mod } \varphi(n))$. Таким образом, две процедуры возведения в степень по модулю n будут взаимно обратными, если произведение степеней равно единице по модулю функции Эйлера от числа n. Параметр d является ключом к потайному коду, поэтому он является секретным.

Теперь задача состоит в выборе необходимых степеней e и d. Очевидно, что первоначально необходимо установить значение функции Эйлера от модуля n. Известно, что для любого простого числа p имеем $\varphi(p) = p - 1$. Поскольку мы выбираем $n = p \cdot q$, где p, q – простые, то $\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p - 1) \cdot (q - 1)$.

Как мы знаем, если целые числа e и n удовлетворяют условиям $0 < e < n$ и НОД$(e, \varphi(n)) = 1$, то существует единственное d, удовлетворяющее условиям $0 < d < n$ и $d \cdot e \equiv 1(\text{mod } \varphi(n))$ и, кроме того, d может быть вычислено с помощью расширенного алгоритма Евклида.

Функция f в системе RSA может быть вычислена достаточно быстро. Обратная к ней функция вычисляется по тем же правилам, лишь с заменой показателя степени e на d. Таким образом, функция $f(x)$ удовлетворяет условиям a), b).

Для вычисления функции f достаточно знать лишь числа e и n. Именно они составляют открытый ключ для шифрования. А вот для вычисления обратной функции требуется знать число d, оно и является «секретом», о котором шла речь в пункте c). Казалось бы, ничего не стоит, зная число n:

1. Разложить его на простые множители.
2. Вычислить по известным правилам значение $\varphi(n)$.
3. С помощью условия $d \cdot e \equiv 1(\text{mod } \varphi(n))$ определить d.

Все шаги этого вычисления могут быть реализованы достаточно быстро, за исключением первого. Именно разложение числа n на простые множители и составляет наиболее трудоемкую часть вычислений, о чем мы уже писали. В теории чисел несмотря на многолетнюю ее историю и на очень интенсивные поиски в течение последних 20 лет, эффективный алгоритм разложения натуральных чисел на множители так и не найден. Конечно, можно, перебирая все простые числа до $\sqrt{n}$ и деля на них n, найти требуемое разложение. Но, учитывая, что количество простых чисел в этом промежутке асимптотически равно $2\sqrt{n}\cdot(\ln n)^{(-1)}$, находим, что при n, записываемом 100 десятичными цифрами, найдется не менее $4\cdot10^{42}$ простых чисел, на которые придется делить n при разложении его на множители. Очень грубые прикидки показывают, что компьютеру, выполняющему миллион делений в секунду, для разложения числа $n > 10^{99}$ таким способом на простые сомножители потребуется не менее, чем 10^{35} лет. Другие, более эффективные методы разложения целых чисел на простые множители тоже очень медленны.

Общая схема функционирования криптосистемы RSA может быть описана так:

1. Выбираем два больших, не равных между собой простых числа p и q, находим $n = p \cdot q$, вычисляем $\varphi(n) = (p - 1) \cdot (q - 1)$. (Рекомендуется, чтобы длина n составляла 1024 бита).

2. Выбираем целое число e, чтобы $e < \varphi(n)$, НОД(e, $\varphi(n)$) = 1 и вычисляется d, удовлетворяющее условию $e \cdot d \equiv 1 (\mathrm{mod}\ \varphi(n))$.

3. Секретный ключ: p, q, d (на самом деле, только d, т.к. p и q после получения n и d могут быть уничтожены).

4. Пара чисел n, e – открытый ключ – предоставляется всем абонентам криптосистемы RSA.

5. Процедура подписывания сообщения M – это возведение числа M в степень d по модулю n: $S = M^d \pmod n$. Число S есть цифровая подпись, которую может выработать только владелец секретного ключа.

6. Процедура проверки подписи S, соответствующей сообщению M, – это возведение числа S в целую степень e по модулю n: $M' = S^e \pmod n$.

Если $M' = M$, то сообщение M признается подписанным пользователем, который составил ранее открытый ключ e. Следовательно, составить криптограмму, соответствующую данному открытому ключу и данному сообщению можно только по известному секретному ключу d.

Таким образом, секретный ключ служит для подписывания сообщений, открытый – для проверки подписи.

Легко построить и систему шифрованной переписки в RSA.

Пример: зашифруем аббревиатуру RSA, используя $p = 17$, $q = 31$. Для этого вычислим $n = p \cdot q = 527$ и $\varphi(n) = (p - 1) \cdot (q - 1) = 480$. Выберем в качестве e число, взаимно простое с $\varphi(n)$, например, $e = 7$. С помощью расширенного алгоритма Евклида найдем целые числа u и v. Получаем: $u = 2$, $v = -137$. Так как $-137 \equiv 343 \pmod{480}$, то $d = 343$.

Теперь представим данное сообщение в виде последовательности чисел из [0, 526]. Для этого буквы R, S, A закодируем пятимерными двоичными векторами, воспользовавшись двоичной записью их порядковых номеров

в английском алфавите: R соответствует 18 = $(10010)_2$, S соответствует 19 = $(10011)_2$, A соответствует 1 = $(00001)_2$. Тогда RSA в двоичном представлении RSA – 100101001100001. Укладываясь в заданный отрезок [0, 526], получим два двоичных числа: (100101001), (100001). То есть, $M_1 = 297$, $M_2 = 33$.

Далее последовательно шифруем M_1 и M_2:

$$C_1 = M_1^e \pmod{n} = 297^7 \pmod{527} = 474;$$

$$C_2 = M_2^e \pmod{n} = 33^7 \pmod{527} = 407;$$

В итоге получаем шифротекст: $C_1 = 474$, $C_2 = 407$.

Произведем расшифрование. Для этого вычислим D_1 и D_2.

$$D_1 = C_1^d \pmod{n} = 474^{343} \pmod{527} = 297;$$

$$D_1 = C_2^d \pmod{n} = 407^{343} \pmod{527} = 33.$$

Возвращаясь к буквенной записи, получаем после расшифрования RSA.

Проанализируем вопрос о стойкости системы RSA.

Как мы уже видели, сложность нахождения секретного ключа системы RSA определяется сложностью разложения числа n на простые множители. В связи с этим нужно выбирать числа p и q таким образом, чтобы задача разложения числа n была достаточно сложна в вычислительном плане. Для этого рекомендуются следующие требования:

1) числа p и q должны быть достаточно большими, не слишком сильно отличаться друг от друга и в то же время быть не слишком близкими друг другу;

2) числа p и q должны быть такими, чтобы наибольший общий делитель чисел $p - 1$ и $q - 1$ был небольшим; желательно, чтобы НОД $(p - 1, q - 1) = 2$;

3) p и q должны быть сильно простыми числами (сильно простым называется такое простое число r, что $r + 1$ имеет большой простой делитель, $r - 1$ имеет большой простой делитель s такой, что число $s - 1$ также обладает достаточно большим простым делителем).

В случае, когда не выполнено хотя бы одно из указанных условий, имеются эффективные алгоритмы разложения n на простые множители [16; 12].

В настоящее время самые большие простые числа, вида $n = p \cdot q$, которые удается разложить на множители известными методами, содержат в своей записи 140 десятичных знаков. Поэтому, согласно указанным рекомендациям, числа p и q в системе RSA должны содержать не менее 100 десятичных знаков.

Следует подчеркнуть необходимость соблюдения осторожности в выборе модуля RSA (числа n) для каждого из корреспондентов сети. В связи с этим можно сказать следующее. Зная одну из трех величин: p, q или $\varphi(n)$, можно легко найти секретный ключ RSA. Известно также, что, зная секретную экспоненту расшифрования d, можно легко разложить модуль n на множители. В этом случае удается построить вероятностный алгоритм разложения n. Отсюда следует, что каждый корреспондент сети, в которой для шифрования используется система RSA, должен иметь свой уникальный модуль.

В самом деле, если в сети используется единый для всех модуль n, то такая организация связи не обеспечивает конфиденциальности, несмотря на то, что базовая система RSA может быть стойкой. Выражаясь другими словами, говорят о несостоятельности протокола с общим модулем. Несостоятельность следует из того, что знание

произвольной пары экспонент (e_1, d_1) позволяет, как было отмечено, разложить n на множители. Поэтому любой корреспондент данной сети имеет возможность найти секретный ключ любого другого корреспондента. Более того, это можно сделать даже без разложения n на множители [16].

К сожалению, системы шифрования с открытыми ключами работают сравнительно медленно. Для повышения скорости шифрования RSA на практике используют малую экспоненту зашифрования.

До последнего времени наиболее совершенным устройством для быстрого разложения чисел на множители была оптоэлектронная система TWINKLE. Устройство TWIRL, разработанное Шамиром и Тромером, использует близкий к TWINKLE подход к реализации алгоритма разложения чисел на множители.

2.6.3 Классификация алгоритмов шифрования

Симметричные алгоритмы шифрования основаны на том, что отправитель и получатель информации используют один и тот же ключ. Этот ключ должен храниться в тайне и передаваться способом, исключающим его перехват.

Обмен информацией осуществляется в три этапа:

- отправитель передает получателю ключ (в сети с несколькими абонентами у каждой пары абонентов должен быть свой ключ, отличный от ключей других пар);
- отправитель, используя ключ, зашифровывает сообщение, которое пересылается получателю;
- получатель получает сообщение и расшифровывает его.

Если для каждого дня и для каждого сеанса связи будет использоваться уникальный ключ, это повысит защищенность системы.

Симметричные алгоритмы шифрования делятся на:

– потоковые (с одноразовым или бесконечным ключом (*infinite-key cipher*), с конечным ключом (система Вернама), на основе генератора псевдослучайных чисел (ГПСЧ);

– блочные (шифры перестановки (permutation), подстановки (*substitution*): моноалфавитные (код Цезаря), полиалфавитные (шифр Видженера, цилиндр Джефферсона, диск Уэтстоуна, *Enigma*));

– составные (*Lucipher* (фирма IBM, США), DES (*Data Encryption Standard*, США), FEAL-1 (*Fast Enciphering Algorithm*, Япония), IDEA/IPES (*International Data Encryption Algorithm*), *Improved Proposed Encryption Standard*, фирма *Ascom-Tech* AG, Швейцария), *B-Crypt* (фирма *British Telecom*, Великобритания), ГОСТ 28147-89 (СССР), *Skipjack* (США).

В *асимметричных* алгоритмах шифрования (или криптографии с открытым ключом) для зашифровывания информации используют один ключ (открытый), а для расшифровывания – другой (секретный). Эти ключи различны и не могут быть получены один из другого.

Схема обмена информацией следующая:

● получатель вычисляет открытый и секретный ключи, секретный ключ хранит в тайне, открытый же делает доступным (сообщает отправителю, группе пользователей сети, публикует);

● отправитель, используя открытый ключ получателя, зашифровывает сообщение, которое пересылается получателю;

• получатель получает сообщение и расшифровывает его, используя свой секретный ключ.

Асимметричные (с открытым ключом, *public-key*):

• Диффи–Хеллман DH (*Diffie*, *Hellman*);

• Райвест–Шамир–Адлеман RSA (*Rivest*, *Shamir*, *Adleman*);

• Эль-Гамаль (*ElGamal*).

В *асимметричных* системах необходимо применять длинные ключи (512 битов и больше). Длинный ключ резко увеличивает время шифрования. Кроме того, процедура генерации ключей весьма продолжительная. Зато распределять ключи можно по незащищенным каналам.

В *симметричных* алгоритмах используют более короткие ключи, т.е. шифрование происходит быстрее. Но распределять ключи в таких системах сложнее, поэтому при проектировании защищенной системы часто применяют и симметричные, и асимметричные алгоритмы. Система с открытыми ключами позволяет распределять ключи и в симметричных системах, поэтому в системе передачи защищенной информации можно объединить асимметричный и симметричный алгоритмы шифрования. С помощью первого рассылать ключи, вторым – собственно шифровать передаваемую информацию.

В правительственных и военных системах связи используют только симметричные алгоритмы, так как строго математического обоснования стойкости систем с открытыми ключами пока нет, как, впрочем, не доказано и обратное.

При передаче информации должны быть обеспечены одновременно или по отдельности:

• конфиденциальность (*privacy*) – злоумышленник не должен иметь возможности узнать содержание передаваемого сообщения;

• подлинность (*authenticity*), включающая два понятия: целостность (*integrity*) – сообщение должно быть защищено от случайного или умышленного изменения;

• идентификация отправителя (проверка авторства) – получатель должен иметь возможность проверить, кем отправлено сообщение.

Шифрование может обеспечить конфиденциальность, а в некоторых системах и целостность. Сейчас широко применяется цифровая подпись (цифровое дополнение к передаваемой информации, гарантирующее целостность последней и позволяющее проверить ее авторство). Известны модели цифровой подписи (*digital signature*) на основе алгоритмов симметричного шифрования, но при использовании систем с открытыми ключами цифровая подпись осуществляется более удобно.

Кроме того, существует разделение алгоритмов шифрования на собственно шифры (*ciphers*) и коды (*codes*). Шифры работают с отдельными битами, буквами, символами. Коды оперируют лингвистическими элементами (слоги, слова, фразы).

Первым систематическим трудом по криптографии принято считать работу великого архитектора Леона Баттиста Альберти (1404–1472). Период до середины XVII в. уже насыщен работами по криптографии и криптоанализу. Интриги вокруг шифрограмм в Европе того времени удивительно интересны. Еще одна известная фамилия – Франсуа Виет (1540–1603), который при дворе короля Франции Генриха IV столь успешно занимал-

ся криптоанализом (в те времена еще не носившим этого гордого названия), что испанский король Филипп II жаловался Папе Римскому на то, что французы прибегают к черной магии. Но все обошлось без кровопролития – при дворе Папы в то время уже служили советники из семейства Ардженти, которых сегодня назвали бы крипто-аналитиками.

Контрольные вопросы

1. В чем состоит задача оптимального кодирования?
2. Дать определение пропускной способности канала.
3. Сформулировать первую теорему Шеннона.
4. Какие условия необходимы для осуществления оптимального кодирования?
5. Как решается вопрос выбора кода?
6. Дать определение коэффициента сжатия.
7. Какой код считается оптимальным?
8. В чем заключается метод Шеннона–Фено?
9. Как доказать, что код составлен оптимально? Привести пример.
10. Кем заложены теоретические основы сжатия информации?
11. Что лежит в основе алгоритма Лемпеля–Зива?
12. В чем заключается метод повторяющихся последовательностей?
13. В основе какого метода лежит идея замены часто встречающихся последовательностей символов в файле ссылками на образцы?
14. Какая система счисления должна использоваться в компьютере с целью минимизации количества элементов в устройствах хранения?

15. Какая схема кодирования называется разделимой?
16. Является ли префиксная схема кодирования разделимой?
17. Какие методы относятся к вероятностным методам?
18. Перечислить единицы измерения пропускной способности канала, энтропии, скорости.
19. Какие помехи вызывают внешние источники помех?
20. Что такое «кодовое расстояние»?
21. Чему равно кодовое расстояние между двумя кодовыми комбинациями 011 и 100?
22. Является ли разделимая схема префиксной?
23. Какие искажения называются краевыми? Дробления?
24. Задача оптимальное кодирование. Метод Шеннона–Фено.
25. В чем суть метода Хаффмена?
26. В чем состоит процесс шифрования?
27. Чем занимается область знаний «криптография»?
28. В чем заключается принцип перемешивания Шеннона?
29. В чем отличия между рассеивающими преобразованиями и перемешивающими при шифровании информации?
30. Какой параметр в криптосистеме RSA является секретным?
31. Как может быть вычислен секретный ключ в RSA?
32. Какие системы используются для разложения чисел на множители?

ГЛАВА III. АВТОМАТЫ

3.1. Определение автомата

Автомат (lat. самодвижущийся) – техническое устройство, работающее без непосредственного участия человека.

Пример: банкоматы, турникеты, игровые автоматы.

В кибернетике в слово «автомат» вкладывается более узкое понятие так называемого синхронного дискретного устройства, обладающего следующими свойствами.

Такой автомат характеризуется:

– множеством входных сигналов $X = \{x_1, x_2, \ldots, x_n\}$;

– множеством выходных сигналов $Y = \{y_1, y_2, \ldots, y_n\}$;

– множеством состояний $Z = \{z_1, z_2, \ldots, z_s\}$.

Изменение сигналов происходит только на границах тактов безразмерного времени Δt.

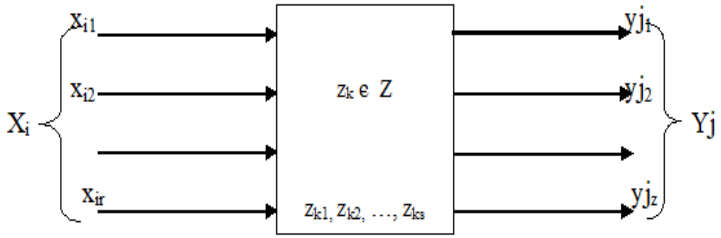


Рис. 3.1. Пример автомата

Организация работы элементов памяти, при которой моменты их возможных переключений следуют друг за другом через одни и те же фиксированные промежутки времени называется тактом. Импульсы вырабатываются с помощью тактового синхронизатора.

Тактовая частота – количество тактовых импульсов, выдаваемых в течение 1 сек. (Гц, КГц, МГц, ГГц).

В каждый момент времени модель может находиться в одном из множества состояний $Z = \{z_1, z_2, \ldots, z_s\}$. Таким образом, автомат можно представить в виде некоторого блока.

Функционирование автомата характеризуется его функцией переходов φ: $X \times Z \rightarrow Z$, определяющей изменение состояний под воздействием входных сигналов; и функцией выходов ψ: $X \times Z \rightarrow Y$, определяющей изменения выходов под влиянием входных сигналов.

Автомат, у которого множества X, Y, Z конечные, называется конечным автоматом.

Различают автоматы с памятью и без памяти.

Автомат без памяти – конечный автомат, имеющий только одно внутреннее состояние.

Пример: автомат размена денег, автомат газированной воды.

Автоматы с памятью также называют цифровыми автоматами.

Работа цифрового автомата описывается уравнениями:

$$z(t) = \varphi\,[x(t), z(t-1)]; \qquad (3.1)$$

$$y(t) = \psi\,[x(t), z(t-1)], \qquad (3.2)$$

где $x(t)$ – состояние входа в момент t дискретного времени;

$z(t)$ – состояние автомата в момент t;

$z(t-1)$ – состояние автомата в предыдущий момент времени $t-1$;

$y(t)$ – состояние выхода в момент t;

φ, ψ – функции переходов и выходов.

Таблица 12. Таблица переходов

Φ	x_1	x_2
z_1	z_1	z_2
z_2	z_2	z_3
z_3	z_3	z_1

Пример: поведение автомата с 2 состояниями входа x_1, x_2 и двумя состояниями выхода y_1, y_2, 3 внутренними состояниями z_1, z_2, z_3. задано таблицами переходов и выходов (см. таблица 12 и таблица 13).

Таблица 13. Таблица выходов

Ψ	x_1	x_2
z_1	y_1	y_1
z_2	y_1	y_2
z_3	y_2	y_1

Поведение автоматов может быть описано графом, где вершины 3 состояния:

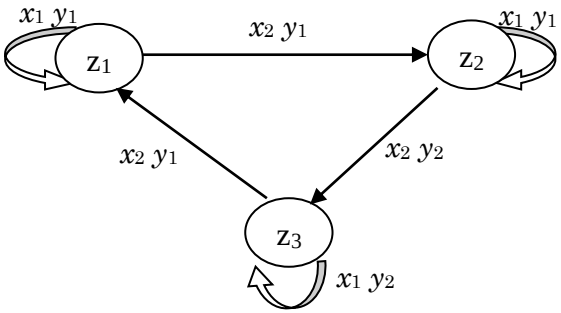


Рис. 3.2. Диаграмма, описывающая поведение автомата

Все это относится к детерминированным автоматам, функции переходов которых являются определенными

однозначными функциями множества состояний и входного алфавита.

Существуют также вероятностные автоматы, которые представляют собой дискретные, потактные преобразователи информации с памятью, функционирование которых описывается статистически.

3.2. Синтез логических и цифровых автоматов

Задача синтеза автоматов может быть определена как процесс построения структур автоматов, реализующих некоторые заданные условия работы. При этом, как и при проектировании любой машины, здания, системы, может быть создано бесчисленное множество различных вариантов, отличающихся друг от друга количеством элементов, стоимостью, быстродействием, габаритами и рядом других характеристик. Поэтому в задачу синтеза входит и оптимизация структуры автомата по тому или иному критерию, чаще всего по количеству элементов, а значит и стоимости, быстродействию и надежности.

Наиболее просто решается задача синтеза автомата без памяти, то есть логических или комбинационных автоматов (схем). При этом успешно исполняется аппарат математической логики или булевой алгебры.

В качестве примера рассмотрим процедуру синтеза логической схемы одноразрядного сумматора:

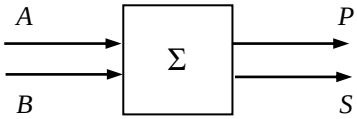


Рис. 3.3. Пример схемы одноразрядного двоичного сумматора

Таблица 14. Логическая схема одноразрядного сумматора

Вход		Выход	
A	B	S	P
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

$$S = (A \,\&\, \neg B) \vee (\neg A \,\&\, B) \qquad (3.3)$$

$$P = A \,\&\, B \qquad (3.4)$$

Этапы синтеза цифрового автомата

I этап: точная математическая формулировка заданных условий работы устройства. Эта формулировка может быть успешно реализована в виде так называемой логической таблицы.

Логическая схема для суммирования двух двоичных одноразрядных чисел A и B, которые могут принимать только 2 значения «0» или «1». В следующей схеме S – сумма, P – перенос (см. табл. 14).

Пользуясь этими формулами, можно синтезировать заданную логическую схему сумматора. Для составления этой схемы необходимы два «не», три «и» и один «или»

II этап: синтез логической схемы сумматора.

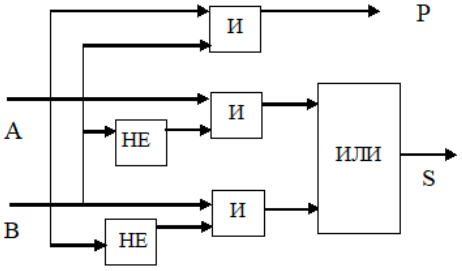


Рис. 3.4. Схема сумматора

III этап: оптимизация схемы по критерию минимума оборудования. Этого можно достичь упрощением логических уравнений. Перепишем уравнение (3), прибавив к его правой части тождественно равные нулю выражения (A & ¬A) и (B & ¬B), и преобразуем все выражения, вынося за скобки логические переменные в соответствие с распределенным законом:

$$S = (A \,\&\, \neg B) \vee (\neg A \,\&\, B) \vee (A \,\&\, \neg A) \vee (B \,\&\, \neg B) =$$
$$= A \,\&\, (\neg A \vee \neg B) \vee B \,\&\, (\neg A \vee \neg B) =$$
$$= (\neg A \vee \neg B) \,\&\, (A \vee B) = \neg (A \,\&\, B) \,\&\, (A \vee B)$$

$$S = \neg (A \,\&\, B) \,\&\, (A \vee B) \qquad (3.5)$$

$$P = A \,\&\, B \qquad (3.6)$$

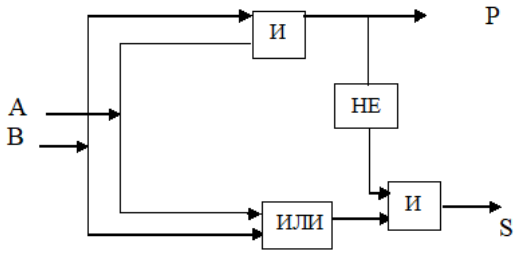


Рис. 3.5. Логическая схема сумматора минимизированная

Вывод: из-за уменьшения количества элементов увеличится быстродействие и, соответственно, стоимость схемы будет ниже.

Значительно сложнее проблема синтеза цифровых автоматов (автоматов с памятью), так как выходные сигналы цифровых автоматов зависят не только от входного сигнала, но и от его состояния, определяемого предшествующими входными сигналами. Схема цифрового автомата должна содержать наряду с логическими элементами, преобразующими информацию, также и элементы,

предназначенные для накопления и хранения информации.

В качестве таких элементов могут использоваться либо специальные запоминающие элементы (триггеры), либо так называемые элементы задержки.

Рассмотрим очень простую схему цифрового автомата, предназначенную для последовательного (поразрядного) сложения многоразрядных двоичных чисел.

3.3. Логическая схема многоразрядного сумматора

Эта схема содержит два одноразрядных сумматора и элемент задержки 3, задерживающий проходящие через него импульсы на один такт. Одноразрядные сумматоры Σ_1, Σ_2 являются как бы частями описываемого многоразрядного сумматора, называются полусумматорами. Оба полусумматора совершенно одинаковы, каждый из них имеет 2 входа для ввода слагаемых и 2 выхода для суммы и переноса.

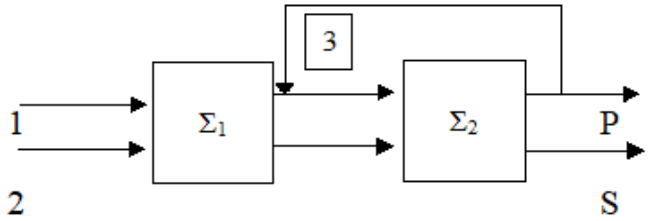


Рис. 3.6. Схема многоразрядного сумматора

Здесь Σ_1, Σ_2 – полусумматоры, 3 – элемент задержки.

Полусумматоры можно собрать по любой схеме, обеспечивая реализацию логической таблицы (см. выше). Действия схемы проследим на примере сложения чисел 7 (в двоичной системе счисления – 0111) и 14 (1110) = 21 (10101).

Числа в виде последовательности импульсов подаются на соответствующие входы Σ_1 последовательно разряд за разрядом, начиная с младшего разряда. Рассмотрим состояние схемы в различные моменты. В первый такт на входе Σ_1 поступает импульс (1) на его входе 2 импульс отсутствует (0). При этом на выходе S первого полусумматора получается импульс, который поставляет на вход 2 Σ_2, в результате второй полусумматор дает на выходе S импульс; во втором такте импульс есть на обоих входах Σ_1 и первый полусумматор выдает импульс на выходе P. Этот импульс проходит через элемент задержки 3 и попадает на вход 1 Σ_2 лишь в третьем такте. Следовательно, во втором такте на обоих входах Σ_2 импульсы отсутствуют, и на его выходе S импульса также не будет, в третьем такте на оба входа полусумматора Σ_2 снова поступают импульсы, дающие импульсы на выходе P первого полусумматора, который попадает на вход Σ_2 лишь в четвертом такте. В то же время за счет импульса на входе 1 Σ_2, поступает от второго такта, на выходе S этого полусумматора есть импульсы и др.

При помощи описанного сумматора можно осуществить последовательное сложение чисел с любым количеством разрядов, причем время, затраченное на сложение, равно количеству разрядов суммы.

Более сложным цифровым автоматом является арифметическое устройство (сумматор), предназначенное для параллельного сложения многоразрядных чисел. В этом сумматоре обеспечивается одновременное сложение всех разрядов слагаемых. Такой сумматор должен содержать $n + 1$ разрядных элементов, где n – разрядность слагаемых, а каждый i-й элемент представляет более сложную логическую схему, чем рассмотренная выше.

Типичным примером сложного цифрового автомата является компьютер, а также дискретные устройства типа машины Тьюринга.

Процесс синтеза сложных автоматов сводится к 3 этапам:

I. Блочный синтез – разбиение автомата на отдельные блоки, определение задач, которые будут решаться этими блоками и общего плана обмена информацией между блоками.

II. Абстрактный синтез – определение объема памяти блока на основании анализа задач, решаемых любым отдельным блоком и процесса изменения ее состояний в зависимости от исходной информации.

III. Структурный синтез – выбор элементов для построения схемы и способа соединения этих элементов между собой.

3.4. Абстрактный автомат

Абстрактным автоматом называют математическую модель дискретного устройства, имеющего один входной канал, куда поступают последовательности символов какого-либо языка, один выходной канал, с которого снимают последовательности символов какого-либо другого языка и находящегося в каждый из моментов дискретного времени в каком-либо состоянии. Графически абстрактный автомат представлен на рис. 3.7.

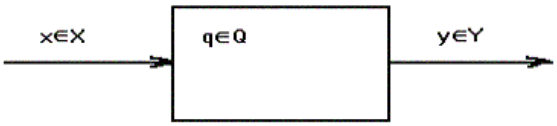


Рис. 3.7. Абстрактный автомат

Слова входного языка можно представить символами множества $X = \{x_1, x_2, ..., x_n\}$, который называют входным алфавитом, а слова выходного языка – символами множества $Y = \{y_1, y_2, ..., y_p\}$, который называют выходным алфавитом. Множество состояний автомата $Q = \{q_1, q_2, ..., q_m\}$ называют алфавитом состояний. С позиции формальных языков множество Q есть множество нетерминальных символов, а множества X и Y – множества терминальных символов [9; 21].

3.4.1. Модель абстрактного автомата

Понятие «состояние» используют для того, чтобы установить функциональную зависимость генерируемых автоматом символов и/или слов выходного языка от символов и/или слов входного языка при реализации автоматом заданного алгоритма. Для каждого состояния автомата $q \in Q$ и для каждого символа $x \in X$ в момент дискретного времени $[\tau]$ на выходе устройства генерируется символ $y \in Y$. Эту зависимость определяет функция выходов автомата ψ. Для каждого текущего состояния автомата $q \in Q$ и для каждого символа $x \in X$ в момент дискретного времени $[\tau]$ автомат переходит в очередное состояние $q \in Q$. Эту зависимость определяет функция переходов автомата φ. Функционирование автомата состоит в порождении двух последовательностей: последовательности очередных состояний автомата ($q_1[1]$, $q_2[2]$, $q_3[3]$, ...) и последовательности выходных символов ($y_1[1]$, $y_2[2]$, $y_3[3]$, ...), которые для последовательности символов ($x_1[1]$, $x_2[2]$, $x_3[3]$, ...) разворачиваются в моменты дискретного времени $\tau = 1, 2, 3, ...$ В прямоугольных скобках указывают моменты дискретного времени, которые

называют иначе тактами, в круглых скобках – последовательности символов алфавитов X, Y и Q.

Итак, математическая модель конечного автомата есть трехосновная алгебра, носителями которой являются три множества X, Y и Q, а операциями – две функции φ и ψ:

$$M = \langle X, Y, Q, \varphi, \psi \rangle \tag{3.7}$$

Таблица 15. Характеристики автомата M

$X = \{ x_1; x_2; ...; x_n \}$	– множество символов входного алфавита
$Y = \{ y_1; y_2; ...; y_p \}$	– множество символов выходного алфавита
$Q = \{q_1; q_2; ...; q_m\}$	– множество символов состояний автомата
φ: $(Q \otimes X) \rightarrow Q$	– функция переходов автомата для отображения пары $(q; x)$ текущего момента дискретного времени $[\tau]$ в состояние q очередного момента дискретного времени $[\tau+1]$
ψ: $(Q \otimes X) \rightarrow Y$	– функция выходов автомата для отображения пары $(q; x)$ текущего момента дискретного времени $[\tau]$ в символ y выходного канала этого же момента дискретного времени $[\tau]$

Так как области определения функций переходов и выходов совпадают, то обобщенный оператор поведения автомата можно представить так:

$$(\varphi, \psi): (Q \otimes X) \rightarrow (Q \otimes Y) \tag{3.8}$$

Функционирование автомата в дискретные моменты времени τ может быть описано системой рекуррентных соотношений:

$$\begin{cases} q_{\tau+1} = \varphi(q(\tau), x(\tau)) \\ y_\tau = \psi(q(\tau), x(\tau)) \end{cases}$$

Функциональная схема абстрактного автомата представлена на рис. 3.8.

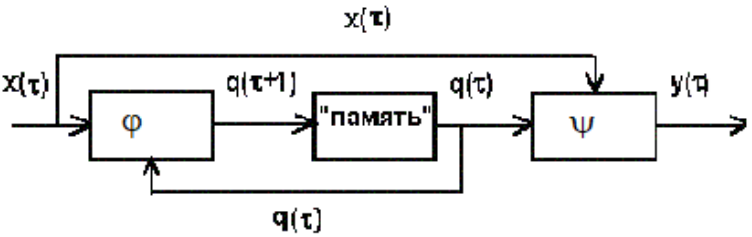


Рис. 3.8. Функциональная схема абстрактного автомата

Если функции переходов и выходов однозначно определены для каждой пары $(q;\ x)\in(Q\otimes X)$, то автомат называют детерминированным. В противном случае автомат называют недетерминированным или частично определенным.

Если функция переходов и/или функция выходов являются случайными, то автомат называют вероятностным.

Если у автомата задано начальное состояние $q = q_0\in Q$, в котором он находится всегда до приема первого символа входного слова, то автомат называют инициальным. В этом случае модель автомата записывают так:

$$M = \langle X;\ Y;\ Q;\ \varphi;\ \psi;\ q_0\rangle. \qquad (3.9)$$

Последовательность символов в слове b и последовательность состояний автомата q однозначно определяются начальным состоянием автомата $q = q_0$ и последовательностью символов во входном канале a. Поэтому отображение входного слова a на выходное слово b чаще называют автоматным отображением, то есть $b = M(q_0;\ a)$, а M – автоматным оператором.

Автоматное отображение обладает свойствами:

1) входное и выходное слова имеют одинаковую длину (свойство сохранения длины);

2) y_i-й символ выходного слова зависит от всей последовательности символов входного слова, до x_i-го включительно; кроме того, если $a = a_1a_2$, то $b = b_1b_2$.

3.4.2 Типы конечных автоматов

По способу формирования функций выхода выделяют автоматы Мили и Мура.

В автомате Мили функция выходов ψ определяет значение выходного символа по классической схеме абстрактного автомата. Математическая модель автомата Мили и схема рекуррентных соотношений не отличаются от математической модели и схемы рекуррентных соотношений абстрактного автомата, т.е.

$$\begin{cases} M = \langle X; Y; Q;\ \varphi;\ \psi \rangle \\ \varphi: (Q \otimes \mathrm{X}) \to \mathrm{Q} \\ \psi: (Q \otimes \mathrm{X}) \to \mathrm{Y} \end{cases}. \qquad (3.10)$$

$$\begin{cases} q_{\tau+1} = \varphi(q[\tau], x[\tau]) \\ y_\tau = \psi(q[\tau], x[\tau]) \end{cases}. \qquad (3.11)$$

Особенностью автомата Мили является то, что функция выходов является двухаргументной и символ в выходном канале $y[\tau]$ обнаруживается только при наличии символа во входном канале $x[\tau]$. Функциональная схема не отличается от схемы абстрактного автомата (см. рис 3.8 и 3.9).

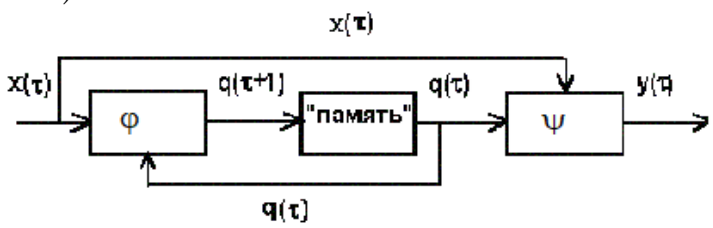


Рис. 3.9. Функциональная схема автомата Мили

В автомате Мура функция ψ определяет значение выходного символа только по одному аргументу – состоянию автомата. Эту функцию называют также функцией меток, так как она каждому состоянию автомата ставит метку на выходе. Математическая модель и схема рекуррентных соотношений автомата Мура имеют вид:

$$\begin{cases} M = \langle X; Y; Q;\ \varphi;\ \psi \rangle \\ \varphi: (Q \otimes \mathrm{X}) \to \mathrm{Q} \\ \psi: \mathrm{Q} \to \mathrm{Y} \end{cases}. \tag{3.12}$$

$$\begin{cases} q_{\tau+1} = \varphi(q[\tau], x[\tau]) \\ y_\tau = \psi(q[\tau]) \end{cases}. \tag{3.13}$$

Особенностью автомата Мура является то, что символ $y[\tau]$ в выходном канале существует все время пока автомат находится в состоянии $q[\tau]$. Функциональная схема автомата Мура представлена на рис. 3.10.

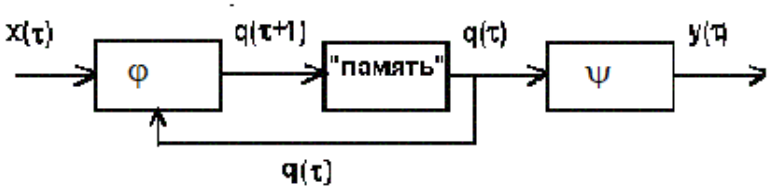


Рис. 3.10. Функциональная схема автомата Мура

Объединение автоматов Мили и Мура представляет *C*-автомат, для которого схема рекуррентных соотношений имеет вид:

$$\begin{cases} q_{\tau+1} = \varphi(q[\tau], x[\tau]) \\ y_\tau = \psi(q[\tau], x[\tau]) \\ y_\tau = \psi(q[\tau]) \end{cases}. \tag{3.14}$$

Потребность такого автомата возникает при формировании автоматных сетей. Функциональная схема *C*-автомата представлена на рис. 3.11.

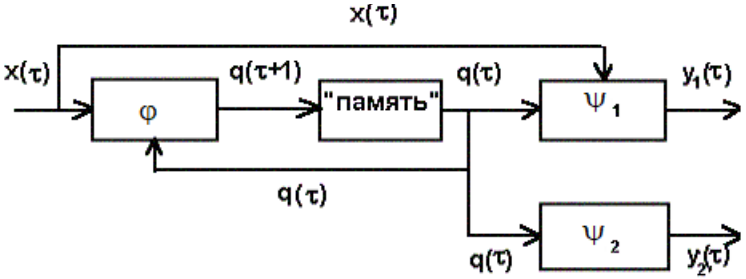


Рис. 3.11. Функциональная схема *C*-автомата

Интересно выделить особые классы автоматов, математические модели которых опираются только на два носителя алгебры.

Пусть $X = \varnothing$. Тогда математическая модель и система рекуррентных соотношений имеют вид:

$$\begin{cases} M = \langle Y; \mathrm{Q};\ \varphi;\ \psi \rangle \\ \varphi: \mathrm{Q} \to \mathrm{Q} \\ \psi: \mathrm{Q} \to \mathrm{Y} \end{cases}. \tag{3.15}$$

$$\begin{cases} q_{\tau+1} = \varphi(q[\tau]) \\ y_\tau = \psi(q[\tau]) \end{cases}. \tag{3.16}$$

Функциональная схема автомата приведена на рис. 3.12.

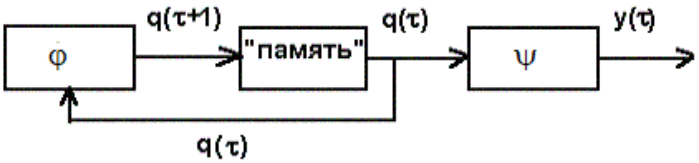


Рис. 3.12. Функциональная схема порождающего автомата.

Особенностью функционирования такого автомата является генерация последовательности символов вы-

ходного слова только в зависимости от последовательности состояний автомата. Такие автоматы называют порождающими или автономными. С помощью такого автомата генерируется последовательность управляющих команд на какие-либо объекты внешней среды.

Пусть $Y = \varnothing$. Тогда математическая модель и система рекуррентных соотношений имеют вид:

$$\begin{cases} M = \langle X; Q; \varphi \rangle \\ \varphi: (Q \otimes X) \to Q \end{cases} . \quad (3.17)$$

$$q_{\tau+1} = \varphi(q[\tau], x[\tau]) . \quad (3.18)$$

Функциональная схема автомата приведена на рис. 3.13.

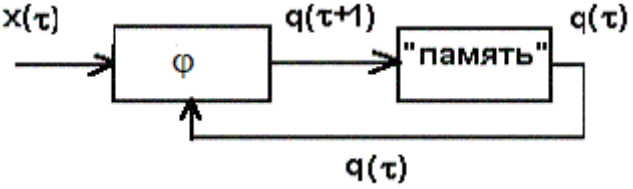


Рис. 3.13. Функциональная схема распознающего автомата

Особенностью функционирования такого автомата является распознавание в последовательности изменений аргумента функции переходов значения ($q_i[\tau]$; $x_l[\tau]$) и перевод автомата в заключительное состояние q_k. С помощью такого автомата обнаруживают заданные возмущения со стороны объектов внешней среды или распознают заданную последовательность входных символов. Поэтому такие автоматы называют распознающими. Часто автомат Мура представляют автоматом без выхода, так как его выходной сигнал эквивалентен состоянию автомата.

Пусть $Q = \varnothing$. Тогда математическая модель и система рекуррентных соотношений имеют вид:

$$\begin{cases} M = \langle X; Y; \psi \rangle \\ \psi : X \to Y \end{cases} . \tag{3.19}$$

$$y_\tau = \psi(x[\tau]) \quad . \tag{3.20}$$

Функциональная схема автомата приведена на рис. 3.14.

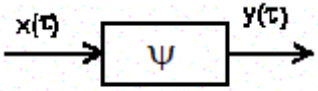


Рис. 3.14. Функциональная схема комбинационного автомата.

3.5. Структурный автомат

Если автомат представляет собой устройство, имеющее вход и выход, то последовательное и/или параллельное соединение нескольких автоматов формирует сеть. Под действием входных сигналов происходит изменение внутренних состояний автоматов, что порождает изменение состояния всей сети. При описании сети необходимо также вводить понятие дискретного времени τ. Функция φ каждого автомата реализует задержку на один такт изменения внутреннего состояния, что формирует задержку изменения состояния всей сети. Совокупность функций ψ автоматов, принадлежащих сети, формирует выходной сигнал всей сети.

Сеть автоматов, вход которой имеет n каналов, т.е. $(x_1, x_2, \ldots, x_n) \in X^n$, а выход – p каналов, т.е. $(y_1, y_2, \ldots, y_p) \in Y^p$, называют структурным автоматом. На рис. 3.15 и 3.16 даны схемы последовательного и параллельного соединений двух автоматов, формирующих структурный автомат.

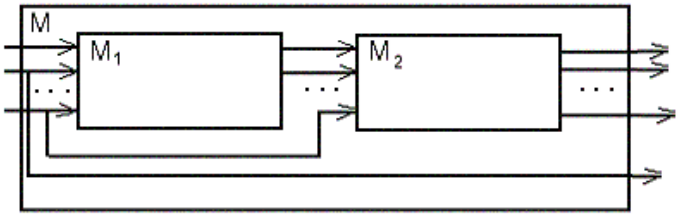


Рис. 3.15. Последовательное соединение автоматов

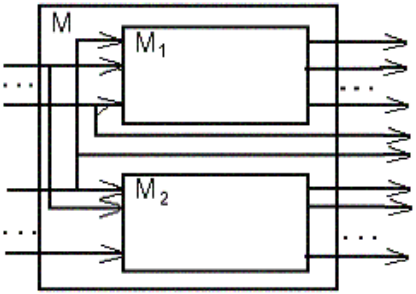


Рис. 3.16. Параллельное соединение автоматов

Структурный автомат, как правило, содержит m автоматов, состояния которых формируют состояние сети в виде кортежа $(q_1, q_2, ..., q_m) \in Q_m$.

Одновременное изменение внутренних состояний всех автоматов определяет синхронный режим работы сети. В этом случае состояние сети из m автоматов M_1, M_2, ..., M_m может быть представлено для каждого момента времени t вектором $q[\tau] = (q_1[\tau]; q_2[\tau]; ...; q_m[\tau])$. Каждая компонента этого вектора описывает внутреннее состояние соответствующего автомата, т.е. $q_1 \in Q_1$, $q_2 \in Q_2$, ..., $q_m \in Q_m$. Число состояний сети равно произведению числа состояний составляющих его автоматов, так как $Q = Q_1 \otimes Q_2 \otimes \ldots \otimes Q_m$. Поэтому синхронный режим работы сети часто называют произведением автоматов.

Разновременное и последовательное изменение внутренних состояний автоматов формирует асинхронный режим работы сети. Изменение состояния такой сети из m автоматов M_1, M_2, ..., M_m для каждого момента времени τ может быть описано изменением внутреннего состояния только одного автомата, т.е. $q[\tau] = q_i[\tau]$ где $q_i \in Q_i$. Число состояний сети равно сумме числа внутренних состояний составляющих его автоматов, так как $Q = Q_1 \cup Q_2 \cup \ldots \cup Q_m$. Поэтому асинхронный режим работы сети часто называют суммой автоматов.

3.5.1. Произведение автоматов

Композиция автоматов M_1 и M_2 при синхронном режиме их работы есть автомат $M = \langle X, Y, Q, \varphi, \psi \rangle$, внутренние состояния которого $q = (q_{1i} ; q_{2j}) \in (Q_1 \otimes Q_2)$ приведены в таблице 16.

Таблица 16. Композиция автоматов M_1 и M_2

q_1	q_2	q_3	q_4	q_5	q_6
$(q_{11} ; q_{21})$	$(q_{11} ; q_{22})$	$(q_{12} ; q_{21})$	$(q_{12} ; q_{22})$	$(q_{13} ; q_{21})$	$(q_{13} ; q_{22})$

Последовательное соединение двух автоматов

Пусть автоматы M_1 и M_2 работают в синхронном режиме и соединены так, как показано на рис. 3.17. При этом имеем $X = X_1$, $Y_1 = X_2$, $Q = (Q_1 \otimes Q_2)$ и $Y = Y_2$.

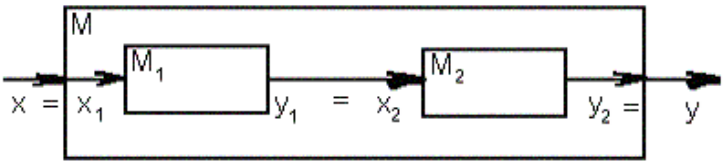


Рис. 3.17. Последовательное соединение автоматов

Функционирование автомата M может быть описано системой рекуррентных соотношений:

$$\begin{cases} q[\tau+1] = (q_1[\tau+1];\ q_2[\tau+1]) = (\varphi_1(q_1[\tau]; x_1[\tau]); \varphi_2(q_2[\tau]; x_2[\tau])); \\ y[\tau] = \psi(q[\tau]; x[\tau]) = \psi_2\left(q_2[\tau]; \psi_1(q_1[\tau];\ x_1[\tau]\right) \end{cases}$$

Последовательное соединение автоматов – некоммутативная операция. Поэтому при смене мест автоматов M_1 и M_2 меняется поведение автомата M.

Пусть автоматы M_1 и M_2 соединены так, как показано на рис. 3.18

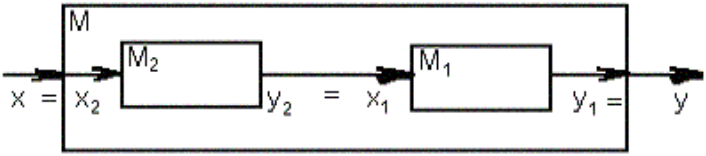


Рис. 3.18. Последовательное соединение автоматов

Часто операцию последовательного соединения автоматов называют их суперпозицией.

Параллельное соединение двух автоматов

Пусть автоматы M_1 и M_2 работают в синхронном режиме и соединены так, как показано на рис. 3.19.

Отличие схем заключено в формировании входных и выходных сигналов сети при различном соединении автоматов.

Функционирование автомата M (рисунок 3.19а) может быть описано системой рекуррентных соотношений:

$$\begin{cases} q[\tau+1] = (q_1[\tau+1];\ q_2[\tau+1]) = (\varphi_1(q_1[\tau]; x_1[\tau]); \varphi_2(q_2[\tau]; x_2[\tau]))); \\ y[\tau] = (y_1[\tau]; y_2[\tau]) = \psi_1((q_1[\tau]; x_1[\tau]); \psi_2(q_2[\tau]; x_2[\tau])) \end{cases}$$

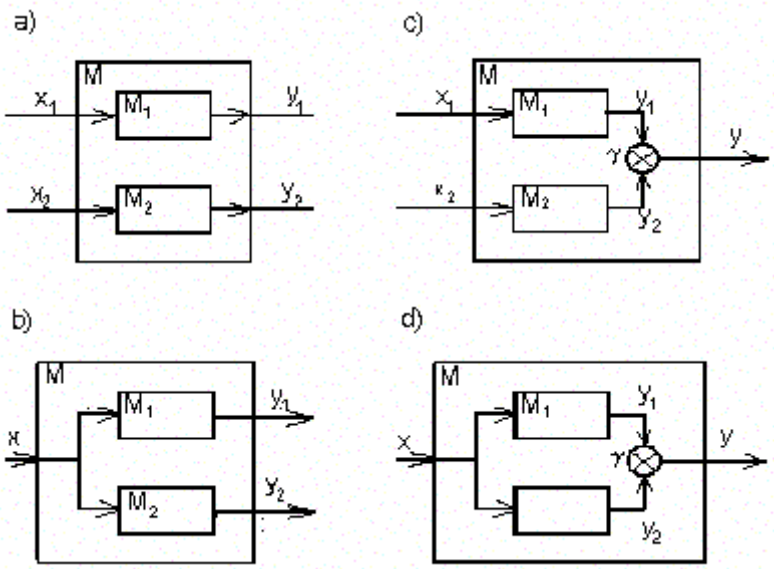


Рис. 3.19. Схемы параллельного соединения автоматов M_1 и M_2

Смена состояний и формирование выходных символов в каждом автомате сети происходит независимо и одновременно. Такая схема позволяет исследовать сложные автоматы, имеющие несколько входов и выходов.

Функционирование автомата M (рисунок 3.19b) может быть описано системой рекуррентных соотношений:

$$\begin{cases} q[\tau+1] = (q_1[\tau+1];\ q_2[\tau+1]) = (\varphi_1(q_1[\tau]; x_1[\tau]); \varphi_2(q_2[\tau]; x_2[\tau]))); \\ y[\tau] = (y_1[\tau]; y_2[\tau]) = \psi_1((q_1[\tau]; x[\tau]); \psi_2(q_2[\tau]; x[\tau])) \end{cases}$$

3.5.2. Обратная связь двух автоматов

Пусть автоматы M_1 и M_2 соединены так, как показано на рис. 3.20. При таком соединении $Q = (Q_1 \otimes Q_2)$, $Y_1 = X_2 = Y$ и δ: $(X \otimes Y_2) \to X_1$.

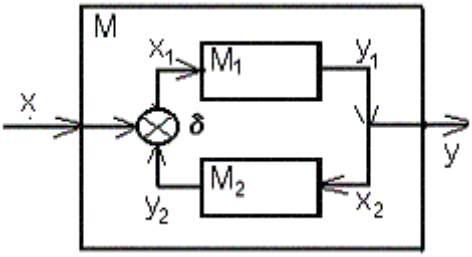


Рис. 3.20. Композиция двух автоматов по схеме обратной связи

Функционирование такого автомата M может быть описано системой рекуррентных соотношений:

$$\begin{cases} q[\tau+1] = (q_1[\tau+1];\ q_2[\tau+1]) = (\varphi_1(q_1[\tau]; \delta(x_1[\tau]; y_2[\tau])); \varphi_2(q_2[\tau]; x_2[\tau])); \\ y[\tau] = \psi_1((q_1[\tau]; \delta(x_1[\tau]; y_2[\tau])) \end{cases}$$

Оператор δ определяет значение входного символа для автомата M_1 согласно задаче, поставленной автомату M, так как δ: $(X \otimes Y_2) \to X_1$.

Контрольные вопросы

1. В чем заключается синтез автомата?
2. Для чего необходимо оптимизировать структуру автомата?
3. В каких единицах измеряется тактовая частота?
4. Какая логическая функция дает на выходе 0, только когда оба входа соответствуют 0?

5. В каком автомате функция ψ определяет значение выходного сигнала только по одному аргументу?
6. Как называется функция, определяющая изменения состояний под воздействием входных сигналов?
7. Как задается автомат?
8. Что собой представляет функция переходов?
9. Что собой представляет функция выходов?
10. Как называется автомат, если функции переходов и выходов однозначно определены?
11. Чем отличается математическая модель автомата Мили и схемы рекуррентных соотношений от математической модели и схемы рекуррентных соотношений абстрактного автомата?
12. Какой режим работы сети автоматов называют асинхронным?
13. Какой режим работы сети автоматов называют синхронным?
14. Что собой представляет автомат без памяти?
15. В чем заключается блочный синтез?
16. В чем заключается абстрактный синтез?
17. В чем заключается структурный синтез?

ГЛАВА IV. ГРАММАТИКИ

4.1. Формальная порождающая грамматика

Пусть задан алфавит V и тем самым множество V^* всех конечных слов, или цепочек в алфавите V. Формальный язык L в алфавите V – это произвольное подмножество $L \subset V^*$. Конструктивное описание формальных языков осуществляется с помощью формальных систем, называемых формальными порождающими грамматиками (ФПГ).

ФПГ G – это формальная система, определяемая четверкой объектов $G = <V, W, I, P>$, где:

V – алфавит, или словарь, терминальных (вспомогательных) символов;

W – алфавит, или словарь терминальных (основных) символов;

$V \cap W = \emptyset$;

I – начальный символ грамматики, или аксиома грамматики;

P – система правил.

Язык $L(G)$, порождаемый грамматикой G, – это множество всех цепочек в терминальном алфавите V выводимых из I.

Пример: множество нечетных чисел в унарном представлении – это множество цепочек вида a, aaa, $aaaaa$, т.е. язык $\{a^{2n-1}\}$

Он порождается грамматикой $G = <V, W, I, R>$, где $V = \{a\}$; $W = \{I\}$; R – множество правил вида:

$$\begin{cases} I \to a \\ I \to aaI \end{cases}.$$

Рассмотрим слово *аааа*.

аааа → *ааа*I → *а*I .

Это значит, что слово *аааа* не является словом на данном языке.

Иначе обстоят дела со словом *ааа*. *ааа* → *аа*I→ I, т.е. *ааа* является словом на данном языке.

4.2. Синтаксис

Чтобы понять текст на каком-либо языке, необходимо два условия: наличие в нем структуры и смысла.

Правила, определяющие структуру текста, называются грамматикой или синтаксисом языка. Однако не удастся создать понятный фрагмент текста из слов, которых нет в словаре, не получится построить предложения, имеющие смысл и др., то есть мы должны построить текст, имеющий смысл.

Правила, управляющие смыслом языка, называются семантикой.

Существует два метода описания синтаксиса:

– синтаксические диаграммы;

– расширенная формула Бекуса–Наура.

Для иллюстрации будем использовать крайне ограниченный язык Показушка [6]. Он содержит 10 слов.

«Черное», «кошка», «ест», «упитанная», «тощая», «белая», «мышка», «бежит», «быстро», «медленно», «.», «,».

Эти слова и знаки препинания называются ***лексемами***.

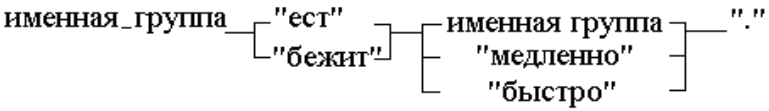
На этом языке можно составлять предложения о кошках и мышках.

Например, «Белая мышка бежит быстро.».

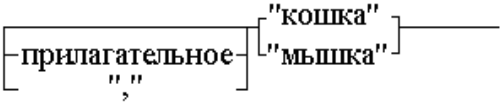
4.2.1. Синтаксические диаграммы

Рассмотрим следующие синтаксические диаграммы.

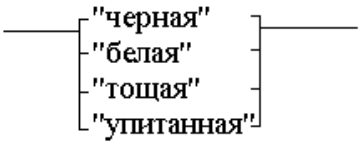
предложение



именная_группа



прилагательное



Каждая диаграмма определяет синтаксический класс рассматриваемого языка, в данном случае – Показушки.

Верхняя диаграмма определяет синтаксический класс «предложение», для чего привлекается другой класс, называющийся «именная группа». Именная группа определяется в средней диаграмме, и для этого, в свою очередь, используется «прилагательное», определенное на нижней диаграмме.

Чтобы построить отрывок текста по синтаксической диаграмме, нужно проделать следующее:

a) двигаясь слева направо, пройти по линии диаграммы от начала до конца, помещая в текст все встретившиеся элементы;

b) когда в определении упомянуто называние другого синтаксического класса, вместо него можно подставить

любой фрагмент, построенный на соответствующей синтаксической диаграмме.

В этом процессе имя синтаксического класса в определении другого синтаксического класса действует аналогично оператору процедуры в программе. Пользуясь синтаксической диаграммой можно, получить следующие примеры предложений:

Именная_группа «ест» (или «бежит») именная_ группа «.».

Именная_группа «ест» (или «бежит») «быстро» (или «медленно») «.»,

где каждая Именная группа должна быть заменена любым фрагментом текста, порожденным при прохождении синтаксического класса «именная_группа».

Обратившись к ней, мы обнаруживаем бесконечное многообразие Именных групп, которые можно построить благодаря тому, что диаграмма содержит цикл – путь, замкнутый на себя через лексему «,». Этот путь может проходить сколько угодно раз, всякий раз порождая одну из четырех лексем диаграммы прилагательных:

«кошка»

«мышка»

«упитанная» «,» «черная» «кошка»

«черная» «,» «черная» «кошка»

«белая» «,» «черная» «,» «упитанная» «,» «тощая» «,» «тощая» «,» «белая» «кошка»

Они позволяют образовать предложения типа:

«кошка» «ест» «мышку» «.»

«черная» «кошка» «ест» «медленно» «.»

«упитанная» «,» «черная» «кошка» «ест» «тощую» «,» «белую» «мышку» «.»

Или другие предложения.

Все эти предложения синтаксически правильны в Показушке. Некоторые предложения имеют смысл, некоторые не имеют. Таким образом, следование синтаксическим правилам не гарантируют получение осмысленных предложений.

4.2.2. Расширенная форма Бэкуса–Наура

Второй метод определения синтаксиса состоит в применении особых обозначений, называемых расширенной формой Бэкуса-Наура (РФБН).

РБНФ – язык для определения других языков. Он широко используется в документации языков программирования. Определение языка «Показушка» в РБНФ:

1. предложение	= подлежащее предикат «.»
2. подлежащее	= именная_группа
3. предикат	= глагол (именная_группа | наречие)
4. именная группа	= [список_прил] существительное
5. список_ прилагательных	= прилагательное {«,» прилагательное}
6. существительное	= «кошка» | «мышка»
7. прилагательное	= «белая» | «черная» | «упитанная» | «тощая»
8. глагол	= «ест» | «бежит»
9. наречие	= «медленно» | «быстро»

Каждое из них определяет один синтаксический класс в Показушке, а все вместе они полностью определяют синтаксический класс «предложение».

В определения входят:

a) записанные в кавычках лексемы;

b) называния синтаксических классов;

с) записанные без кавычек дополнительные знаки: =, ., |, {}, [], ().

Рассмотрим определения:

«=» – может читать «определяется как…»

А точка обозначает конец определения.

Второе определение – подлежащее, определяется как именная_группа.

То, что названия синтаксических классов разделены пробелами, означает, что они следуют друг за другом.

Предложение определяется как подлежащее, за которым следует предикат, за которым, в свою очередь, следует лексема «.».

Следует различать точку в конце определения и «.» – как завершение.

Вертикальная черта означает выбор, альтернативу (в определении глагол определяется как лексема «бежит» или как лексема «ест»).

Фигурные скобки означают возможность повторения, что соответствует замкнутому циклу в синтаксической диаграмме (список_прилагательных определяется как прилагательное и повторение). Это означает, что список_прилагательных должен содержать, по крайней мере, одно прилагательное, но что за ним может идти и другое прилагательное через запятую («,»).

«черная»

«упитанная» «,» «черная»

«тощая» «,» «тощая» «,» «белая»

Квадратные скобки обозначают необязательную часть синтаксического класса. В определении 4 именная_группа определяется как существительное, перед

которым следует, но необязательно, список_прилагательных.

Круглые скобки, используемые вместе с вертикальной чертой, обозначают альтернативы внутри определения (определение 3 предиката определяется как глагол, за которым следует либо именная_группа, либо наречие). Например, предикат = глагол (именная_группа | наречие).

Следующие предикаты являются правильными:

«ест» «мышку» «.»

«бежит» «медленно» «.»

«бежит» за «черной» «,» «белой» «кошкой» «.»

4.2.3. Синтаксический анализ

Процесс, обратный построению синтаксических диаграмм называется синтаксическим анализом. Одним из методов его проведения является построение синтаксического дерева.

Этапы синтаксического анализа:

1. Расчленить текст, превратив его в последовательность лексем;

2. Пройти по тексту слева направо и найти последовательно составные части, образующие объект данного синтаксического класса.

Будем выяснять, являются ли следующие предложение синтаксически правильными в Показушке, так как мы пытаемся доказать, что текст – это предложение. Обратимся к синтаксической диаграмме «предложение». Диаграмма показывает, что всякое предложение начинается с именной группы,поэтому в начале текста мы должны найти именную группу.

Именная_группа – другой синтаксический класс, поэтому надо провести еще один разбор, чтобы найти именную группу. Согласно соответствующим синтаксическим диаграммам лексема «кошка» допустима в качестве именной группы (см. рис.4.1).

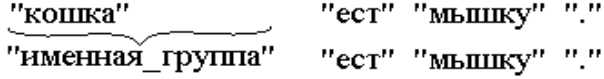


Рис. 4.1. Первый этап разбора

За именной группой, согласно синтаксической диаграмме, следует искать лексему «ест» или «бежит», т.е. в нашем примере лексема «ест». Далее мы должны найти либо еще одну «именную_группу», либо лексему «быстро», либо лексему «медленно» (см. рис.4.2).

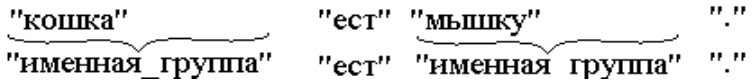


Рис. 4.2. Второй этап разбора

По наличию в конце лексемы «.» можно заключить, что первый фрагмент в нашем примере является правильным «предложением».

Существенно, что синтаксический анализ не является простым разглядыванием текста с целью отнесения его к какому-нибудь из классов. Нужно знать, какой класс мы ищем, и определять, принадлежит ли текст этому классу.

В нашем примере, начиная анализ, мы искали предложение, и это побудило искать именную_группу. На следующих этапах мы тоже точно знали, что ищем.

Пример: проверить, являются ли предложенные фразы предложениями на языке Показушка. Будем считать, что в конце каждой фразы есть точка (см. рис. 4.3).

1) кошка ест мышку;

2) упитанная, черная, черная кошка бежит быстро;

3) быстро бежит мышка.

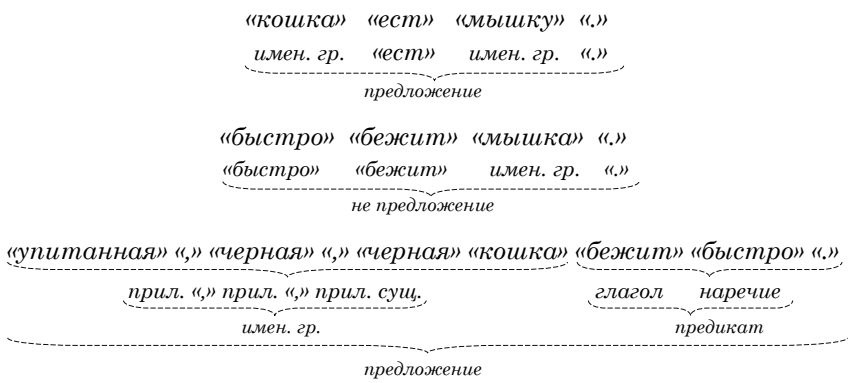


Рис. 4.3. Пример грамматического разбора

4.3. Грамматики

Терминальные символы языка – это базовые элементы, из которых конструируются строки и предложения языка. В естественном языке это слова и части слов.

Пример: стол, окно, a, b.

1. Терминальные символы записываются прописными латинскими буквами (A, B, C, …); V – множество терминальных символов.

2. Синтаксические переменные – строчные греческие буквы (α, β, γ, …); W – множество синтаксических переменных.

Пример: <имя синтаксической переменной>, <существительное>, …

Рассмотрим, как конструируются строки терминальных символов, называемых предложениями языка.

Имеется строка, содержащая синтаксическую переменную, оператор подстановки заменяет эту переменную на подстроку, соответствующую этой переменной.

Пример: $\xi \to \bar{x}$, ξ – синтаксическая переменная, $\bar{x}$ – строка терминальных или переменных символов.

Если две грамматики G_1 и G_2 порождают одно и то же множество предложений, то они называются эквивалентными. Левая строка порождает правую строку.

4.3.1. Определение грамматик

V^* – множество строк конечной длины из элементов множества V, W^* – множество строк конечной длины из элементов множества W, ε – пустая строка.

Правило подстановки – это упорядоченная пара $(\bar{x}, \bar{y})$, где $\bar{x} \in (W^* - \varepsilon); \bar{y} \in (V \cup W)^*; \bar{x} \to \bar{y}$, где $\bar{x}$ – корень правила; $\bar{y}$ – аргумент правила.

Грамматикой с фразовой структурой называется упорядоченная четверка $G = <V, W, P, \sigma>$; где V – множество терминальных символов, W – множество переменных символов, P – пронумерованное множество правил подстановок, σ – начальный символ грамматики.

В русском языке σ – пробел.

Пусть $\bar{x}, \bar{y} \in (V \cup W)$; $\bar{x} \neq \varepsilon$, тогда говорят, что $\bar{x}$ порождает $\bar{y}$ $(\bar{x} => \bar{y})$.

Пример: $G = <\{A, B, C\}, \{\alpha, \beta, \gamma, \delta\}, P, \sigma>$

P: $\{\sigma \to \alpha\beta{:}1$

$\alpha \to AB{:}2$

$\beta\gamma \to \alpha{:}3$

$\beta \to C{:}4\}$

$$\sigma \xrightarrow{1} \alpha\beta \xrightarrow{2} AB\beta \xrightarrow{4} ABC$$

$$\sigma \xrightarrow{1} \alpha\beta \xrightarrow{4} \alpha C \xrightarrow{2} ABC$$

Множество Λ строк терминальных символов, ($\Lambda \subset V$) такое, что $\bar{x} \in \Lambda$, $\delta \Rightarrow \bar{x}$. Элемент Λ называется предложением языка. Фактически язык – это множество (необязательно конечное) строк терминальных символов в соответствии с правилами заданной грамматики.

Дадим обобщенное понятие предложения.

Сентенциальной формой $\bar{s}$ грамматики $G = <V, W, P, \sigma>$ называется строка терминальных и/или переменных символов, получаемая из σ.

Грамматика $G = <V, W, P, \sigma>$ называется контекстно-свободной грамматикой, если она есть грамматика с фразовой структурой, в которой выполняется правило подстановки $\bar{x} \to \bar{y}, x \in W$, где x – одиночная переменная.

Отдельные правила, обладающие этими свойствами, называются контекстно-свободными правилами грамматики.

Грамматика $G = <V, W, P, \sigma>$ называется контекстно-зависимой, если она есть грамматика с фразовой структурой и если каждое правило $\bar{x} \to \bar{y}$ из P имеет вид $\bar{x} = \bar{u}_1 \xi \bar{u}_2; \bar{y} = \bar{u}_1 z \bar{u}_2; z \neq \varepsilon$

Отдельные правила, обладающие этими свойствами, называются контекстно-зависимыми правилами подстановки (КС).

Свойства формальных грамматик

1. Свойство локальности – это когда результат подстановки, выполненный в некоторой области строки, остается в этой области.

Лемма: Пусть G – грамматика, $\bar{x}, \bar{y}$ – сентенциальные формы такие, что $\bar{x} \Rightarrow \bar{y}$. Если $\bar{x} = \overline{x_1 x_2}$, а $\bar{y} = \overline{y_1 y_2}$, то существует разбиение на две подстроки такие, что $\overline{x_1} \Rightarrow \overline{y_1}, \overline{x_2} = \overline{y_2}$ или $\overline{x_1} = \overline{y_1}, \overline{x_2} \Rightarrow \overline{y_2}$.

Теорема 1. G – грамматика, $\bar{x} \overset{*}{\Rightarrow} \bar{y}$. Тогда, если $\bar{x} = \overline{x_1 x_2}$, то существует такое разбиение $\bar{y} = \overline{y_1 y_2}$, что $\overline{x_1} \overset{*}{\Rightarrow} \overline{y_1}$, а $\overline{x_2} \overset{*}{\Rightarrow} \overline{y_2}$.

Теорема 2. Пусть G – грамматика, $\bar{x}, \bar{y}$ – сентенциальные формы в этой грамматике, что $\bar{x} \overset{*}{\Rightarrow} \bar{y}$. Тогда, если $\bar{x} = \bar{x}_1 \ldots \bar{x}_n$, то существуют такие $\bar{y}_1 \ldots \bar{y}_n$, что $\bar{y} = \bar{y}_1 \ldots \bar{y}_n$, $\bar{x} \Rightarrow \bar{y}_1, \ldots, \bar{x}_n = \bar{y}_n$.

На основании данной теоремы общая подстановка разбивается на некоторое множество последовательных подстановок. Это приводит к простоте в определении возможности данного вывода. Введем обозначение: $|\bar{x}|_a$ – количество символов a в строке $\bar{x}$.

4.3.2. Канонические формы

Теорема: Пусть вывод $\bar{x} \overset{*}{\Rightarrow} \bar{y}$ получен последовательностью подстановок $B_1, B_2, \ldots, B_i, B_{i+1} \ldots, B_n$.

Пусть существует разбиение $\bar{x}$: $\bar{x} = \overline{x_1 x_2}$, $\bar{y} = \overline{y_1 y_2}$ такие, что B_i входит в цепочку правил, для вывода $\bar{x} \overset{*}{\Rightarrow} \bar{y}$,

а B_{i+1} входит в цепочку правил для вывода $\overline{x_2} \overset{*}{\Rightarrow} \overline{y_2}$. Тогда вывод $\bar{x} \overset{*}{\Rightarrow} \bar{y}$ получается последовательности правил $B_1, B_2, ..., B_i, B_{i+1} ..., B_n$.

Суть проблемы состоит в том, что мы хотим представить «двумерную» структуру $\left(\overline{x_1}, \overline{x_2} \Rightarrow \overline{y_1}, \overline{y_2}\right)$, как набор одномерных структур. Но таких одномерных представлений много, следовательно, нам необходимо выделить какие-то разбиения $\bar{x}$ на $\overline{x_1}\,\overline{x_2}$, которые бы позволили алгоритмизировать этот процесс.

Пусть $\overline{z_1} \Rightarrow \overline{z_n}$. Последовательность

$$z_1 \overset{B_1}{\Rightarrow} z_2 \overset{B_2}{\Rightarrow} ... \Rightarrow z_i \overset{B_i}{\Rightarrow} ... \overset{B_{n-1}}{\Rightarrow} z_{n-1} \overset{B_n}{\Rightarrow} z_n$$

называется канонической последовательностью подстановок канонической последовательности правил, если каждое правило подстановки применяется к самой левой переменной, имеющейся в строке.

Теорема. Любая последовательность правил может быть приведена к канонической форме.

Каноническая форма не однозначна (не единственна). Если имеется предложение, у которого более одного канонически последовательного правила, начиная с начальной переменной, то порождаемая грамматика называется неоднозначной грамматикой.

Один и тот же язык может порождаться как однозначной, так и неоднозначной грамматикой.

Пример 1. Грамматика задана правилами:

P_1: $\{\sigma \to \alpha B : 1$
$\sigma \to A\beta : 2$
$\alpha \to A : 3$

$$\beta \to B : 4\}$$

Можем получить из σ либо $\sigma \overset{1}{\Rightarrow} \alpha B \overset{3}{\Rightarrow} AB$; либо $\sigma \overset{2}{\Rightarrow} A\beta \overset{4}{\Rightarrow} AB$. Следовательно, P_1 неоднозначная грамматика.

Пример 2.

$$P_2: \{\sigma \to \alpha B : 1$$
$$\alpha \to A : 2$$
$$\beta \to B : 4\}$$

Можем получить из σ только $\sigma \overset{1}{\Rightarrow} \alpha B \overset{2}{\Rightarrow} AB$. Следовательно, P_2 – однозначная грамматика.

4.3.3. Двоичные деревья трансляции

Если T – произвольное дерево, то $B(T)$ строится по следующим правилам: для любых вершин a, b дерева T:

1) вершина «b» – левый потомок «a» в $B(T)$, тогда и только тогда, когда «b» – самый левый потомок «a» в T.

2) вершина «a» – правый потомок «b» в $B(T)$, тогда и только тогда, когда «b» – ближайший из правых потомков «a» в T.

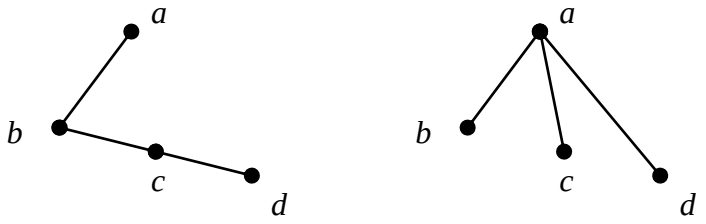


Рис. 4.4. Примеры двоичных деревьев

Рассмотрим особенности двоичных деревьев. Левосторонней (правосторонней) цепочкой двоичного дерева

называется последовательность вершин $a_1, a_2, ..., a_n, n \geq 1$, где a_i – левый (правый) потомок a_{i-1}.

Говорят, что цепочка $a_1,\ldots,a_n$ содержится в цепочке $b_1,b_2,\ldots,b_m$, если существует $j \geq 0$ такое, что $a_i = b_j + 1$, $i = 1,..., n; n \leq m$

Цепочка называется максимальной, если она содержится только сама в себе.

Лемма. Левосторонняя цепочка в $B(T)$ является левой цепочкой в T и наоборот.

Лемма. Максимальная левосторонняя цепочка в $B(T)$ является аргументом некоторого правила подстановки P. Предок первой вершины в цепочке есть корень P.

Максимальная левосторонняя цепочка содержит только один терминальный символ, который является последней вершиной в цепочке.

4.4. Структуры данных

Во время трансляции данные хранятся в трех стеках: ввод, выдача, хранение.

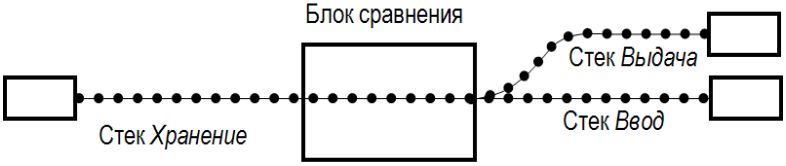


Рис. 4.5. Структура обработки данных

Символы выбираются по одному из ввода и подаются в стек хранения. Блок сравнения проверяет, не содержит ли правая часть стека хранения возможной простой

фразы. При обнаружении возможной основы, эта основа передается в стек выдачи. Трансляция осуществляется по определенному алгоритму (см. рис. 4.6).

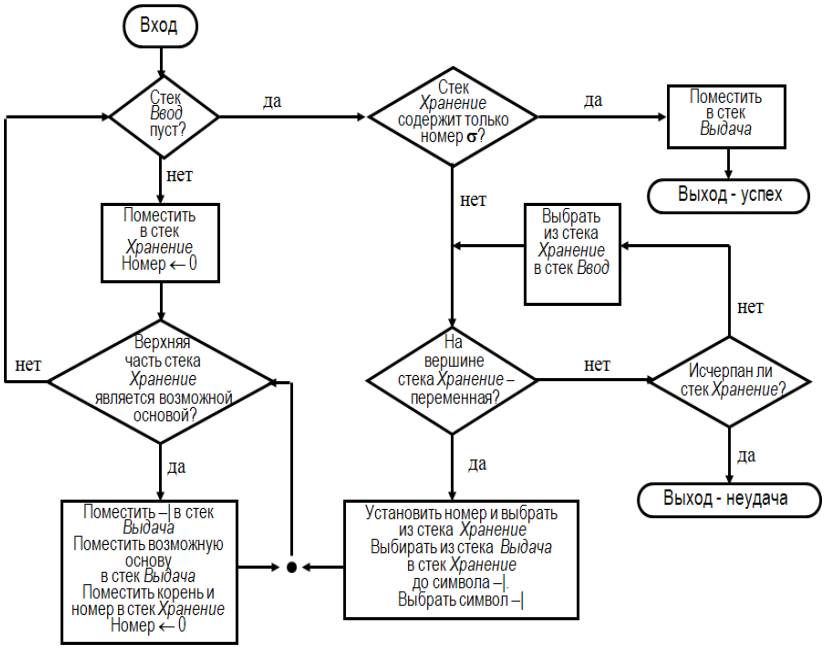


Рис. 4.6. Алгоритм грамматического разбора

Этот же алгоритм грамматического разбора ниже приведен в виде блоков.

B1. (выбрать следующий входной символ)
Если *Ввод* пуст, то перейти к **B4**;
Иначе выбрать из *Ввод* в *Хранение*;
Номер := 0.

B2. (поиск основы)
<искать правило подстановки с номером большим, чем Номер, аргумент которого совпадает с верхней подстрокой в стеке *Хранение*>

Если такого правила нет, то перейти к **B2**.

B3. (выдать возможную основу)

Поместить —| в стек *Выдач*а;

Выбрать совпавшую подстроку из стека *Хранение* в стек *Выдача*;

Поместить корень и номер совпавшего правила в стек *Хранение*;

Номер := 0;

Перейти к **B2**.

B4. (проверка завершенности)

Если стек *Хранение* содержит только σ/<номер правила> на вершине, то поместить в стек *Выдача*. Выход – **успех.**

B5. (откат)

Выбирать из стека *Хранение* в стек *Ввод* до тех пор, пока пара <переменная><номер правила> не окажется на вершине стека *Хранение*;

Если нет, то Выход – **неудача.**

иначе выбрать из стека *Хранение;*

Номер := <номер правила>;

Выбирать из стека *Выдача* в стек *Хранение* до тех пор, пока не обнаружится символ —|.

Выбрать из стека *Выдача* символ —|;

Перейти к **B2**.

Этот алгоритм является примером алгоритма грамматического разбора. Можно проверить является ли выражение $(a+b)/c$ арифметическим. Также с помощью данного алгоритма можно продемонстрировать формальное исполнение алгоритма.

Контрольные вопросы

1. Что собой представляет правило подстановки, строка терминальных символов, сентенциальная форма?
2. Как задается грамматика?
3. Привести пример грамматики.
4. Дать определение предложения.
5. Какие символы относятся к терминальным символам?
6. Что собой представляет синтаксическая переменная?
7. Может ли быть несколько правил подстановки с одинаковыми левыми частями?
8. Какая грамматика называется контекстно-свободной грамматикой?
9. Какая грамматика называется контекстно-зависимой грамматикой?
10. Что такое язык? Сентенциальная форма?
11. В чем заключается свойство локальности?
12. Любая последовательность правил может быть приведена к канонической форме?
13. Каноническая форма однозначна?
14. Какая цепочка называется левосторонней цепочкой двоичного дерева?
15. Какая цепочка называется максимальной цепочкой двоичного дерева?

ГЛАВА V. ФОРМАЛЬНЫЕ ГРАММАТИКИ И АВТОМАТЫ

5.1. Языки и грамматики

Алфавит – это любое множество символов. Понятие символа не определяется. Цепочка символов 0, 1, 2 записывается как «012» (или 012). Другие обозначения:

xR – цепочка x с символами в обратном порядке;

x^n – цепочка x, повторенная n раз;

x^* – цепочка x, повторенная 0 или более раз;

$x+$ – цепочка x, повторенная 1 или более раз;

xy – сцепление (конкатенация) цепочек x и y;

$|x|$ – длина (число символов) цепочки x;

e – пустая цепочка.

Цепочку из одного символа будем обозначать самим символом. Буквы x, y, z, u, v, w, t будем применять для обозначения цепочек. Множество всех цепочек из элементов множества E естественно обозначить через E^*. Язык – это подмножество E^*. Примеры языков: Си, русский, {0 1 | n >= 0}.

Язык можно задать:

– перечислив все цепочки;

– написав программу-распознаватель, которая получает на вход цепочку символов и выдает ответ «да», если цепочка принадлежит языку и «нет» в противном случае;

– с помощью механизма порождения – грамматики.

Чтобы задать грамматику, требуется указать:

– множество символов алфавита (или терминальных символов) E. Будем обозначать их строчными символами алфавита и цифрами;

– множество нетерминальных символов (или метасимволов), не пересекающееся с E со специально выделенным начальным символом S. Будем обозначать их прописными буквами;

– множество правил вывода, определяющих правила подстановки для цепочек.

Каждое правило состоит из двух цепочек (например, *x* и *y*), причем *x* должна содержать, по крайней мере, один нетерминал, и означает, что цепочку *x* в процессе вывода можно заменить на *y*. Вывод цепочек языка начинается с нетерминала *S*. Правило грамматики будем записывать в виде $x : y$. (Также употребляется запись
$x ::= y$ или $x \rightarrow y$).

Более строго определим понятие выводимой цепочки:

– *S* – выводимая цепочка;

– если *xyz* – выводимая цепочка и в грамматике имеется правило $y : t$, то *xtz* – выводимая цепочка;

– определяемый грамматикой язык состоит из выводимых цепочек, содержащих только терминальные символы.

Примеры:

а) $S : e$ б) $S : e$
$S : 0S1$ $S : (S)$
$S : SS$.

Для сокращения записи принято использовать символ «или» – «|». Короткая форма записи предыдущих примеров:

а) $S : e \mid 0S1$ б) $S : e \mid (S) \mid SS$.

Более сложный пример:

в) $S : aSBC \mid abC$

CB : BC
bB : bb
cC : cc
bC : bc

Эта грамматика порождает язык $a^n\ b^n\ c^n$.

Грамматики в свою очередь образуют метаязык. Выше была описана «академическая» форма записи метаязыка. На практике применяется также другая форма записи, традиционно называемая нормальными формами Бэкуса–Наура (НБНФ).

Терминалы в НБНФ записываются как обычные символы алфавита, а нетерминалы – как имена в угловых скобках < >. Например, грамматику целых чисел без знака можно записать в виде:

<число> : <цифра> | <цифра> <число>

<цифра> : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Рассмотрим язык простейших арифметических формул:

<формула> : (<формула>) | <число> | <формула><знак><формула> <знак> : + | *

Почему «3+5*2» является формулой? Приведем последовательность преобразований цепочек (так называемый «разбор» или «вывод»):

<формула> : <формула> <знак><формула> :
<формула><знак><формула><знак><формула> :
<число> <знак><формула><знак><формула> :
3 <знак><формула><знак><формула> :
3 + <формула><знак><формула> :
3 + <число> <знак><формула> :
3 + 5 <знак><формула> :
3 + 5 * <формула> :

3 + 5 * <число> :
3 + 5 * 2

Сокращенно наличие вывода (цепочки преобразований) будем записывать в виде <формула> :: 3 + 5 * 2. Большинство грамматик допускают несколько различных выводов для одной и той же цепочки из языка.

Если в процессе вывода цепочки правила грамматики применяются только к самому левому нетерминалу, говорят, что получен левый вывод цепочки. Аналогично определяется правый вывод. Какой вывод построен в предыдущем примере?

Изобразим выполняемые замены цепочек в виде т.н. «дерева разбора» (или дерева вывода). По традиции дерево изображается снизу вверх (см. рисунок 5.1).

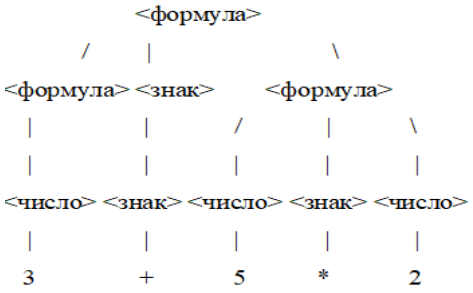


Рис. 5.1. Дерево раздора

Нарисованное дерево имеет ветви (линии) и узлы (помечены терминалами и нетерминалами), из которых растут ветви. Конечные узлы (терминалы) называются листьями. Понятия «поддерево», «корень дерева», видимо, не нуждаются в определении.

Одно и то же дерево разбора может описывать различные выводы (в дереве не фиксирован порядок применения правил). Однако между левыми (или правыми)

выводами и деревьями разбора для цепочек существует однозначное соответствие.

Если для одной и той же цепочки можно изобразить два разных дерева разбора (или, что тоже самое, построить, два разных правых вывода), грамматика называется неоднозначной. Описанная грамматика неоднозначна. Тот же самый язык можно описать однозначной грамматикой:

<формула> : <терм> | <терм><знак><формула>

<терм> : (<формула>) | <число>

<знак> : + | *

Дерево разбора определяет некоторую структуру цепочки языка. Так, мы видим, что подцепочка «3+5» является «формулой». Это противоречит нашим (интуитивным) понятиям о смысле исходной формулы: 3+5 в отличие от 5*2 не является подвыражением.

Мы можем учесть приоритет операций, изменив грамматику:

<формула> : <терм> | <формула> + <терм>

<терм> : <элемент> | <терм> * <элемент>

<элемент> : (<формула>) | <число>

Кроме привычной формы записи арифметических формул (так называемой «инфиксной», то есть со знаком операции между операндами), широко распространена «постфиксная» (или обратная польская) форма записи, в которой операция расположена после операндов.

Примеры:

2+3	2 3 +
2*3+4	2 3 * 4 +
2*(3+4)	2 3 4 + *

В обратной польской записи скобки не требуются, а значение формулы очень легко вычислить при использовании стека чисел.

5.2. Иерархия Хомского. Регулярные языки

Классификация грамматик по сложности соответствующих программ-распознавателей называется иерархией Хомского. В ней выделены 4 класса грамматик (в порядке возрастания сложности):

а) регулярные (или автоматные). Правила имеют вид: $A : xB$ или $A : x$, где x – цепочка терминалов или e;

б) контекстно-свободные (или КС). Правила имеют вид: $A : y$, где y – цепочка из терминалов и нетерминалов;

Примеры – «скобочный язык» из предыдущей главы, язык арифметических формул;

в) контекстно-зависимые (неукорачивающие). Правила имеют вид: $z : y$, где z и y – цепочки из терминалов и нетерминалов, z содержит нетерминал, $|z| <= |y|$.

Пример: $a^n\ b^n\ c^n$;

г) без ограничений.

Класс языка определяется классом самой простой (в смысле иерархии Хомского) из описывающих его грамматик. Следующие вложения для классов языков очевидны, если не рассматривать КС-грамматики, содержащие так называемые e-правила – правила с пустой правой частью.

а < б < в < рекурсивные множества < г = рекурсивные перечислимые множества.

Для языка, определенного регулярной грамматикой, всегда можно написать контекстно-свободную грамматику (и даже грамматику без ограничений!). Тем не менее,

все вложения строгие: в каждом классе существуют языки, которые нельзя задать грамматиками более простого класса.

5.3. Конечные автоматы

Рассмотрим другой способ задания регулярных языков. Конечный автомат (КА) задается:

– алфавитом входных символов E;

– множеством состояний S;

– тернарным отношением переходов на множестве $\{S, E\ U\ e, S\}$;

– начальным состоянием – выделенным состоянием в S;

– конечными состояниями – непустым подмножеством S.

Принято изображать автомат в виде ориентированного графа, узлы которого соответствуют состояниям (конечные состояния мы будем заключать в двойную рамку), а ребра, помеченные символами входного алфавита или e, изображают отношение переходов.

Цепочка допускается автоматом, если и только если существует пусть из начального в одно из конечных состояний, такой что, прочитав метки ребер вдоль этого пути, мы получим исходную цепочку (буква e, естественно, не читается). Например, автомат, изображенный на рисунке 5.2 допускает цепочки (01)+. Этот язык можно описать регулярной грамматикой: $S : 0\ B$ B : 1 C $C : 0\ B\ C : e$

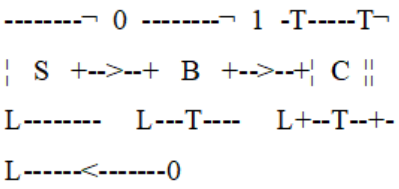


Рис. 5.2. Пример автомата

Несложно показать, что для каждого автомата можно построить регулярную грамматику, описывающую тот же самый язык, и наоборот.

Детерминированный конечный автомат – частный случай недетерминированного, в котором:

– нет *e*-переходов;

– отношение переходов является однозначной функцией $f : (S \times E \cup e) \to S$, определенной, может быть, не для всех пар из $S \times E \cup e$.

В терминах графа это означает, что из одного состояния выходит не более одного ребра с одинаковой меткой.

Для детерминированного автомата очень просто проверять принадлежность цепочки из E^* языку. Добавив, если нужно, еще одно «тупиковое» состояние, можно сделать функцию переходов определенной на всех парах из $S \times E$.

Такая интерпретация детерминированного конечного автомата более наглядна и общепринята: КА – устройство, которое может находится в конечном множестве состояний, и переходит из одного состояния в другое под действием внешних событий из алфавита E.

Можно ли подобным образом интерпретировать недетерминированный автомат (или хотя бы эффективно определять принадлежность цепочки языку)? Да, можно считать, что в том случае, когда возможен более чем один

переход, создается необходимое число экземпляров КА, которые переводятся во все возможные в этой ситуации состояния. Цепочка считается допущенной, если хотя бы один из экземпляров оказался в конечном состоянии.

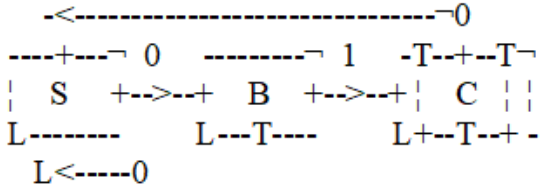


Рис. 5.3. Пример недетерминированного автомата

5.3.1. Преобразование недетерминированного КА в детерминированный

Недетерминированный КА всегда можно преобразовать в эквивалентный (т.е. допускающий то же множество цепочек) детерминированный КА. Рассмотрим новый КА, состояниями которого будут подмножества множества состояний исходного КА.

Исходным для нового автомата будет состояние {Начало}, конечными – все состояния, содержащие исходное конечное состояние. Переход A –> B по символу d имеется в новом автомате тогда и только тогда, когда в исходном автомате для любого состояния b из B, существует а из A такое, что по символу d возможен переход a –> b, и других переходов по d из А нет. Новый автомат будет детерминированным и эквивалентным исходному.

Действительно, состояние нового автомата $a + b$ соответствуют размещению экземпляров исходного недетерминированного КА в состояниях a и b, а переход в новом

автомате соответствует переходам всех экземпляров недетерминированного КА.

5.3.2. Минимизация конечного автомата

По конечному автомату часто можно построить автомат с меньшим числом состояний, эквивалентный исходному. Соответствующий процесс называется минимизацией конечного автомата. Вначале убирают из автомата все состояния, недостижимые из начального.

Затем разобивают все состояния КА на классы эквивалентности следующим способом: в первый класс отнесем все конечные состояния, а во второй – все остальные.

Эти состояния называются 0-эквивалентными. Теперь надо построить новое 1-эквивалентное разбиение, выделив те состояния, которые по одинаковым символам переходят в 0-эквивалентные состояния. Далее последовательно строятся n+1-эквивалентные состояния по n-эквивалентным, увеличивая число классов эквивалентности. Прекращается этот процесс тогда, когда n+1-эквивалентное состояние совпадет с n-эквивалентным. Каждый полученный класс эквивалентности и будет состоянием нового минимизированного КА, эквивалентного исходному.

Выше рассмотрены 4 способа описания языков: регулярные (автоматные, праволинейные) грамматики, недетерминированные КА, детерминированные КА, регулярные выражения. Они описывают один класс языков – регулярные языки. Этот класс языков «устроен очень хорошо»: для всех типичных вопросов известны ответы и эффективные алгоритмы. Примеры таких вопросов:

– является ли объединение, пересечение, дополнение регулярных языков регулярным;

– является ли регулярный язык конечным, пустым;

– совпадают ли два регулярных языка;

– является ли один регулярный язык подмножеством другого, и др.

Замечание. Для других классов иерархии Хомского дела обстоят значительно хуже, например, проблема эквивалентности для КС-языков алгоритмически неразрешима.

5.4. Контекстно-свободные языки

Класс контекстно-свободных языков допускает распознавание с помощью недетерминированного КА со стековой (или магазинной) памятью.

Контекстно-зависимые языки – последний класс языков, которые можно эффективно распознать с помощью компьютера. Они допускаются двусторонними недетерминированными линейно ограниченными автоматами. Для допуска цепочек языка без ограничений в общем случае требуется универсальный вычислитель (машина Тьюринга, машина с неограниченным числом регистров и др.).

Контекстно-зависимые языки и языки без ограничений не используются при построении компиляторов.

Автомат со стековой памятью в отличие от обычного КА имеет стек, в который можно помещать специальные «магазинные» символы. Переход из одного состояния в другое зависит не только от входного символа, но и от верхних символов магазина (на рисунке в круглых скоб-

ках). В начале работы в магазине лежит специальный символ *S*.

При выполнении перехода из магазина удаляются верхние символы, соответствующие правилу, и добавляется цепочка символов, соответствующих переходу (на рисунке изображены в квадратных скобках, первый символ цепочки становится верхним символом магазина). Допускаются также переходы, при которых входной символ игнорируется (и, тем самым, будет входным символом при следующем переходе). Эти переходы называются *e*-переходами. Автомат называется недетерминированным, если при одних и тех же состоянии, входном символе и вершине магазина возможен более чем один переход. Если при окончании цепочки автомат находится в одном из конечных состояний, а стек пуст, цепочка считается допущенной (после окончания цепочки могут быть сделаны е-переходы). Пример автомата со стековой памятью представлен на рисунке 5.4.

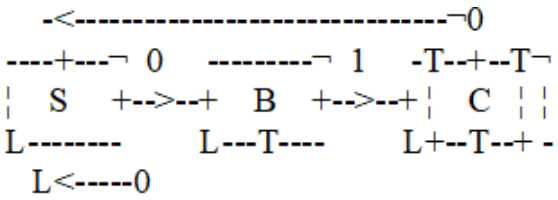


Рис. 5.4. Пример автомата со стековой памятью

По контекстно-свободной грамматике легко строится недетерминированный КА с магазинной памятью, который допускает цепочки этого языка. Он использует только одно состояние и следующий набор переходов:

1) *e*(*A*) [*x*] для каждого правила грамматики *A* : *x*;

2) $a(a)$ [] для каждого терминала a.

Можно построить и другой автомат, который также содержит одно состояние и имеет следующие переходы:

1) $a(e)$ [a] для каждого терминала a;

2) $e(x)$ [A] для каждого правила грамматики $A : x$.

Удобно считать, что в начале разбора магазин этого автомата пуст. Тогда в конце разбора в магазине останется символ S.

По недетерминированному КА с магазинной памятью можно построить КС-грамматику. Таким образом, класс КС-языков и класс языков, допускаемых автоматами с магазинной памятью, эквивалентны.

К сожалению, не каждый КС-язык допускает разбор с помощью детерминированного автомата. Например, язык цепочек-палиндромов из 0 и 1 не может быть допущен детерминированным КА с магазинной памятью. Таким образом, недетерминированные автоматы со стеком могут распознавать более широкий класс языков, чем детерминированные автоматы со стеком – в этом их существенное отличие от КА. Практический интерес для реализации компиляторов представляют детерминированные КС-языки – собственное подмножество КС-языков, допускающее распознавание с помощью детерминированных автоматов с магазинной памятью.

Два (недетерминированных) автомата, построенных выше для КС-грамматики, определяют два класса методов разбора КС-языков: сверху вниз, снизу вверх.

В первом случае (пример – рекурсивный спуск) при просмотре цепочки слева направо, естественно, будут получаться левые выводы. При детерминированном разбо-

ре проблема будет состоять в том, какое правило применить для раскрытия самого левого нетерминала.

Во втором случае детерминированный разбор удобно формализовать в терминах «сдвиг» (перенос символа из входной цепочки в магазин) и «свертка» (применение к вершине магазина какого-либо правила грамматики). При просмотре входной цепочки слева направо при этом будут получаться правые выводы. Проблема в этом случае состоит в выборе между сдвигом и сверткой и в выборе правила для свертки.

5.5. Преобразование КС-грамматик

Проблема эквивалентности двух языков, описываемых различными КС-грамматиками, в общем случае неразрешима. Однако часто удается преобразовать грамматику к требуемому для того или иного детерминированного разбора виду не «испортив» описываемый ею язык.

Удаление бесполезных символов и правил

Назовем символ A бесполезным, если он не встречается ни в одном выводе цепочки терминалов из начального символа грамматики. Очевидно, что бесполезные символы и все содержащие их правила могут быть удалены из грамматики. Например, в грамматике

$S : a \mid A \,;\, A : Ab \,;\, B : b$

из нетерминала A нельзя вывести ни одной терминальной цепочки, а нетерминал B не может быть выведен из начального символа S.

Множество нетерминалов, из которых выводятся терминальные цепочки, легко вычислить:

K := { *A* | существует правило *A*:терминалы }; *K*1 := пусто;
цикл пока *K* != *K*1 выполнять
*K*1 := *K*
K := *K U* { *A* | существует правило
A : (нетерминалы из К1 и и терминалы) }
конец цикла.

Следствие: проблема пустоты КС-языка разрешима: язык пуст, если и только если *S* не принадлежит *K*.

После удаления из грамматики нетерминалов, не принадлежащих *K*, несложно найти множество *L* небесполезных символов:

L := {*S*}; *L*1 := пусто;
цикл пока
L != *L*1 выполнять *L*1 := *L*
L := *L U* { *B* | сущ. правило *A*:..*B*.., и *A* принадлежит *L*1 }
конец цикла.

После удаления дополнения *L* из грамматики она не будет содержать бесполезных символов. Будут ли удалены все бесполезные символы из грамматики, если применить сначала второй, а затем первый алгоритм?

Устранение e-правил

Правило вида *A* : *e* будем называть *e*-правилом. Если язык не содержит пустой цепочки, то из грамматики можно удалить все e-правила, в противном случае можно ввести новый начальный символ *S*1, правило *S*1 : *S* | *e*, и удалить все остальные *e*-правила.

Следствие: КС-язык, не содержащий пустой цепочки, может быть описан неукорачивающей грамматикой.

Устранение циклов и цепных правил

Обозначим символом :: выводимость одной цепочки из другой с помощью непустой последовательности правил. Назовем циклом вывод вида A::A... . Для устранения циклов можно удалить из грамматики все цепные правила вида A:B. Для каждого нецепного правила вида A:... добавим в грамматику правила B:..., для всех B, из которых с помощью цепных правил выводится A. После этого удаление цепных правил не изменит языка. Грамматику без циклов, е-правил и лишних символов называют приведенной.

Устранение левой рекурсии

Нетерминал A называется рекурсивным, если существует вывод: если x = e, то A называется леворекурсивным, если y=e, то праворекурсивным.

Пример: A :: xAy

Метод рекурсивного спуска не работал для леворекурсивных грамматик. От левой рекурсии всегда можно избавиться (это не означает, что метод рекурсивного спуска применим к любому языку).

Покажем вначале, как устранить прямую левую рекурсию, т.е. правила вида A : A... . Пусть

$$A : Aa_1 \mid ... \mid Aa_n \mid b_1 \mid ... \mid b_n$$

все A-правила грамматики, и ни одна из цепочек b_i не начинается с A. Тогда, применяя A-правила к самому левому нетерминалу, из A можно вывести следующее регулярное множество цепочек:

$$(b_1 \mid ... \mid b_n)\,(a_1 \mid ... \mid a_n)^*$$

Это множество выводится и из преобразованной грамматики:

$$A : b_1 \mid \ldots \mid b_n \mid b_1T \mid \ldots \mid b_nT$$
$$T : a_1 \mid \ldots \mid a_n \mid a_1T \mid \ldots \mid a_nT$$

Простое и операторное предшествование

Рассмотрим разбор снизу-вверх, при котором промежуточные выводы перемещаются по дереву по направлению к корню. Если считывать символы цепочки слева направо, то дерево разбора будет выглядеть следующим образом:

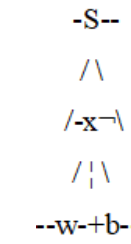


Рис. 5.5. Дерево раздора снизу-вверх

Промежуточный вывод имеет вид xbu, где x – цепочка терминалов и нетерминалов, из которой выводится просмотренная часть терминальной цепочки w, bu – непросмотренная часть терминальной цепочки, b – очередной символ. Чтобы продолжить разбор, можно либо добавить символ b к просмотренной части цепочки (выполнить так называемый «сдвиг»), либо выделить в конце x такую цепочку z ($x = yz$), что к z можно применить одно из правил грамматики $B : z$ и заменить x на цепочку yB (выполнить так называемую «свертку»):

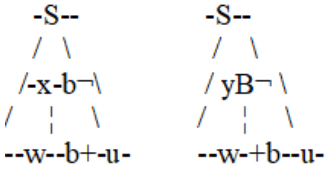


Рис.5.6. Дерево разбора после сдвига и после свертки

Если свертку применять только к последним символам *x*, то мы будем получать правые выводы цепочки. Такой разбор получил название LR, где символ *L* (*Left*, левый) относится к просмотру цепочки слева направо, а *R* (*Right*, правый) относится к получаемым выводам.

Пример: LR-разбор цепочки aabb в соответствии с грамматикой S : SaSb | e.

Символ _ указывает на границу между просмотренной и непросмотренной частями цепочки:

_aabb : *S_aabb* : *Sa_abb* : *Sa*S_*abb* : *SaSa_bb* : *SaSaS_bb* : *SaSaSb_b* : *SaS_b* : *SaSb_* : *S_* .

Последовательность операций сдвига и свертки существенна. Если бы в приведенном примере первой операцией был бы сдвиг, разбор не удалось бы довести до конца, поэтому для детерминированного разбора требуется в каждый момент выбирать между сдвигом и сверткой (и различными правилами свертки). Ниже будут рассмотрены два способа выбора: простое и операторное предшествования.

Грамматики простого предшествования

Грамматика называется обратимой, если в ней нет двух правил с одинаковыми правыми частями. Напомним, что грамматика называется приведенной, если в ней нет e-правил, бесполезных символов и циклов.

Зададимся целью ввести на множестве терминальных и нетерминальных символов три отношения (так называемые «отношения предшествования») (будем обозначать их <', =' и >') так, чтобы в момент, когда требуется свертка цепочки z, отношения между символами построенной части вывода (цепочки x в обозначениях из начала лекции) и очередным символом *b* были следующими:

<' или =' <' =' =' >'

L----------+--------+---+---------

y *z* *b* *u*

Если удастся построить такие отношения, то *LR*-разбор для обратимой грамматики можно проводить очень просто:

– сделать сдвиг, если последний символ *x* <' *b* или последний символ x =' b;

– сделать свертку, если последний символ *x* >' *b*, при этом правая часть правила z заключена между отношениями <' и >'.

Грамматика называется грамматикой простого предшествования, если она приведенная, обратимая и между любыми двумя терминалами или нетерминалами выполняется не более одного отношения предшествования.

Практически отношения предшествования можно вычислить следующим образом (*X*, *Y* – терминалы или нетерминалы, *A*, *B*, *C* –нетерминалы, *y* – терминал, ... – любая цепочка (может быть пустая)):

X =' *Y*, если в грамматике есть правило *A* : ...*XY*...

X <' *Y*, если в грамматике есть правило *A* : ...*XB*... и *B* :: *Y*...

X >' y, если в грамматике есть правило A : ...By... или A : ...BC... и B :: ...X и C :: y...

Вычисляя отношения предшествования для грамматики S : $aSSb \mid c$, получим: a ='S, S ='S, S ='b, {a, S} <' {a, c}, {b, c} >' {a, b, c}.

Эта грамматика является грамматикой простого предшествования.

На практике иногда удобно считать, что в начале и конце цепочки языка стоят символы-ограничители #. Для них отношения предшествования определяются так:

<' X, если S :: X ...

X >' #, если S :: ... X

В нашем примере {b, c} >' #, # <' {a, c}.

Отметим, что отношения <', >', =' не похожи на обычные арифметические отношения <, >, =: =' не является отношением эквивалентности, <' и >' не обязательно транзитивны. Отношения предшествования удобно занести в матрицу, в которой строки и столбцы помечены терминалами и нетерминалами грамматики.

Грамматики операторного предшествования

Если в правилах приведенной обратимой грамматики не встречаются рядом два нетерминала, говорят, что грамматика является операторной. Классический пример – грамматика арифметических формул. Для таких грамматик можно вычислять отношения предшествования только на множестве терминальных символов. Для этого модифицируем правила вычисления отношений предшествования:

1) a =' b, если в грамматике имеется правило A : ... ab... или A : ... aBb... ;

2) $a <' b$, если в грамматике имеется правило $A : ... aB ...$ и $B :: b ...$ или $B :: Cb ...$;

3) $a >' b$, если в грамматике имеется правило $A : ... Bb...$ и $B :: ... a$ или $B :: ... aC$;

4) $\# <' a$, если в грамматике имеется вывод $S :: Ca ...$ или $S :: a ...$;

5) $a >' \#$, если в грамматике имеется вывод $S :: ... aC$ или $S :: ... a \ldots$.

Если между любыми двумя терминалами выполняется не более одного отношения предшествования, операторная грамматика называется грамматикой операторного предшествования.

Контрольные вопросы

1. Привести иерархию Хомского.
2. Как описывается недетерминированный конечный автомат?
3. Можно ли преобразовать недетерминированный конечный автомат в эквивалентный детерминированный конечный автомат?
4. Какой автомат называется детерминированным?
5. Как осуществляется перенос символа из входной цепочки в магазин?
6. Что собой представляет сдвиг, свертка, рекурсивный спуск?
7. Допускает ли класс контекстно-свободных языков распознавание с помощью недетерминированного конечного автомата со стековой (или магазинной) памятью.
8. Для чего используются стек выдачи, стек хранения, блок сравнения?

9. Какая грамматика называется обратимой, приведенной, операторной?

ГЛАВА VI. РАСПОЗНАВАНИЕ ОБРАЗОВ

6.1. Распознавание образов и анализ сцен

В последние годы распознавание образов находит все большее применение в повседневной жизни. Распознавание речи и рукописного текста значительно упрощает взаимодействие человека с компьютером, распознавание печатного текста используется для перевода документов в электронную форму. Популярно мнение, что распознавание, как и прочие алгоритмы искусственного интеллекта, есть черная магия, недоступная простым смертным. На самом же деле алгоритмы, лежащие в основе распознавания, довольно очевидны, нужно лишь зайти чуть издалека и определиться с терминами.

Базовым является неопределимое понятие множества. В компьютере множество представляется набором неповторяющихся однотипных элементов. Слово «неповторяющихся» означает, что какой-то элемент в множестве либо есть, либо его там нет. Универсальное множество включает все возможные для решаемой задачи элементы, пустое не содержит ни одного.

В классической постановке задачи распознавания универсальное множество разбивается на части-образы. Образ какого-либо объекта задается набором его частных проявлений. В случае с распознаванием текста в универсальное множество войдут все возможные знаки, в образ «Й» – все возможные начертания этой буквы, а программа распознавания занимается тем, что на основе небольшого набора примеров начертаний каждой буквы

(обучающей выборки) определяет, какую из них символизирует введенная закорючка.

Методика отнесения элемента к какому-либо образу называется решающим правилом. Еще одно важное понятие – метрика, способ определения расстояния между элементами универсального множества. Чем меньше это расстояние, тем более похожими являются символы, звуки – это то, что мы распознаем. Обычно элементы задаются в виде набора чисел, а метрика – в виде функции. От выбора представления образов и реализации метрики зависит эффективность программы, один алгоритм распознавания с разными метриками будет ошибаться с разной частотой (право на ошибку для программ распознавания так же характерно, как и для людей).

Показывает принцип работы распознавания образов элементарный алгоритм на основе метода множества эталонов. На входе его имеется обучающая выборка – набор примеров A'_{ij} для каждого образа A_i, метрика d и сам распознаваемый объект x. С помощью метрики вычисляем расстояние от x до каждого элемента обучающей выборки $d(x, a_{ij})$ и находим условное расстояние $d(x, A_i)$ как расстояние от x до ближайшего элемента из A_i. Элемент x относится к образу, который окажется ближе всех.

Практически требуется найти минимум расстояния по каждому классу и еще раз взять минимум. Любители трогать руками могут взять в качестве представления элемента пару координат, в качестве метрики – расстояние по теореме Пифагора, и набросать программку, которая будет выполнять описанную операцию над массивом точек двухмерного пространства и отображать это в графике.

Еще один элементарный алгоритм – метод k-ближайших соседей. Как следует из названия, в нем вводится дополнительный входной параметр, целое число k. Тут все еще проще – берется k ближайших к x элементов обучающей выборки и подсчитывается, сколько из них принадлежит к какому образу. К какому образу принадлежит больше, к тому относится и x.

В обоих алгоритмах может возникнуть неопределенная ситуация – когда x будет находиться на одинаковом расстоянии от нескольких образов. В таком случае программа должна либо спросить у пользователя, к какому образу относить элемент, либо тихо бросить жребий. Это зависит от требований к точности с одной стороны, и удобства использования с другой. Лучше всего реализовать оба варианта.

6.1.1. Понятие образа

Образ, класс – классификационная группировка в системе классификации, объединяющая (выделяющая) определенную группу объектов по некоторому признаку.

Образное восприятие мира – одно из загадочных свойств живого мозга, позволяющее разобраться в бесконечном потоке воспринимаемой информации, и сохранять ориентацию в океане разрозненных данных о внешнем мире. Воспринимая внешний мир, мы всегда производим классификацию воспринимаемых ощущений, то есть разбиваем их на группы похожих, но не тождественных явлений. Например, несмотря на существенное различие, к одной группе относятся все буквы В, написанные различными почерками, или все звуки, которые соответствуют одной и той же ноте, взятой в любой

октаве и на любом инструменте, а оператор, управляющий техническим объектом, на целое множество состояний объекта реагирует одной и той же реакцией. Характерно, что для составления понятия о группе восприятий определенного класса достаточно познакомиться с незначительным количеством ее представителей. Ребенку можно показать всего один раз какую-либо букву, чтобы он смог найти её в тексте, написанном различными шрифтами, или узнать ее, даже если она написана в умышленно искаженном виде. Это свойство мозга позволяет сформулировать такое понятие, как образ.

Образы обладают характерным свойством, проявляющимся в том, что ознакомление с конечным числом явлений из одного и того же множества дает возможность узнавать сколько угодно большое число его представителей. Примерами образов могут быть: река, море, жидкость, музыка Чайковского, стихи Маяковского и др. В качестве образа можно рассматривать и некоторую совокупность состояний объекта управления, причем вся эта совокупность состояний характеризуется тем, что для достижения заданной цели требуется одинаковое воздействие на объект. Образы обладают характерными объективными свойствами в том смысле, что разные люди, обучающиеся на различном материале наблюдений, большей частью одинаково и независимо друг от друга классифицируют одни и те же объекты. Именно эта объективность образов позволяет людям всего мира понимать друг друга.

Способность восприятия внешнего мира в форме образов позволяет с определенной достоверностью узнавать бесконечное число объектов на основании ознакомления

с конечным их числом, а объективный характер основного свойства образов позволяет моделировать процесс их распознавания. Будучи отражением объективной реальности, понятие образа столь же объективно, как и сама реальность, а поэтому может быть само по себе объектом специального исследования.

В литературе, посвященной проблеме обучения распознавания образов (ОРО), часто вместо понятия образа вводится понятие класса.

6.1.2. Проблема обучения распознаванию образов (ОРО)

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

Рассмотрим пример задач из области ОРО. На рисунке 6.1 представлены 12 изображений, и следует отобрать признаки, при помощи которых можно отличить левую триаду картинок от правой. Решение данных задач требует моделирования логического мышления в полном объеме.

В целом проблема распознавания образов состоит из двух частей: обучение и распознавание. Обучение осуществляется путем показа отдельных объектов с указанием их принадлежности тому или другому образу.

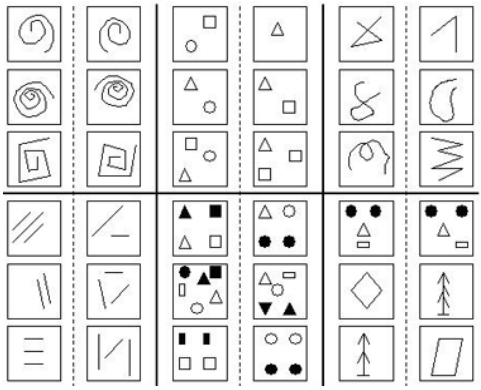


Рис. 6.1. Пример задач из области ОРО

В результате обучения распознающая система должна приобрести способность реагировать одинаковыми реакциями на все объекты одного образа и различными – на все объекты различных образов. Очень важно, что процесс обучения должен завершиться только путем показов конечного числа объектов без каких-либо других подсказок. В качестве объектов обучения могут быть либо картинки, либо другие визуальные изображения (буквы), либо различные явления внешнего мира, например, звуки, состояния организма при медицинском диагнозе, состояние технического объекта в системах управления и др. Важно, что в процессе обучения указываются только сами объекты и их принадлежность образу. За обучением следует процесс распознавания новых объектов, который характеризует действия уже обученной системы. Автоматизация этих процедур и составляет проблему обучения распознаванию образов. В том случае, когда человек сам разгадывает или придумывает, а затем навязывает машине правило классификации, проблема распознавания

решается частично, так как основную и главную часть проблемы (обучение) человек берет на себя.

Проблема обучения распознаванию образов интересна как с прикладной, так и с принципиальной точки зрения. С прикладной точки зрения решение этой проблемы важно, прежде всего потому, что оно открывает возможность автоматизировать многие процессы, которые до сих пор связывали лишь с деятельностью живого мозга. Принципиальное значение проблемы тесно связано с вопросом, который все чаще возникает в связи с развитием идей кибернетики: что может и что принципиально не может делать машина? В какой мере возможности машины могут быть приближены к возможностям живого мозга? В частности, может ли машина развить в себе способность перенять у человека умение производить определенные действия в зависимости от ситуаций, возникающих в окружающей среде? Пока стало ясно только то, что если человек может сначала сам осознать свое умение, а потом его описать, то есть указать, почему он производит действия в ответ на каждое состояние внешней среды или по какому правилу он объединяет отдельные объекты в образы, то такое умение без принципиальных трудностей может быть передано машине. Если же человек обладает умением, но не может объяснить его, то остается только один путь передачи умения машине – обучение примерами.

Круг задач, которые могут решаться с помощью распознающих систем, чрезвычайно широк. Сюда относятся не только задачи распознавания зрительных и слуховых образов, но и задачи распознавания сложных процессов и явлений, возникающих, например, при выборе целесо-

образных действий руководителем предприятия или выборе оптимального управления технологическими, экономическими, транспортными или военными операциями. В каждой из таких задач анализируются некоторые явления, процессы, состояния внешнего мира, всюду далее называемые объектами наблюдения. Прежде чем начать анализ какого-либо объекта, нужно получить о нем определенную, каким-либо способом упорядоченную информацию. Такая информация представляет собой характеристику объектов, их отображение на множестве воспринимающих органов распознающей системы.

Но каждый объект наблюдения может воздействовать на нас по-разному, в зависимости от условий восприятия. Например, какая-либо буква, даже одинаково написанная, может в принципе как угодно смещаться относительно воспринимающих органов. Кроме того, объекты одного и того же образа могут достаточно сильно отличаться друг от друга и, естественно, по-разному воздействовать на воспринимающие органы.

Каждое отображение какого-либо объекта на воспринимающие органы распознающей системы, независимо от его положения относительно этих органов, принято называть изображением объекта, а множества таких изображений, объединенные какими-либо общими свойствами, представляют собой образы.

При решении задач управления методами распознавания образов вместо термина «изображение» применяют термин «состояние». Состояние – это определенной формы отображение измеряемых текущих (или мгновенных) характеристик наблюдаемого объекта. Совокупность состояний определяет ситуацию. Понятие «ситуация» яв-

ляется аналогом понятия «образ». Но эта аналогия не полная, так как не всякий образ можно назвать ситуацией, хотя всякую ситуацию можно назвать образом.

Ситуацией принято называть некоторую совокупность состояний сложного объекта, каждая из которых характеризуется одними и теми же или схожими характеристиками объекта. Например, если в качестве объекта наблюдения рассматривается некоторый объект управления, то ситуация объединяет такие состояния этого объекта, в которых следует применять одни и те же управляющие воздействия. Если объектом наблюдения является военная игра, то ситуация объединяет все состояния игры, которые требуют, например, мощного танкового удара при поддержке авиации.

Выбор исходного описания объектов является одной из центральных задач проблемы ОРО. При удачном выборе исходного описания (пространства признаков) задача распознавания может оказаться тривиальной, и наоборот, неудачно выбранное исходное описание может привести либо к очень сложной дальнейшей переработке информации, либо вообще к отсутствию решения. Например, если решается задача распознавания объектов, отличающихся по цвету, а в качестве исходного описания выбраны сигналы, получаемые от датчиков веса, то задача распознавания в принципе не может быть решена.

6.1.3. Геометрический и структурный подходы

Каждый раз, когда сталкиваешься с незнакомыми задачами, появляется естественное желание представить их в виде некоторой легко понимаемой модели – она поз-

волила бы осмыслить задачу в таких терминах, которые легко воспроизводятся нашим воображением. А так как мы существуем в пространстве и во времени, наиболее понятной для нас является пространственно-временная интерпретация задач.

Любое изображение, которое возникает в результате наблюдения какого-либо объекта в процессе обучения или экзамена, можно представить в виде вектора, а значит, и в виде точки некоторого пространства признаков. Если утверждается, что при показе изображений возможно однозначно отнести их к одному из двух (или нескольких) образов, то тем самым утверждается, что в некотором пространстве существует две (или несколько) области, не имеющие общих точек, и что изображения – точки из этих областей. Каждой такой области можно приписать наименование, т.е. дать название, соответствующее образу.

Проинтерпретируем теперь в терминах геометрической картины процесс обучения распознаванию образов, ограничившись пока случаем распознавания только двух образов. Заранее считается известным лишь то, что требуется разделить две области в некотором пространстве и что показываются точки только из этих областей. Сами эти области заранее не определены, т.е. нет каких-либо сведений о расположении их границ или правил определения принадлежности точки к той или иной области.

В ходе обучения предъявляются точки, случайно выбранные из этих областей, и сообщается информация о том, к какой области принадлежат предъявляемые точки. Никакой дополнительной информации об этих областях, то есть о расположении их границ, в ходе обучения

не сообщается. Цель обучения состоит либо в построении поверхности, которая разделяла бы не только показанные в процессе обучения точки, но и все остальные точки, принадлежащие этим областям, либо в построении поверхностей, ограничивающих эти области так, чтобы в каждой из них находились только точки одного образа. Иначе говоря, цель обучения состоит в построении таких функций от векторов-изображений, которые были бы, например, положительны на всех точках одного и отрицательны на всех точках другого образа. В связи с тем, что области не имеют общих точек, всегда существует целое множество таких разделяющих функций, а в результате обучения должна быть построена одна из них.

Если предъявляемые изображения принадлежат не двум, а большему числу образов, то задача состоит в построении по показанным в ходе обучения точкам поверхности, разделяющей друг от друга все области, которые соответствуют этим образам. Задача эта может быть решена, например, путем построения функции, принимающей над точками каждой из областей одинаковое значение, а над точками из разных областей значение этой функции должно быть различно.

На первый взгляд, кажется, что знание всего лишь некоторого количества точек из области недостаточно, чтобы отделить всю область. Действительно, можно указать бесчисленное количество различных областей, которые содержат эти точки, и, как бы ни была построена по ним поверхность, выделяющая область, всегда можно указать другую область, которая пересекает поверхность и вместе с тем содержит показанные точки.

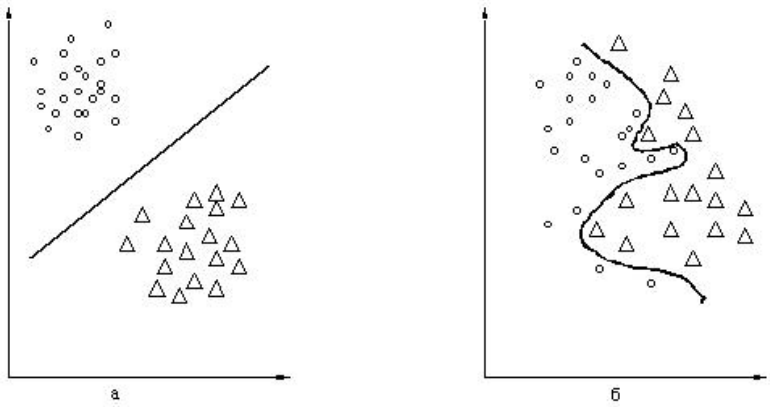


Рис. 6.2. Примеры разделения
областей просмотра изображения

Однако известно, что задача о приближении функции по информации о ней в ограниченном множестве точек, существенно более узкой, чем все множество, на котором функция задана, является обычной математической задачей об аппроксимации функций. Разумеется, решение таких задач требует введения определенных ограничений на классе рассматриваемых функций, а выбор этих ограничений зависит от характера информации, которую может добавить учитель в процессе обучения. Одной из таких подсказок является гипотеза о компактности образов. Интуитивно ясно, что аппроксимация разделяющей функции будет задачей тем более легкой, чем более компактны и чем более разнесены в пространстве области, подлежащие разделению. Так, например, в случае, показанном на 6.2а, разделение заведомо более просто, чем в случае, показанном на 6.2б. Действительно, в случае, изображенном на 6.2а, области могут быть разделены плоскостью, и даже при больших погрешностях в

определении разделяющей функции она все же будет продолжать разделять области. В случае же на 6.2б, разделение осуществляется замысловатой поверхностью, и даже незначительные отклонения в ее форме приводят к ошибкам разделения. Именно это интуитивное представление о сравнительно легко разделимых областях привело к гипотезе компактности.

Наряду с геометрической интерпретацией проблемы обучения распознаванию образов существует и иной подход, который назван структурным, или лингвистическим. Поясним его на примере распознавания зрительных изображений. Сначала выделяется набор исходных понятий – типичных фрагментов, встречающихся на изображениях, и характеристик взаимного расположения фрагментов – «слева», «снизу», «внутри» и др. Эти исходные понятия образуют словарь, который позволяет строить различные логические высказывания, иногда называемые предположениями. Задача состоит в том, чтобы из большого количества высказываний, которые могли бы быть построены с использованием этих понятий, отобрать наиболее существенные для данного конкретного случая.

Далее, просматривая конечное и по возможности небольшое число объектов из каждого образа, нужно построить описание этих образов. Построенные описания должны быть столь полными, чтобы решить вопрос о том, к какому образу принадлежит данный объект. При реализации лингвистического подхода возникают две задачи: задача построения исходного словаря, т. е. набор типичных фрагментов, и задача построения правил описания из элементов заданного словаря.

В рамках лингвистической интерпретации проводится аналогия между структурой изображений и синтаксисом языка. Стремление к этой аналогии было вызвано возможностью использовать аппарат математической лингвистики, то есть методов, по своей природе являющихся синтаксическими. Использование аппарата математической лингвистики для описания структуры изображений можно применять только после того, как произведена сегментация изображений на составные части, то есть выработаны слова для описания типичных фрагментов и методы их поиска. После предварительной работы, обеспечивающей выделение слов, возникают собственно лингвистические задачи, состоящие из задач автоматического грамматического разбора описаний для распознавания изображений. При этом проявляется самостоятельная область исследований, которая требует не только знания основ математической лингвистики, но и владения приемами, разработанными специально для лингвистической обработки изображений.

Гипотеза компактности

Если предположить, что в процессе обучения пространство признаков формируется, исходя из задуманной классификации, то тогда можно надеяться, что задание пространство признаков само по себе задает свойство, под действием которого образы в этом пространстве легко разделяются. Именно эти надежды по мере развития работ в области распознавания образов стимулировали появление гипотезы компактности, которая гласит: образам соответствуют компактные множества в пространстве признаков. Под компактным множеством пока будем по-

нимать некие «сгустки» точек в пространстве изображений, предполагая, что между этими сгустками существуют разделяющие их разряжения.

Однако эту гипотезу не всегда удавалось подтвердить экспериментально, но, что самое главное, те задачи, в рамках которых гипотеза компактности хорошо выполнялась (рис. 6.2а), все без исключения находили простое решение. И наоборот, те задачи, для которых гипотеза не подтверждалась (рис. 6.2б), либо совсем не решались, либо решались с большим трудом. Этот факт заставил усомниться в справедливости гипотезы компактности, так как для опровержения любой гипотезы достаточно одного отрицающего ее примера. Вместе с этим, выполнение гипотезы всюду там, где удавалось хорошо решить задачу обучения распознаванию образов, сохраняло к ней интерес. Сама гипотеза компактности превратилась в признак возможности удовлетворительного решения задач распознавания.

Формулировка гипотезы компактности подводит вплотную к понятию абстрактного образа. Если координаты пространства выбирать случайно, то и изображения в нем будут распределены случайно. Они будут в некоторых частях пространства располагаться более плотно, чем в других. Назовем некоторое случайно выбранное пространство абстрактным изображением. В этом абстрактном пространстве почти наверняка будут существовать компактные множества точек. Поэтому в соответствии с гипотезой компактности множества объекты, которым в абстрактном пространстве соответствуют компактные множества точек, разумно назвать абстрактными образами данного пространства.

6.1.4. Обучение и самообучение

Если бы удалось подметить некое всеобщее свойство, не зависящее ни от природы образов, ни от их изображений, а определяющее лишь их способность к разделимости, то наряду с обычной задачей обучения распознаванию, с использованием информации о принадлежности каждого объекта из обучающей последовательности тому или иному образу, можно было бы поставить иную классификационную задачу – так называемую задачу обучения без учителя. Задачу такого рода на описательном уровне можно сформулировать так: системе одновременно или последовательно предъявляются объекты без каких-либо указаний об их принадлежности к образам. Входное устройство системы отображает множество объектов на множество изображений и, используя некоторое заложенное в нее заранее свойство разделимости образов, производит самостоятельную классификацию этих объектов. После такого процесса самообучения система должна приобрести способность к распознаванию не только уже знакомых объектов (объектов из обучающей последовательности), но и тех, которые ранее не предъявлялись. Процессом самообучения некоторой системы называется такой процесс, в результате которого эта система без подсказки учителя приобретает способность к выработке одинаковых реакций на изображения объектов одного и того же образа, и различных реакций на изображения различных образов. Роль учителя при этом состоит лишь в подсказке системе некоторого объективного свойства, одинакового для всех образов и определяющего способность к разделению множества объектов на образы.

Оказывается, таким объективным свойством является свойство компактности образов. Взаимное расположение точек в выбранном пространстве уже содержит информацию о том, как следует разделить множество точек. Эта информация и определяет то свойство разделимости образов, которое оказывается достаточным для самообучения системы распознаванию образов.

Большинство известных алгоритмов самообучения способны выделять только абстрактные образы, то есть компактные множества в заданных пространствах. Различие между ними состоит, по-видимому, в формализации понятия компактности. Однако это не снижает, а иногда и повышает ценность алгоритмов самообучения, так как часто сами образы заранее никем не определены, а задача состоит в том, чтобы определить, какие подмножества изображений в заданном пространстве представляют собой образы. Хорошим примером такой постановки задачи являются социологические исследования, когда по набору вопросов выделяются группы людей. В таком понимании задачи алгоритмы самообучения генерируют заранее не известную информацию о существовании в заданном пространстве образов, о которых ранее никто не имел никакого представления.

Кроме того, результат самообучения характеризует пригодность выбранного пространства для конкретной задачи обучения распознаванию. Если абстрактные образы, выделяемые в процессе самообучения, совпадают с реальными, то пространство выбрано удачно. Чем сильнее абстрактные образы отличаются от реальных, тем «неудобнее» выбранное пространство для конкретной задачи.

Обучением обычно называют процесс выработки в некоторой системе той или иной реакции на группы внешних идентичных сигналов путем многократного воздействия на систему внешней корректировки. Такую внешнюю корректировку в обучении принято называть «поощрениями» и «наказаниями». Механизм генерации этой корректировки практически полностью определяет алгоритм обучения. Самообучение отличается от обучения тем, что здесь дополнительная информация о верности реакции системе не сообщается.

6.2. Общая характеристика задач распознавания образов и их типы

Под образом понимается структурированное описание изучаемого объекта или явления, представленное вектором признаков, каждый элемент которого представляет числовое значение одного из признаков, характеризующих соответствующий объект. Общая структура системы распознавания и этапы в процессе ее разработки показаны на рис. 6.3.

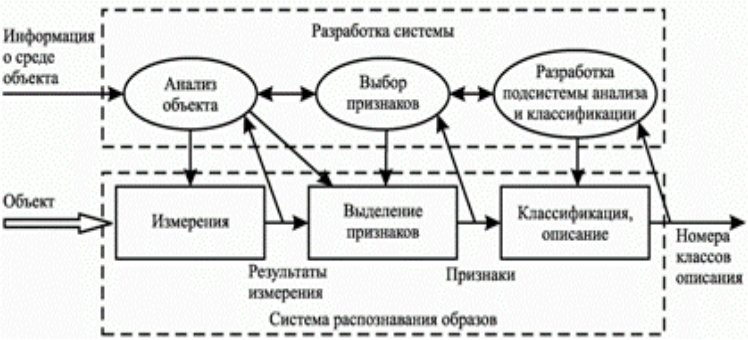


Рис. 6.3. Структура системы распознавания

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

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

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

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

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

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

Целесообразно выделить следующие типы задач распознавания.

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

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

3. Задача выбора информативного набора признаков при распознавании.

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

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

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

6.2.1. Основы теории анализа и распознавания изображений

Пусть дано множество M объектов; на этом множестве существует разбиение на конечное число подмножеств (классов) Ω_i, $i = \{1, m\}$, $M = \cup\, \Omega_i \quad (i = 1, ..., m)$. Объекты ω задаются значениями некоторых признаков x_j, $j = \{1, N\}$. Описание объекта $I(\omega) = (x_1(\omega), ..., x_N(\omega))$ называют стандартным, если $x_j(\omega)$ принимает значение из множества допустимых значений.

Пусть задана таблица обучения (таблица 17). Задача распознавания состоит в том, чтобы для заданного объекта ω и набора классов $\Omega_1, ..., \Omega_m$ по обучающей инфор-

мации в таблице обучения $I_0(\Omega_1, ..., \Omega_m)$ о классах и описанию $I(\omega)$ вычислить предикаты:

$$P_i(\omega \in \Omega_i) = \{1(\omega \in \Omega_i), 0(\omega \in \Omega_i), (\omega \in \Omega_i)\},$$

где $i = \{1, m\}$, Δ – неизвестно.

Таблица 17. Таблица обучения

Объект	Признаки и их значения			Класс
	x_1	x_j	x_n	
ω_1	α_{11}	α_{1j}	α_{1n}	Ω_1
...				
ω_{r1}	α_{r11}	α_{r1j}	α_{r1n}	
...				
ω_{rk}	α_{rk1}	α_{rkj}	α_{rkn}	Ω_m
...				
ω_{rm}	α_{rm1}	α_{rmj}	α_{rmn}	

Рассмотрим алгоритмы распознавания, основанные на вычислении оценок. В их основе лежит принцип прецедентности (в аналогичных ситуациях следует действовать аналогично).

Пусть задан полный набор признаков $x_1, ..., x_n$. Выделим систему подмножеств множества признаков $S_1, ..., S_k$. Удалим произвольный набор признаков из строк $\omega_1, \omega_2, ..., \omega_{rm}$ и обозначим полученные строки через $S\omega_1, S\omega_2, ..., S\omega_{rm}, S\omega'$.

Правило близости, позволяющее оценить похожесть строк $S\omega'$ и $S\omega_r$ состоит в следующем. Пусть «усеченные» строки содержат q первых символов, то есть $S\omega_r = (a_1, ..., a_q)$ и $S\omega' = (b_1, ..., b_q)$. Заданы пороги $\varepsilon_1...\varepsilon_q, \delta$. Строки $S\omega_r$ и $S\omega'$ считаются похожими, если выполняется не менее чем δ неравенств вида

$$| a_j - b_j | <= \varepsilon_j, \; j = 1, 2, ..., q.$$

Величины $\varepsilon_1 ... \varepsilon_q$, δ входят в качестве параметров в модель класса алгоритмов на основе оценок.

Пусть $\Gamma_i(\omega')$ – оценка объекта ω' по классу Ω_i.

Описания объектов $\{\omega'\}$, предъявленные для распознавания, переводятся в числовую матрицу оценок. Решение о том, к какому классу отнести объект, выносится на основе вычисления степени сходства распознавания объекта (строки) со строками, принадлежность которых к заданным классам известна.

Проиллюстрируем описанный алгоритм распознавания на примере. Задано 10 классов объектов (рис. 6.4*а*). Требуется определить признаки таблицы обучения, пороги и построить оценки близости для классов объектов, показанных на рис. 6.4*б*. Предлагаются следующие признаки таблицы обучения:

x_1 – количество вертикальных линий минимального размера;

x_2 – количество горизонтальных линий;

x_3 – количество наклонных линий;

x_4 – количество горизонтальных линий снизу объекта.

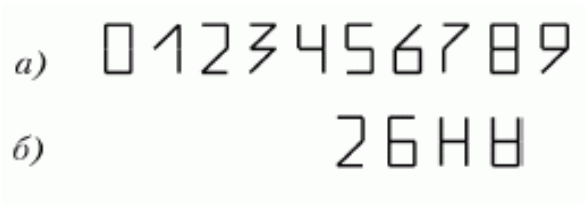


Рис. 6.4. Пример задачи распознавания

На рис. 6.5 приведена таблица обучения и пороги $\varepsilon_1 = 1$, $\varepsilon_2 = 1$, $\varepsilon_3 = 1$, $\varepsilon_4 = 1$, $\delta = 1$.

Из этой таблицы видно, что неразличимость символов 6 и 9 привела к необходимости ввода еще одного признака x_4.

	X_1	X_2	X_3	X_4	
	4	2	0		0
	2	0	1		1
	1	2	1		2
	0	2	2		3
	3	1	0		4
	2	3	0		5
✓	2	2	1	1	6
	1	1	1		7
	4	3	0		8
✓	2	2	1	0	9
	$\varepsilon_1=1$	$\varepsilon_2=1$	$\varepsilon_3=1$	$\varepsilon_4=1$	$\delta=1$

Рис. 6.5. Таблица обучения для задачи распознавания

Теперь может быть построена таблица распознавания для объектов на рис. 6.4.

Таблица 18. Таблица распознавания

Объект	*x*1	*x*2	*x*3	x4	Результат распознавания
Объект 1	1	2	1		Цифра 2
Объект 2	3	3			Цифра 8 или 5
Объект 3	4	1			Цифра 0 или 4
Объект 4	4	2			Цифра 0 или 8

Качество распознавания во многом зависит от того, насколько удачно создан алфавит признаков, придуманный разработчиками системы. Поэтому признаки должны быть инвариантны к ориентации, размеру и вариациям формы объектов.

6.2.2. Распознавание по методу аналогий

Рассмотрим этот метод на примере задачи П. Уинстона [18] по поиску геометрических аналогий, представленном на рис. 6.6. Среди фигур второго ряда требуется выбрать $X \in \{1, 2, 3, 4, 5\}$ такое, что A так соотносится с B, как C соотносится с X, и такое, которое лучше всего при этом подходит. Для решения задачи необходимо понять, в чем разница между фигурами A и B (наличие/отсутствие жирной точки), и после этого ясно, что лучше всего для C подходит $X = 3$.

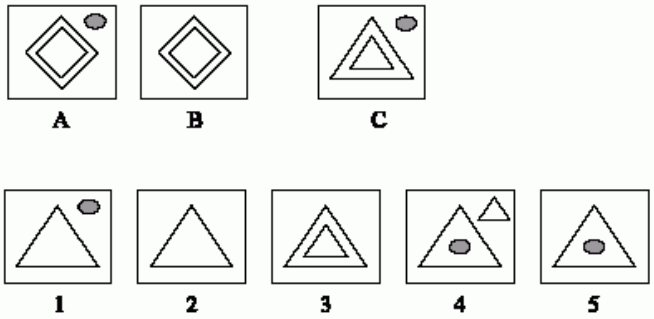


Рис. 6.6. Задача поиска геометрических аналогий

Решение таких задач предполагает описание изображения и преобразования (отношения между фигурами на изображениях), а также описание изменения отдельных фигур, составление правил и оценка изменений.

В качестве примера запишем три правила, показывающие, каким образом одно изображение (исходное) становится результирующим (рис. 6.7).

Правило 1 (исходное изображение): n внутри m, k внутри m, k внутри n.

Правило 2 (результирующее изображение): k выше m, k выше n, m внутри n.

Правило 3 (масштабирование, повороты): m, k изменили масштаб 1:1; n изменил масштаб 1:2.

Отметим важные моменты при таких преобразованиях. В исходном и результирующем изображениях допускаются отношения ВЫШЕ, ВНУТРИ, СЛЕВА. В результате преобразования изображение может стать МЕНЬШЕ, БОЛЬШЕ, испытать ПОВОРОТ или ВРАЩЕНИЕ, ОТРАЖЕНИЕ, УДАЛЕНИЕ, ДОБАВЛЕНИЕ.

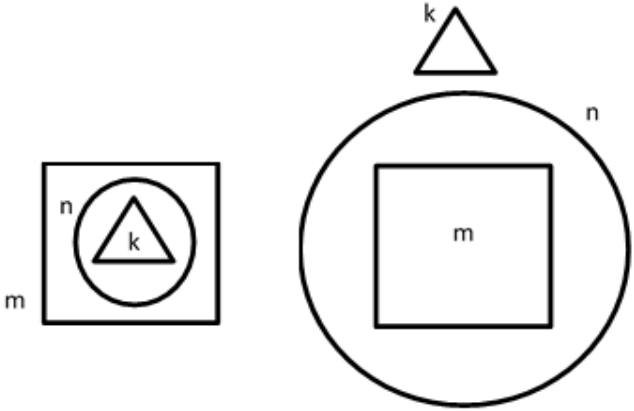


Рис. 6.7. Правила преобразования

Написание правил лучше всего начинать с проведения диагональных линий через центры фигур. Лишние отношения (СПРАВА ОТ и СЛЕВА ОТ, ВЫШЕ и НИЖЕ, ИЗНУТРИ и СНАРУЖИ,) использовать не рекомендуется.

Теперь задачи распознавания мы можем решать достаточно просто, записав для отношений правила 1, 2, 3

и проведя сопоставление. Например, так, как это сделано для следующей задачи: найти X такое, что $A \Rightarrow B$, как $C \Rightarrow X$ (см. рис. 6.8).

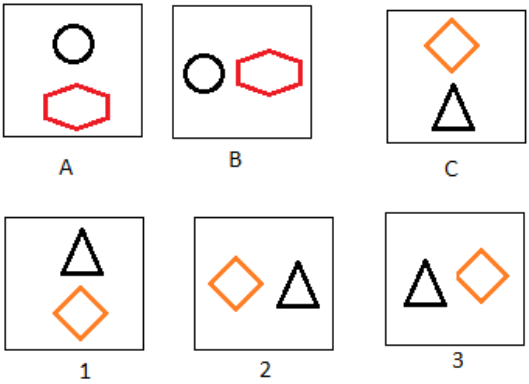


Рис. 6.8. Пример задачи распознавания по аналогии

Таблица 19. Пример преобразований

	Правило 1	Правило 2	Правило 3	Результат
$A \Rightarrow B$	K выше m	K слева m	k, m масштаб 1:1 поворот 00	
$C \Rightarrow 1$	X выше y	Y выше x	x, y масштаб 1:1 поворот 00	
$C \Rightarrow 2$	X выше y	Y слева x	x, y масштаб 1:1 поворот 00	
$C \Rightarrow 3$	X выше y	X слева y	x, y масштаб 1:1 поворот 00	Сопоставление успешно

Дополнительно следует отметить, что разные виды преобразований могут иметь различные веса, например, исчезновению фигуры целесообразно назначить больший

вес, чем преобразованию масштаба; а вращение фигуры может иметь меньший вес, чем отражение.

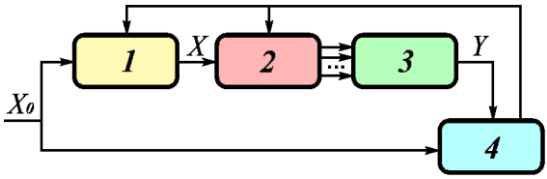
Методы распознавания по аналогии могут быть эффективнее, если используется обучение. Различают обучение с учителем, обучение по образцу (эталону) и другие виды обучения [2; 5; 7; 15; 17].

Методы восстановления изображений, основанные на принципах теории распознавания образов.

Основная идея: восстановление «по прецеденту» – построение алгоритма восстановления на основе эталонной пары «идеальное изображение – искаженное изображение» и его использование для других сигналов того же класса.

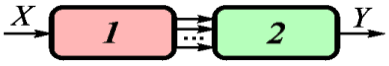
Общая схема преобразования данных.

а) обучение;



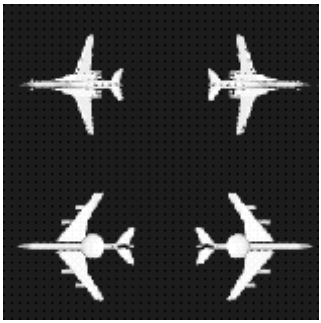
1. Модель системы искажений.
2. Формирование признаков.
3. Классификатор.
4. Оценка погрешности.

б) обработка;



1. Формирование признаков.
2. Классификатор.

Результаты обработки:



Исходное изображение

Искаженное изображение

Восстановленное фильтром Винера

Восстановленное с применением методики распознавания образов

Рис. 6.9. Пример восстановления изображения

6.3. Принципы классификации методов распознавания

Распознаванием образов называются задачи построения и применения формальных операций над числовыми или символьными отображениями объектов реального или идеального мира, результаты решения которых

отражают отношения эквивалентности между этими объектами. Отношения эквивалентности выражают принадлежность оцениваемых объектов к каким-либо классам, рассматриваемым как самостоятельные семантические единицы.

При построении алгоритмов распознавания классы эквивалентности могут задаваться исследователем, который пользуется собственными содержательными представлениями или использует внешнюю дополнительную информацию о сходстве и различии объектов в контексте решаемой задачи. Тогда говорят о «распознавании с учителем». В противном случае, то есть когда автоматизированная система решает задачу классификации без привлечения внешней обучающей информации, говорят об автоматической классификации или «распознавании без учителя». Большинство алгоритмов распознавания образов требует привлечения весьма значительных вычислительных мощностей, которые могут быть обеспечены только высокопроизводительной компьютерной техникой.

Различные авторы дают различную типологию методов распознавания образов. Одни авторы различают параметрические, непараметрические и эвристические методы, другие – выделяют группы методов, исходя из исторически сложившихся школ и направлений в данной области. Например, в работе [8], в которой дан прекрасный обзор методов распознавания, используется следующая типология методов распознавания образов:

– методы, основанные на принципе разделения;

– статистические методы;

– методы, построенные на основе «потенциальных функций»;

– методы вычисления оценок (голосования);

– методы, основанные на исчислении высказываний, в частности на аппарате алгебры логики.

Подобная типология методов распознавания с той или иной степенью детализации встречается во многих работах по распознаванию. В то же время известные типологии не учитывают одну очень существенную характеристику, которая отражает специфику способа представления знаний о предметной области с помощью какого-либо формального алгоритма распознавания образов. Д.А. Поспелов выделяет два основных способа представления знаний [8]:

1. Интенсиональное представление – в виде схемы связей между атрибутами (признаками).

2. Экстенсиональное представление – с помощью конкретных фактов (объекты, примеры).

Интенсиональное представление фиксируют закономерности и связи, которыми объясняется структура данных. Применительно к диагностическим задачам такая фиксация заключается в определении операций над атрибутами (признаками) объектов, приводящих к требуемому диагностическому результату. Интенсиональные представления реализуются посредством операций над значениями атрибутов и не предполагают произведения операций над конкретными информационными фактами (объектами).

В свою очередь, экстенсиональные представления знаний связаны с описанием и фиксацией конкретных объектов из предметной области и реализуются в операциях, элементами которых служат объекты как целостные системы.

Можно провести аналогию между интенсиональными и экстенсиональными представлениями знаний и механизмами, лежащими в основе деятельности левого и правого полушарий головного мозга человека. Если для правого полушария характерна целостная прототипная репрезентация окружающего мира, то левое полушарие оперирует закономерностями, отражающими связи атрибутов этого мира [8].

Описанные выше два фундаментальных способа представления знаний позволяют предложить следующую классификацию методов распознавания образов:

1) интенсиональные методы распознавания образов – методы, основанные на операциях с признаками;

2) экстенсиональные методы распознавания образов – методы, основанные на операциях с объектами.

Существование именно этих двух и только двух групп методов распознавания (оперирующих с признаками, и оперирующих с объектами) глубоко закономерно и отражает фундаментальные характеристики взаимодействия Реальности и Сознания. С этой точки зрения ни один из этих методов, взятый отдельно от другого, не позволяет сформировать адекватное отражение Реальности.

В приводимой ниже классификации основное внимание уделено формальным методам распознавания образов и поэтому опущено рассмотрение эвристического подхода к распознаванию, получившего полное и адекватное развитие в экспертных системах. По поводу этого подхода ограничимся лишь несколькими замечаниями.

Эвристический подход основывается на трудно формализуемых знаниях и интуиции исследователя. В этом подходе исследователь сам определяет, какую информа-

цию и каким образом нужно использовать для достижения требуемого эффекта распознавания.

6.3.1. Интенсиональные методы распознавания образов

Отличительной особенностью интенсиональных методов является то, что в качестве элементов операций при построении и применении алгоритмов распознавания образов они используют различные характеристики признаков и их связей. Такими элементами могут быть отдельные значения или интервалы значений признаков, средние величины и дисперсии, матрицы связи признаков и др., над которыми производятся действия, выражаемые в аналитической или конструктивной форме. При этом объекты в данных методах не рассматриваются как целостные информационные единицы, а выступают в роли индикаторов для оценки взаимодействия и поведения своих атрибутов. К интенсиональным методам относятся:

– методы, основанные на оценках плотностей распределения значений признаков;

– методы, основанные на предположениях о классе решающих функций;

– логические методы;

– лингвистические (структурные) методы и др.

Группа интенсиональных методов распознавания образов обширна, и ее деление на подклассы носит в определенной мере условный характер.

6.3.2. Экстенсиональные методы распознавания образов

В методах данной группы, в отличие от интенсионального направления, каждому изучаемому объекту в большей или меньшей мере придается самостоятельное диагностическое значение. По своей сути эти методы близки к клиническому подходу, который рассматривает людей не как проранжированную по тому или иному показателю цепочку объектов, а как целостные системы, каждая из которых индивидуальна и имеет особенную диагностическую ценность [8]. Такое бережное отношение к объектам исследования не позволяет исключать или утрачивать информацию о каждом отдельном объекте, что происходит при применении методов интенсионального направления, использующих объекты только для обнаружения и фиксации закономерностей поведения их атрибутов.

К экстенсиональным методам относятся:

– метод сравнения с прототипом;
– метод k-ближайших соседей;
– алгоритмы вычисления оценок (голосования);
– коллективы решающих правил и др.

Основными операциями в распознавании образов с помощью обсуждаемых методов являются операции определения сходства и различия объектов. Объекты в указанной группе методов играют роль диагностических прецедентов. При этом, в зависимости от условий конкретной задачи, роль отдельного прецедента может меняться в самых широких пределах от главной до весьма косвенного участия в процессе распознавания. В свою

очередь, условия задачи могут требовать для успешного решения участия различного количества диагностических прецедентов от одного в каждом распознаваемом классе до полного объема выборки, а также разных способов вычисления мер сходства и различия объектов. Этими требованиями объясняется дальнейшее разделение экстенсиональных методов на подклассы.

Контрольные вопросы

1. Какие понятия в теории распознавания образов является базовыми?
2. В чем состоит задача распознавание образов?
3. Что понимают под решающим правилом?
4. Что такое метрика?
5. Дать определение ситуации, состояния.
6. Что используется для распознавания сложных процессов и явлений?
7. Какие задачи относится к задачам распознавания?
8. Дать характеристику задач распознавания.
9. В чем заключается алгоритм распознавания, основанный на вычислении оценок?
10. К какому этапу относится оценка погрешности в общей схеме преобразования данных?
11. Привести пример распознавания образов по методу аналогий.
12. Перечислить методы распознавания образов.
13. Какие методы распознавания образов относятся к интенсиональным?
14. Какие методы распознавания образов относятся к экстенсиональным?
15. В чем заключается принцип обучение без учителя?

ЗАКЛЮЧЕНИЕ

В настоящее время уделяется большое внимание вопросам фундаментализации обучения информатике. В учебных планах вузов предусмотрено изучение теоретической информатики, как по направлениям подготовки бакалавров, так и по направлениям подготовки магистров.

Теоретическая информатика – это раздел науки, посвященный изучению теоретических основ компьютерных наук (*Computer Science*). Основными разделами теоретической информатики являются теория информации, теория кодирования, теория распознавания и др.

В учебном пособии представлены материалы для изучения курсов «Теоретические основы информатики» и «Теория информации, данные, знания». Для организации самостоятельной работы студентов, обучающихся по направлениям «Педагогическое образование» и «Информационные системы и технологии», в пособии представлены контрольные вопросы.

Пособие может быть использовано при обучении по программам бакалавриата и магистратуры. Для организации обучения с использованием модульно-рейтинговой системы контроля знаний студентов разработан практикум [14].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алферов, А.П. Основы криптографии / А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин. – Москва: Гемос АРВ, 2001. – 232 с. – ISBN 5-85438-025-0.
2. Вапник, В.Н. Теория распознавания образов / В.Н. Вапник, А.Я. Червоненкис. – Москва: Наука, 1979. – 132 с.
3. Введение в криптографию / под редакцией В.В. Ященко. – Москва: МЦ НМО, 2012. – 348 с. – ISBN 978-5-4439-0026-1.
4. Глушков, В.М. Основы безбумажной информатики / В.М. Глушков. – Москва: Наука, 1987. – 552 с.
5. Горелик, А.Л. Методы распознавания / А.Л. Горелик, В.А. Скрипкин. – Москва: Высшая школа, 1977. – 321 с.
6. Джонстон, Г. Учитесь программировать / Г. Джонстон. – Москва, 1989. – 368 с.
7. Завалишин, Н.В. Методы зрительного восприятия и алгоритмы анализа изображений / Н.В. Завалишин, И.Б. Мучник. – Москва: Наука, 1974. – 344 с.
8. Искусственный интеллект: справочник: в 3-х кн. / ред. Д.А. Поспелов. – Кн. 2: Модели и методы. – Москва: Радио и связь, 1990. – 304 с.
9. Кудрявцев, Б.Б. Введение в теорию автоматов / Б.Б. Кудрявцев, С.В. Алешин, А.С. Подколозин. – Москва: Наука, 1985. – 320 с.
10. Крайзмер, П. Кибернетика / П. Крайзмер. – Москва: Агропромиздат, 1985. – 225 с.
11. Молдовян, Н.А. Криптография. От примитивов к синтезу алгоритмов / Н.А. Молдовян, А.А. Молдовян, М.А. Еремеев. – Санкт-Петербург: BHV–Петербург, 2014. – 448 с. – ISBN 5-94157-524-6.
12. Нечаев, В.И. Элементы криптографии (Основы теории защиты информации): учебное пособие для университетов и педвузов / В.И. Нечаев; под. ред. В.А. Садовничего. – Москва: Высш. шк., 1999. –110 с. – ISBN: 5-06-003644-8.

13. Питерсон, У. Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон. – Москва: Мир, 1976. – 593 с.
14. Поднебесова, Г.Б. Теоретические основы информатики. Практикум / Г.Б. Поднебесова. – Челябинск: Изд-во ЧГПУ, 2015. – 110 с. – ISBN 978-5-906777-56-0.
15. Розенфельд, А. Распознавание и обработка изображений / А. Розенфельд. – Москва: Мир, 1972. – 256 с.
16. Саломаа А. Криптография с открытым ключом / А. Саломаа. – Москва: Мир, 1995. – 318 с. – ISBN 5-03-001991-X (рус.), ISBN 0-387-52831-8 (англ.).
17. Ту, Дж. Принципы распознавания образов / Дж. Ту, Р. Гонсалес. – Москва: Мир, 1978. – 405 с.
18. Уинстон, П. Искусственный интеллект / П. Уинстон; пер. с англ. – Москва, 1980. – 520 с.
19. Шеннон, К. Математическая теория связи / К. Шеннон // The Bell System Technical Journal. – Том 27, 1948. – С. 379–423. – URL: http://www.astronet.ru/db/msg/1188817 .
20. Шеннон, К. Теория связи в секретных системах / К. Шеннон // В кн. : Работа по теории информации и кибернетики. – Москва: Ил, 1963. С. 333–369. – URL: https://www.enlight.ru/crypto/articles/shannon/shann_i.htm .
21. Шоломов, Л.А. Основы теории дискретных логических и вычислительных устройств / Л.А. Шоломов. – Москва: Наука, 2011. – 432 с. – ISBN: 978-5-8114-1197-9 .

Учебное издание

Поднебесова Галина Борисовна

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Учебное пособие

ISBN 978-5-907611-09-2

Работа рекомендована РИС университета
Протокол № 25 от 2022 г.

Издательство ЮУрГГПУ
454080, г. Челябинск, пр. Ленина, 69

Редактор О.В. Куныгина
Технический редактор Т.Н. Никитенко

Подписано в печать 04.09.2021 г.
Объем 5,74 уч.-изд.л. (11,4 усл. печ. л.)
Формат 60×84/16. Тираж 100 экз.
Заказ №

Отпечатано с готового оригинал-макета в типографии ЮУрГГПУ
454080, г. Челябинск, пр. Ленина, 69