В. М. Глушков

Основы безбумажной информатики

В. М. ГЛУШКОВ

ОСНОВЫ БЕЗБУМАЖНОЙ ИНФОРМАТИКИ

ИЗДАНИЕ ВТОРОЕ, ИСПРАВЛЕННОЕ

МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1987

ББК 22.18
Г55
УДК 519.6

Глушков В. М. **Основы безбумажной информатики.** Изд. 2-е, испр.— М.: Наука. Гл. ред. физ.-мат. лит., 1987.— 552 с.

Книга посвящена проблемам автоматизированной обработки и хранению информации в безбумажном (машинном) представлении, т. е. с помощью ЭВМ. В ней представлено комплексное изложение состояния и перспектив безбумажной информатики и связанных с нею научно-технических проблем. Может быть использована для первоначального ознакомления.

1-е издание вышло в 1982 г.

Табл. 25. Ил. 74. Библиогр. 88 назв.

Виктор Михайлович Глушков

ОСНОВЫ БЕЗБУМАЖНОЙ ИНФОРМАТИКИ

Редактор *О. И. Сухова*

Художественный редактор *Г. М. Коровина*

Технический редактор *С. Я. Шкляр* Корректор *И. Я. Кришталь*

ИБ № 32347

Сдано в набор 28.05.86. Подписано к печати 24.11.86. Т-23909. Формат 60×90¹/₁₆. Бумага кн.-журн. Гарнитура обыкновенная новая. Печать высокая. Усл. печ. л. 34,5. Усл. кр.-отт. 34,5. Уч.-изд. л. 37,37. Тираж 37 500 экз. Заказ № 217. Цена 2 р. 60 к.

Ордена Трудового Красного Знамени издательство «Наука»
Главная редакция физико-математической литературы
117071 Москва В-71, Ленинский проспект, 15

4-я типография издательства «Наука» 630077 г. Новосибирск, 77, Станиславского, 25

1702070000—011
Г 053(02)-87 13-87

ПРЕДИСЛОВИЕ КО ВТОРОМУ ИЗДАНИЮ

XXVII съезд КПСС нацелил деятельность нашего народа на реализацию всех резервов, использование всех возможностей для ускорения социально-экономического прогресса советского общества.

В документах съезда выдвинуты масштабные и сложные задачи повышения уровня производительности труда, перестройки хозяйства и особенно совершенствования управления и планирования с учетом требований интенсивного развития экономики. Важную роль в решении этих задач играет применение электронных вычислительных машин, кибернетики и информатики. Перестройка производства, управления народным хозяйством, внедрение средств автоматизации в проектирование, планирование и управление, реализация качественно новых требований к сфере обслуживания, выполнение сложнейших научно-технических расчетов и разработок невозможны без новых интенсивных технологий, базирующихся на средствах машинной переработки информации.

Кибернетика и вычислительная техника активно влияют на все стороны жизни общества. В XII пятилетке парк кибернетических устройств, машинных средств обработки информации должен увеличиться во много раз. «Встраивание» автоматизированных средств в социальную среду с наименьшими потерями, обеспечение комфортного взаимодействия людей с машинами, создание и реализация новых высокопродуктивных технологических звеньев деятельности людей и машин — являются весьма трудными проблемами. Решение их — актуальная задача.

Последняя книга выдающегося ученого, академика Виктора Михайловича Глушкова, написанная им в 1981—1982 гг., посвящена вопросам автоматизированной обработки и хранения информации в безбумажном (машинном) представлении, а также проблемам, возникающим при создании безбумажных технологий в процессе обращения информации в обществе для решения практических задач автоматизации обработки результатов экспе-

риментов, управления технологическими процессами, организационного управления, проектно-конструкторских работ, решения творческих задач и т. п. Идеи, высказанные В. М. Глушковым в этой области, сформулированные им принципы построения новых кибернетических устройств, технических и математических средств преобразования информации, ряд его фундаментальных результатов могут явиться источником плодотворной деятельности научных исследователей и разработчиков новых информационных технологий. Книга является весьма полезной как комплексное изложение проблем, подходов и способов решения ряда задач безбумажной информатики.

В. С. Михалевич

ПРЕДИСЛОВИЕ К ПЕРВОМУ ИЗДАНИЮ

Книга, которую читатель держит в руках, является одной из последних работ выдающегося ученого и организатора науки, академика Виктора Михайловича Глушкова. Замечательный ученый нашего времени Виктор Михайлович Глушков работал на переднем крае науки — в области математики, кибернетики и вычислительной техники. Идеи, высказанные и разработанные Виктором Михайловичем Глушковым, оказали большое влияние практически на все области исследований в кибернетике и вычислительной технике не только в нашей стране, но и за рубежом. На этих идеях воспитывалось несколько поколений советских ученых, разработчиков средств кибернетической техники.

Руководствуясь принципами единства далеких и близких целей, единства теории и практики, Виктор Михайлович Глушков успешно работал над сложнейшими проблемами обработки информации и построением необходимых нашему обществу машин и систем для проведения научно-технических расчетов в проектировании, планировании и управлении, создании средств автоматизации различного уровня процессов в народном хозяйстве. Ему принадлежат многие пионерские идеи в этой области. Широко известны его монографии, посвященные проблемам создания кибернетических систем.

Виктор Михайлович Глушков руководил реализацией ряда проектов вычислительных машин, систем управления и обработки данных. Такие машины и системы, как Киев, МИР, Днепр, Проект, Львов, Дисплан и др., в которых нашли воплощение и практическое подтверждение оригинальные технические идеи и результаты фундаментальных исследований большого коллектива разработчиков, работающих под его руководством, составляют гордость отечественной науки.

Хотя проблемы безбумажной информатики, которым посвящена настоящая книга, являясь новым направлением в области обработки информации, еще не нашли полного своего решения, постановка их на повестку дня является чрезвычайно актуальной и своевременной.

Виктор Михайлович Глушков считал, что нет неразрешимых проблем, когда их решения требует практика деятельности общества. Для безбумажной информатики наступление этого этапа он предвидел и работал над тем, чтобы те, кому предстоит реализовать эти идеи, были заранее подготовлены. Именно этой цели служит настоящая книга, в которой изложен не только широкий спектр вопросов, относящихся к безбумажной информатике, но компактно и доходчиво изложены новейшие идеи и достижения в этой области научных исследований и практических разработок.

Герой Социалистического Труда, лауреат Ленинской премии, Государственных премий СССР и УССР, вице-президент Академии наук УССР, директор ордена Ленина Института кибернетики АН УССР*), академик Виктор Михайлович Глушков, безвременно ушедший от нас, отдал всю свою жизнь делу служения советской науке. Памятью о нем остаются его замечательные труды, среди которых и настоящая книга займет почетное место.

А. А. Дородницын

*) В настоящем Институт кибернетики АН УССР имени В. М. Глушкова. (*Примеч. ред.*)

ПРЕДИСЛОВИЕ АВТОРА К ПЕРВОМУ ИЗДАНИЮ

Одним из наиболее важных результатов научно-технической революции является бурная «компьютеризация» практически всех областей человеческой деятельности. Развитие сетей ЭВМ и систем терминального доступа к ним приводит к тому, что все большая часть информации, прежде всего научно-технической, экономической и социально-политической, перемещается в память ЭВМ.

Большинство выполненных к настоящему времени прогнозов сходятся на том, что к началу следующего столетия в технически развитых странах основная масса информации будет храниться в безбумажном виде — в памяти ЭВМ. Тем самым человек, который в начале XXI века не будет уметь пользоваться этой информацией, уподобится человеку начала XX века, не умевшему ни читать, ни писать. Поэтому уже в самом ближайшем будущем каждому образованному человеку надлежит быть знакомым с основами безбумажной информатики. Сегодня же эта задача стоит перед всеми выпускниками вузов естественно-научного, технического и социально-экономического профилей.

Между тем основу безбумажной информатики составляет целый ряд разделов науки и техники, настолько различающихся по своему содержанию и методам, что даже работающим в них узким специалистам порою нелегко находить общий язык. По этим разделам написано огромное количество специальной литературы. Однако она, как правило, труднодоступна для ознакомления с предметом специалистам соседних областей. Так, специалисту по математическим методам нелегко разобраться в современных монографиях по базам данных, специалисту в области прикладного программирования — в монографиях по сетям ЭВМ и т. д. Тем более трудно человеку, не являющемуся специалистом в области ЭВМ и обработки информации, составить по подобным публикациям общее представление об основах безбумажной информатики.

Главная цель настоящей книги как раз и состоит в том, чтобы подготовить читателя к чтению специальных монографий по раз-

личным разделам безбумажной информатики. В первую очередь книга адресована широкому кругу лиц со средним специальным, высшим и незаконченным высшим образованием, не являющихся специалистами по обработке информации. В связи с этим она не предполагает у читателя никаких специальных знаний в этой области, поскольку в ней излагаются все основные разделы безбумажной информатики, начиная с основных понятий.

Будучи ориентирована в первую очередь на пользователей, а не разработчиков ЭВМ и систем обработки информации, книга по возможности не содержит описания мелких технических деталей — упор делается именно на основные принципы, а не на технические подробности их реализации. Несмотря на это, автор надеется, что книга может оказаться полезной также и узким специалистам в различных разделах безбумажной информатики, которые захотят получить представление о предмете в целом. Следует при этом иметь в виду, что разделы, менее важные для пользователей (например, элементные базы), в книге лишь затрагиваются. Другие же разделы (например, базы данных) излагаются подробнее.

В. М. Глушков

ВВЕДЕНИЕ

Задача накопления, обработки и распространения (обмена) информации стояла перед человечеством на всех этапах его развития. В течение долгого времени основными инструментами для ее решения были мозг, язык и слух человека. Первое кардинальное изменение произошло в связи с приходом письменности, а затем изобретения книгопечатания. Эти два этапа создали принципиально новую технологию накопления и распространения информации, избавившую человечество от необходимости всецело полагаться на такой зыбкий и ненадежный инструмент, каким является человеческая память. Поскольку в эпоху книгопечатания основным носителем накапливаемой информации стала бумага, технологию накопления и распространения информации естественно называть *бумажной информатикой.*

Следует подчеркнуть, что революция в информатике, связанная со становлением письменности и книгопечатания, практически не затронула область переработки информации. Здесь основным и практически единственным рабочим инструментом продолжал оставаться человеческий мозг. Механизация отдельных операций по переработке информации (прежде всего вычислительных) с помощью тех или иных технических средств (логарифмическая линейка, арифмометр и т. п.) по существу ничего кардинально не меняла. Во-первых, при их использовании в бумажной форме представлялась не только исходная и заключительная информация, но и большинство промежуточных результатов, а во-вторых, каждый шаг по преобразованию информации контролировался и направлялся человеком. По существу все эти средства оставались лишь инструментами, а не подлинными машинами. О комплексной механизации и тем более автоматизации переработки информации на основе этих средств не могло быть и речи.

Положение в корне изменилось с появлением электронных вычислительных машин (ЭВМ). Подобно тому как изобретение механического двигателя открыло эру комплексной механизации и автоматизации физического труда, изобретение ЭВМ сделало

то же самое в отношении труда умственного. Правда, первые ЭВМ использовались в основном как большие автоматические арифмометры. Но уже тогда от обычных арифмометров их выгодно отличало отсутствие необходимости вмешательства человека в управление вычислительным процессом и запоминание промежуточных результатов.

Принципиально новый шаг был совершен, когда от применения ЭВМ для решения отдельных задач перешли к их использованию для комплексной автоматизации тех или иных законченных участков деятельности человека по переработке информации. Одними из первых примеров подобного системного применения ЭВМ в мировой практике были так называемые *административные системы обработки данных*: автоматизация банковских операций, бухгалтерского учета, материально-технического снабжения, резервирования и оформления билетов и т. п.

Решающее значение для эффективности систем подобного рода имеет то обстоятельство, что они опираются на автоматизированные информационные базы. Это означает, что в памяти ЭВМ постоянно сохраняется информация, нужная для решения тех задач, на которые рассчитана система. Например, для резервирования и продажи авиационных билетов системе постоянно требуется информация о расписании движения самолетов, о ценах на билеты и о проданных уже или заказанных билетах. Она и составляет содержание информационной базы соответствующей системы. При решении каждой очередной задачи (оформления заказа на билет) система нуждается во вводе только одной небольшой порции дополнительной информации (содержания заказа) — остальное берется из информационной базы.

Заметим, что каждая порция вновь вводимой информации, вообще говоря, изменяет информационную базу системы. Эта база находится, таким образом, в состоянии непрерывного обновления, отражая все изменения, происходящие в реальном объекте, с которым имеет дело система. Создание и поддержание подобного рода информационных баз, называемых обычно *базами данных*, представляют собой первый шаг на пути перехода к *безбумажной информатике*.

Заметим, что хранение информации в памяти ЭВМ придает ей принципиально новое качество *динамичности*, т. е. способности к быстрой перестройке и непосредственному оперативному ее использованию в решаемых на ЭВМ задачах (без кропотливой работы по ее вводу в ЭВМ). Устройства автоматической печати, которыми снабжаются практически все современные ЭВМ, позволяют в случае необходимости быстро представлять любую выборку из этой информации в привычной бумажной форме. Важно, что при этом отпадает необходимость в хранении и переписывании бесчисленных документов, являющихся непременными

атрибутами любой административной системы, работающей на основе старой «бумажной» (безмашинной) технологии.

По мере своего дальнейшего развития административные системы обработки данных перерастают в *автоматизированные системы управления* (АСУ) соответствующими объектами. Вначале эти объекты были в основном локальными (завод, аэропорт и т. п.). Крупным шагом вперед было объединение ЭВМ и ВЦ в *сети* с помощью телефонно-телеграфных каналов связи. В результате возникла возможность безбумажного обмена информацией между удаленными ЭВМ. Впрочем, еще до создания сетей ЭВМ подобный обмен уже осуществлялся путем пересылки информации на *машинных носителях*, прежде всего на магнитных лентах. Дополняя друг друга, эти два способа информационного обмена создали прочную основу для перестройки управления социально-экономическими процессами на основе безбумажной технологии в масштабах целой отрасли, крупного региона и даже целой страны.

Необходимость такой перестройки обуславливается тем, что в эпоху научно-технической революции резко усложнились задачи социально-экономического управления. Темпы роста сложности управления экономикой, особенно в эпоху НТР, значительно превосходят темпы роста любых других показателей экономического развития. В результате в развитии каждой страны неизбежно наступает момент, когда резервы традиционных приемов совершенствования управления экономикой — организация и социально-экономические механизмы — оказываются исчерпанными (*второй информационный барьер*).

Причина подобного явления заключается в том, что все традиционные организационные и социально-экономические механизмы реализуются непосредственно через людей, точнее — через их мыслительный аппарат — мозг. Пропускная же способность мозга как преобразователя информации хотя и велика, но тем не менее ограничена. То же самое имеет место и для совокупности всех людей в любой данной социально-экономической системе. Момент, когда сложность задач управления системой превосходит этот порог, и есть второй информационный барьер *).

С этого момента дальнейшее увеличение мощности управленческого аппарата возможно лишь на основе непрерывного повышения производительности труда всех занятых в управлении людей. Такого повышения нельзя достичь в рамках традиционной (бумажной) технологии за счет оснащения людей теми или иными инструментами, действующими «россыпью», т. е. тогда, когда

*) *Первым информационным барьером* называется порог сложности управления экономической системой, превосходящей возможности одного человека.

все информационные потоки замыкаются в конечном счете через людей. Резервы роста производительности труда в такой технологии быстро исчерпываются за счет наличия в ней узких мест, определяемых пропускной способностью человеческого звена. Необходима *комплексная автоматизация* управленческого труда, при которой все большая и большая часть информационных потоков замыкается вне человека.

В этом и состоит сущность *безбумажной технологии*. Следует особо подчеркнуть, что она никоим образом не устраняет человека из системы управления, а лишь передвигает его усилия от рутинной работы в более творческие области. В конечном счете обязанности человека в системе управления сведутся к постановке задач, выбору окончательных вариантов управленческих решений (приданию им юридической силы) и к неформализуемой работе с людьми. Точно так же не следует впадать в вульгаризацию, считая, что новая технология устраняет абсолютно все бумажные документы. Различного рода обобщенные показатели, наиболее важные решения, равным образом как и различного рода неформальные заявления, письма и другая информация, ориентированная на человека, могут оставаться в бумажной форме даже на высших стадиях развития безбумажной технологии.

К высказанным здесь мыслям, из которых, в частности, следует историческая неизбежность безбумажной технологии организационного управления, автор пришел в основном еще в начале 60-х годов. К середине 1964 г. под руководством автора был разработан первый эскизный проект Единой Государственной сети вычислительных центров (ЕГСВЦ), предназначенной для перестройки на основе безбумажной технологии организационно-экономического управления на всех уровнях (от отдельных предприятий и учреждений до Госплана СССР).

В 1965—1970 гг. в нашей стране был выполнен определенный объем работ по созданию автоматизированных систем управления предприятиями (АСУП) и отраслевых автоматизированных систем управления (ОАСУ) в отдельных министерствах.

Мы остановились столь подробно на задачах автоматизации организационного управления потому, что именно здесь в первую очередь концентрируются основные проблемы становления безбумажной информатики. Однако этими задачами рамки безбумажной информатики отнюдь не ограничиваются.

Интенсивно развиваются системные применения ЭВМ для автоматизации сложных технологических процессов — автоматизированные системы управления технологическими процессами (АСУТП), для испытаний сложных объектов, экспериментальных исследований, проектно-конструкторских работ и др. Успехи числового программного управления позволили поставить и ре-

шить задачу создания полностью автоматизированных, гибко перестраиваемых технологических линий, участков и даже целых цехов.

Наряду со станками с числовым программным управлением на этих линиях и участках успешно работают роботы первого поколения. Происходит слияние АСУТП и АСУП в единые (интегрированные) системы управления производством. При этом все новые и новые информационные потоки замыкаются через ЭВМ, минуя людей.

Успехи автоматизации экспериментальных исследований и испытаний приводят к тому, что данные непосредственно от измерительных приборов через соответствующие системы сопряжения попадают в память ЭВМ и накапливаются на машинных носителях. Это позволяет не только резко убыстрить последующую обработку полученных данных, но и организовать в случае необходимости длительное их хранение для возможного использования в будущем (например, при появлении новых улучшенных методов их обработки или изменении характера обработки при постановке новых целей перед исследователями). Тем самым создаются центры накопления экспериментальной научной информации в безбумажном виде. Наряду с ними возникают ВЦ, в которых накапливаются рефераты книг, научных статей и докладов. Эти ВЦ объединяются в сети. Сидя за терминалом подобной сети, научный работник может получать любую интересующую его информацию на экране терминального дисплея (а в случае необходимости — и бумажную копию этой информации). По мере удешевления памяти ЭВМ в подобной безбумажной форме будут представляться полные тексты книг, статей и отчетов.

Аналогичные процессы происходят с проектно-конструкторской информацией. Все в большей мере безбумажная информатика проникает в медицину, культуру, спорт, контроль окружающей среды и в другие области человеческой деятельности.

Глава I

ОСНОВНЫЕ ПОНЯТИЯ ОБ ИНФОРМАЦИИ И ЕЕ ПРЕОБРАЗОВАНИЯХ

1.1. Непрерывная и дискретная информация

Информация о различных природных явлениях и технологических процессах воспринимается человеком (с помощью его собственных органов чувств и различной измерительной аппаратуры) в виде тех или иных полей. Математически такие поля представляются с помощью функций $y = f(x, t)$, где t — время, x — точка, в которой измеряется поле, y — величина поля в этой точке. При измерениях поля в фиксированной точке $x = a$ функция $f(x, t)$ вырождается в функцию времени $y(t) = f(a, t)$, которую можно изобразить в виде графика.

В большинстве случаев все скалярные величины, входящие в соотношение $y = f(x, t)$ (т. е. t, y и координаты точки x), могут принимать непрерывный ряд значений, измеряемых вещественными числами. Под *непрерывностью* здесь понимается то, что рассматриваемые величины могут изменяться сколь угодно мелкими шагами. Ввиду этого представляемую таким способом информацию принято называть *непрерывной информацией*. Иногда для этой цели используется также термин *аналоговая информация*.

Если применительно к той же самой информации о поле $y = f(x, t)$ установить минимальные шаги изменения всех характеризующих ее скалярных величин, то получим так называемое *дискретное представление* информации (*дискретная информация*). Поскольку точность измерений (равно как и человеческого восприятия) всегда ограничена, то фактически, даже имея дело с непрерывной информацией, человек воспринимает ее в дискретном виде. Но любая непрерывная информация может быть аппроксимирована дискретной информацией с любой степенью точности, поэтому можно говорить об *универсальности* дискретной формы представления информации.

Результаты измерения любых скалярных величин представляются в конце концов в числовом виде, а поскольку при заданной точности измерений эти числа представимы в виде конечных наборов цифр (с запятой или без нее), то дискретную форму представления информации часто отождествляют с *цифровой информацией*.

1.2. Абстрактные алфавиты

Цифровая информация в действительности представляет собой частный случай так называемого *алфавитного способа* представления дискретной информации. Его основой является произвольный фиксированный конечный набор символов любой природы, называемый *абстрактным алфавитом* или, для краткости, просто *алфавитом*.

Совокупность десятичных цифр вместе с запятой для отделения дробной части числа можно рассматривать как частный случай абстрактного алфавита с 11 символами — *буквами* этого алфавита. Другой пример — алфавит естественного человеческого языка, например, русского. Язык математических и других научных текстов может включать наряду с обычными буквами данного языка буквы других языков (например, греческого), а также различные специальные символы (например, символы арифметических операций +, — и др.).

1.2.1. Кодирование. При обработке информации часто возникает необходимость представлять средствами одного алфавита буквы других алфавитов. Такое представление носит в информатике специальное наименование *кодирования*. Проблема решается просто, если требуется закодировать буквы алфавита X с меньшим числом букв, чем у кодирующего алфавита Y. Если, например, X — алфавит десятичных цифр, а Y — обычный русский алфавит, то для кодирования X в Y достаточно положить 0 = а, 1 = б, 2 = в, 3 = г, ..., 9 = к. Разумеется, возможны и другие способы кодирования, в том числе и такие, в которых буквы алфавита X кодируются несколькими буквами алфавита Y.

Одним из наиболее естественных способов такого кодирования является просто замена десятичных цифр их русскими названиями: нуль, один, два и т. д.

При кодировании алфавитов с большим числом букв в алфавитах с меньшим числом букв использование для кодирования последовательностей букв является обязательным условием для возможности различения кодов различных букв, что является непременным условием правильного кодирования. Так, буквы русского алфавита можно закодировать парами десятичных цифр, например, а = 01, б = 02, ..., к = 10, л = 11, ...

1.2.2. Двоичный алфавит. Простейшим абстрактным алфавитом, достаточным для кодирования любого другого алфавита, является алфавит, состоящий из двух букв. Такой алфавит носит наименование *двоичного*, а две его буквы чаще всего принято отождествлять с цифрами 0 и 1. Величина, способная принимать лишь два различных значения, представляет собой своеобразный *информационный атом*, получивший специальное наименование *бит* (минимальная единица информации).

1.2.3. Байтовый алфавит. Ввиду своей простоты двоичный алфавит наиболее широко распространен в различного рода технических информационных устройствах и в первую очередь в электронных вычислительных машинах. Для кодирования же алфавитов, которыми привык пользоваться человек, употребляются последовательности двоичных цифр. Легко видеть, что последовательностями из n двоичных цифр 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 получила в связи с этим специальное наименование *байт* *). Составляемый же различными подобными последовательностями алфавит из 256 букв естественно называть *байтовым алфавитом*. В практике использования ЭВМ в международном масштабе укоренился единый стандарт байтового кодирования строчных и прописных букв латинского алфавита, знаков препинания, десятичных цифр с десятичной запятой (десятичной точкой), а также ряда наиболее употребительных математических символов (знаки арифметических и логических операций, знаки равенства и неравенства и др.). Специальный байтовый код закрепляется также за знаком *пробела*.

Мощность (число букв) байтового алфавита оказывается достаточной 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. Следует, однако, заметить, что число символов в 200 и более предъявляет слишком большие требования к вводно-выводной аппаратуре. Нетрудно представить себе, например, насколько громоздкой стала бы обычная пишущая машинка, способная печатать 256 различных символов.

Реальные алфавиты, используемые для ЭВМ, ограничиваются, как правило, меньшим числом символов. Здесь пока нет строго установившегося стандарта. Заметим, что достаточно часто на

*) Ранее в информатике употреблялись 7-, 6- и даже 5-битовые байты, но в настоящее время 8-битовый байт можно считать установившимся международным стандартом.

практике в ЭВМ (точнее, в их устройствах ввода-вывода) используют 96-буквенный алфавит, который мы будем называть *базисным*. Если при этом для некоторых символов не хватает кодов, то берутся слова в наличном алфавите: например, вместо греческой буквы α пишется ее полное наименование по-латыни, т. е. alpha. В ряде случаев в целях экономии отказываются от употребления строчных букв, используя лишь заглавные, и т. д.

1.3. Слова и абстрактные языки

Любая конечная последовательность букв в (абстрактном) алфавите X называется *словом* в этом алфавите. Заметим, что при этом не требуется никакой осмысленности слова, даже если речь идет о словах в алфавитах естественных человеческих языков. Более того, если, скажем, алфавит X состоит из букв русского алфавита, а также из знаков препинания и символа пробела, то словом в таком алфавите может считаться любая фраза и даже целая книга.

Для того чтобы из всего получаемого таким образом множества слов выделить п р а в и л ь н ы е в каком-то смысле слова, над исходным алфавитом X строится так называемый *формальный язык*. Кроме алфавита X, формальный язык задается своей особой (формальной) *грамматикой*. Она представляет собой не что иное, как конечную совокупность формальных правил, с помощью которых порождаются в с е слова данного языка (т. е. правильные слова) и т о л ь к о такие слова.

В современной информатике разработано несколько способов задания грамматик, порождающих формальные языки. Одним из простейших способов (достаточным для большинства практических применений) является использование так называемых *нормальных форм Бэкуса*. Этот способ проще всего разъяснить на примере.

Пусть, например, над алфавитом X, состоящим из двоичных цифр ⟨0, 1⟩ и (двоичной) запятой ⟨ , ⟩, требуется построить язык, состоящий из всех рациональных двоичных (неотрицательных) чисел (как целых, так и дробных). В методе Бэкуса для такого определения используются так называемые *метапонятия*. В данном случае ими будут понятия ⟨целое число⟩ и ⟨двоичное число⟩.

Употребляя специальное сокращение ::= для выражения «левая часть есть любое из значений форм, представленных в правой части», запишем следующие выражения:

⟨целое число⟩ ::= ⟨0⟩|⟨1⟩|⟨0⟩⟨целое число⟩|⟨1⟩⟨целое число⟩,
⟨двоичное число⟩ ::= ⟨целое число⟩⟨,⟩⟨целое число⟩*).

*) Вертикальная черта представляет собой особый знак, соответствующий русскому слову «или».

Заметим, что в силу этого определения правильно записанными двоичными числами будут считаться, например, числа 01, 0010, 00 и вообще числа с любым количеством незначащих нулей (как слева, так и справа). Чтобы избежать этого, определение должно быть соответствующим образом усложнено.

Неискушенному читателю приведенная формализация может показаться излишней и надуманной. Действительно, для определения хорошо известных и достаточно простых понятий без нее можно вполне обойтись. Когда же строятся более необычные и сложные понятия, подобные формальные определения становятся не только наиболее точными и короткими, но также (после небольшой практики) и наиболее удобными.

1.4. Данные. Типы элементарных данных

Информация, с которой имеют дело различного рода автоматизированные информационные системы, обычно называется *данными*, а сами такие системы — *автоматизированными системами обработки данных* (АСОД). Различают *исходные* (*входные*), *промежуточные* и *выходные данные*.

Данные разбиваются на отдельные составляющие, называемые *элементарными данными* или *элементами данных*. Употребляются элементы данных различных типов. *Тип данных* (элементарных) зависит от значений, которые эти данные могут принимать.

В современной безбумажной информатике среди различных типов элементарных данных наиболее употребительными являются *целые* и *вещественные* числа, *слова* (в некотором подалфавите байтового алфавита) и так называемые *булевы величины*. Первые два типа величин нуждаются в пояснении только в связи с конкретными особенностями их представления в современных ЭВМ.

Прежде всего различают двоичное и двоично-десятичное представления чисел. В *двоичном* представлении используется *двоичная система счисления* с фиксированным числом двоичных разрядов (чаще всего 32 или, для малых ЭВМ, 16 разрядов, включая разряд для представления знака числа). Если нулем обозначать плюс, а единицей — минус, то 00001010 означает целое число $+(2^3+2^1)=+10$, а 10001100 — число $-(2^3+2^2)=-12$ (для простоты взято 8-разрядное представление). Заметим, что знак числа в машинном представлении часто оказывается удобным ставить не в начале, а в конце числа.

В случае *вещественных* чисел (а фактически, с учетом ограниченной разрядности, дробных двоичных чисел) употребляются две формы представления: с *фиксированной* и с *плавающей запятой*. В первом случае просто заранее условливаются о месте

нахождения запятой, не указывая ее фактически в коде числа. Например, если условиться, что запятая стоит между 3-м и 4-м разрядами справа, то код 00001010 будет означать число $00001{,}010_2 = (1 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3}) = 1{,}25$. Во втором случае код числа разбивается на два кода в соответствии с представлением числа в виде $x = a \cdot 2^b$. При этом число a (со знаком) называется *мантиссой*, а число b (со знаком) — *характеристикой* числа x. О положении кода характеристики и мантиссы (вместе с их знаками) в общем коде числа также устанавливаются заранее.

Для экономии числа разрядов в характеристике b ее часто представляют в виде $b = 2^k b_1$, где k — фиксированная константа (обычно $k = 2$). Вводя еще одну константу m и полагая $b = 2^k b_2 - m$, можно избежать также использования в коде характеристики знака (при малых $b_2 > 0$ число b отрицательно, а при больших — положительно).

В *двоично-десятичном* представлении обычные десятичные цифры (а также запятая и знак) кодируются двоичными цифрами. При этом для экономии места часто используется так называемый *упакованный код*, когда с помощью одного байта кодируется не одна, а две десятичные цифры. Подобное представление позволяет в принципе кодировать числа любой значности. На практике обычно все же ограничивают эту значность, хотя и столь большими пределами, что можно считать их неограниченными.

Тип данных «произвольное слово во входном алфавите» не нуждается в специальных пояснениях. Единственное условие — необходимость различать границы отдельных слов. Это достигается использованием специальных ограничителей и указателей длины слов.

Тип *булева переменная* присваивается элементарным данным, способным принимать лишь два значения: «истина» (и) и «ложь» (л). Для представления булевых величин обычно используется двоичный алфавит с условием и = 1, л = 0.

1.4.1. Переменные и постоянные величины (данные). Для идентификации различных данных употребляются специальные наименования элементарных данных, называемые *идентификаторами*. Они являются аналогами переменных величин, фигурирующих в курсе математики. Область значений, которые может принимать эта переменная, определяется типом элемента данных, именем которого она является.

В качестве идентификаторов элементарных данных в современной информатике принято использовать любые конечные последовательности латинских букв и десятичных цифр, начинающиеся буквой. Такое условие дает неограниченный запас идентификаторов, что немаловажно, поскольку в современной

информатике приходится иметь дело со многими тысячами элементарных данных.

В отличие от переменных величин (данных), для которых требуется четко различать *имя* и *значение величины*, для постоянных величин в принципе значение величины может служить и ее именем. Однако во многих случаях полезно именовать те или иные постоянные величины соответствующими идентификаторами (например, число $e \approx 2{,}718$ и др.). Разумеется, при этом надо помнить, что величина, обозначенная соответствующим идентификатором, представляет собой *константу* и может, следовательно, принимать лишь одно-единственное значение.

1.5. Модели и алгебры данных

Как известно, *моделью* в математике принято называть любое множество объектов, на которых определены те или иные предикаты. Под *предикатом* здесь и далее понимается функция $y = f(x_1, \ldots, x_n)$, аргументы $(x_1, \ldots, x_n)$ которой принадлежат данному множеству M, а значение (y) может являться либо истиной, либо ложью. Иными словами, предикат представляет собой переменное (зависящее от параметров $x_1, \ldots, x_n$) *высказывание*. Оно описывает некоторое свойство, которым может обладать или не обладать набор элементов $(x_1, \ldots, x_n)$ множества M.

Число n элементов этого набора может быть любым. При $n = 2$ возникает особо распространенный тип предиката, который носит наименование *бинарного отношения* или просто *отношения*. Наиболее употребительными видами отношений являются *отношения равенства* $(=)$ и *неравенства* $(\neq)$. Эти отношения естественно вводятся для элементарных данных любого данного типа. Тем самым соответствующий тип данных превращается в модель.

Применительно к числам (целым или вещественным) естественным образом вводятся также *отношения порядка* $>, <, \geqslant, \leqslant$. Тем самым для соответствующих типов данных определяются более богатые модели.

Любое множество M, как известно, превращается в *алгебру*, если на нем задано некоторое конечное множество операций. Под *операцией* понимается функция $y = f(x_1, \ldots, x_n)$, аргументы и значение которой являются элементами множества M. При $n = 1$ операция называется *унарной*, а при $n = 2$ — *бинарной*. Наиболее распространенными являются бинарные операции.

Для целых чисел естественным образом вводятся бинарные операции сложения, вычитания и умножения, а также унарная операция перемены знака числа. В случае вещественных чисел к ним добавляется бинарная операция деления и (если необхо-

димо) унарная операция взятия обратной величины. Разумеется, при необходимости могут быть введены и другие операции.

1.5.1. Булева алгебра. Особое место в машинной информатике занимает *булева алгебра*, вводимая на множестве величин типа булевых. Ее основу составляют две бинарные операции: *конъюнкция* («и»), *дизъюнкция* («или») и одна унарная операция: *отрицание* («не»). Конъюнкция обозначается символом $\wedge$ и задается правилами $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$, $1 \wedge 1 = 1$. Для дизъюнкции используются символ $\vee$ и правила $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 0 = 1$, $1 \vee 1 = 1$. Наконец, отрицание $\neg$ меняет значение булевой величины на противоположное: $\neg 0 = 1$, $\neg 1 = 0$. Последовательность выполнения операций производится в порядке убывания приоритетов от $\neg$ к $\wedge$ и далее к $\vee$ (если специальной расстановкой скобок не оговорено противное). Например, порядок действий в формуле $\neg a \wedge b \vee c \wedge \neg d$ соответствует прямо указанному скобками порядку:

$$((\neg a) \wedge b) \vee (c \wedge (\neg d)).$$

В принципе могут быть введены и другие операции, однако оказывается, что любую такую операцию можно выразить в виде формулы, использующей только конъюнкции, дизъюнкции и отрицания. Таким образом, введенный набор операций является для булевой алгебры *универсальным*.

1.6. Комбинационные схемы

Поскольку любая алфавитная (буквенно-цифровая) информация может быть закодирована в двоичной форме, то подобным образом могут быть закодированы условия и решения задач из любой области знаний. Если число таких задач конечно (хотя, может быть, и очень велико), то существуют максимальная длина m кода условий этих задач и максимальная длина n кода их решений. В таком случае решения всех данных задач (в двоичном коде) могут быть получены из их условий с помощью некоторой системы булевых функций $y_i = f_i(x_1, x_2, \ldots, x_m)$ $(i = 1, \ldots, n)$. В свою очередь все эти функции могут быть выражены через элементарные булевы операции копъюнкции, дизъюнкции и отрицания.

Существуют различные способы представления булевых величин (двоичных цифр) в виде тех или иных физических (обычно электрических) сигналов (высокое и низкое напряжение, импульсы тока разной полярности и т. п.).

Выбрав форму представления (двоичных) сигналов, можно построить элементарные устройства, называемые обычно *логическими вентилями* (или *логическими элементами*), которые реализуют элементарные булевы операции. Иными словами, выходные

сигналы этих устройств представляют собой элементарные булевы функции (результат выполнения элементарных булевых операций) от входных сигналов, как это показано на рис. 1.1.

Имея запас таких элементов, можно строить более сложные схемы, подсоединяя выходы одних элементов к входам других. Если при таких соединениях избегать возникновения замкнутых контуров (например, подсоединения выхода элемента на один из его собственных входов), то возникает класс схем, называемых обычно *комбинационными схемами*. Такие схемы находятся в однозначном соответствии с формулами булевой алгебры, так что с их помощью может быть выражена любая система булевых функций. Например, схема, изображенная на рис. 1.2, реализует систему булевых функций

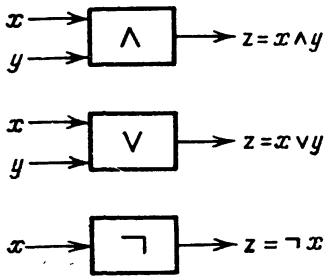

Рис. 1.1

$$u = x \wedge y \vee \neg z \text{ и } v = \neg(x \vee y \vee z).$$

На практике построение комбинационных схем усложняется, поскольку сигналы при прохождении через вентили ослабляются, искажают свою первоначальную форму, запаздывают. Поэтому необходимо наряду с логическими элементами включать в схему различного рода *согласующие элементы* (усилители, формирователи сигналов и др.). Задача этих элементов — сделать схему работоспособной и надежной.

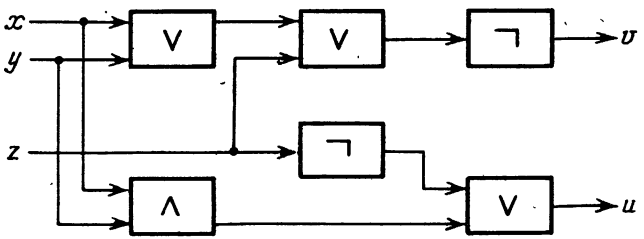

Рис. 1.2

Из сказанного ясно, что можно построить комбинационную схему для решения любого конечного множества задач, решения которых однозначно определяются их условиями (подаваемыми на вход схемы). В частности, если ограничиться какой-либо фиксированной точностью представления вещественных чисел (разрядностью), то можно в принципе построить комбинацион-

ную схему, вычисляющую любую заданную вещественную функцию $y = f(x_1, \ldots, x_n)$ (в двоичных кодах), например, $x_1^{x_2}$, $\sin x_1$ и др.

На практике, однако, оказывается, что уже схема *умножителя* (вычисляющая функцию $y = x_1 \cdot x_2$) при разрядности (двоичной) 32 и более оказывается столь сложной, что умножение в современных ЭВМ предпочитают реализовать другим, так называемым алгоритмическим способом, о котором речь пойдет ниже.

В то же время многие, более простые функции, например функции сложения двух чисел, реализуются комбинационными схемами приемлемой сложности. Соответствующая схема носит наименование *параллельного сумматора.*

Следует заметить, что успехи микроэлектроники делают возможным построение все более сложных схем. Если еще в 60-е годы каждый логический элемент собирался из нескольких физических элементов (транзисторов, диодов, сопротивлений и др.), то уже к началу 80-х годов промышленностью выпускаются так называемые *интегральные схемы*, содержащие многие сотни и даже тысячи логических вентилей. При этом важно подчеркнуть, что не только сами логические элементы, но и соединения между ними (т. е. вся схема в целом) изготовляются одновременно в едином технологическом процессе на тонких пластинках химически чистого кремния и других веществ размерами в доли квадратного сантиметра. Благодаря этому резко уменьшилась стоимость изготовления схем и повысилась их надежность.

1.6.1. Быстродействие комбинационных схем. В теоретических рассмотрениях обычно считается, что сигналы на выходах комбинационной схемы появляются в тот же момент, когда на вход схемы поступают инициирующие их входные сигналы. Иными словами, *быстродействие* схемы предполагается бесконечным.

На практике, однако, каждый элемент схемы осуществляет некоторую *задержку* сигнала (хотя обычно и весьма малую). Иными словами, сигнал на выходе элемента (логического вентиля) появляется лишь через некоторое время $\tau > 0$ после прихода соответствующих входных сигналов.

При уменьшении размера элемента задержка (при прочих равных условиях) уменьшается, стремясь к некоторому пределу, зависящему от физической природы этого элемента. Для полупроводниковых элементов к началу 80-х годов в экспериментальных образцах достигнуты задержки порядка десятых долей *наносекунды* (1 нс $= 10^{-9}$ с), что уже практически подходит к естественному физическому пределу для полупроводниковых схем. Степень микроминиатюризации позволяет при этом разместить на пластинке кремния площадью менее квадратного сантиметра — так называемом *чипе* — несколько сотен тысяч логических элементов.

Серийно производимые схемы, разумеется, уступают экспериментальным как в быстродействии, так и в плотностях размещения элементов. Но уже и в этом случае возникает немало сложных технических проблем, из которых на одном из первых мест стоит проблема отвода выделяющегося тепла (особо критичная для быстродействующих полупроводниковых элементов).

Переход к *пикосекундным* задержкам (1 пс $= 10^{-12}$ с) требует новых физических принципов построения элементов. Заметим, что за одну пикосекунду свет проходит всего 0,3 миллиметра. Поэтому перспективы выхода на подобное быстродействие тесно связаны с проблемой дальнейшей микроминиатюризации схем.

Поскольку при последовательном соединении элементов задержки в них суммируются, в практике построения комбинационных схем предпочитают не употреблять длинных цепей элементов.

1.7. Последовательностные схемы (конечные автоматы)

Обладая возможностью реализовать любые фиксированные зависимости между входными и выходными сигналами, комбинационные схемы неспособны обучаться, адаптироваться к изменяющимся условиям. На первый взгляд кажется, что такая адаптация обязательно требует *структурных изменений* в схеме, т. е. изменения связей между ее элементами, а возможно, и состава этих элементов. Подобные изменения нетрудно реализовать путем механических переключений. Однако такой путь практически неприемлем из-за резкого ухудшения практически всех параметров схемы (быстродействия, габаритов, надежности и др.).

Существует гораздо более эффективный путь решения указанной проблемы, основанный на введении в схему в дополнение к уже перечисленным логическим элементам так называемых *элементов памяти*. Помимо своих входных и выходных сигналов, элемент памяти характеризуется еще третьим информационным параметром — так называемым *состоянием* этого элемента. Состояние элемента памяти может меняться (но не обязательно) лишь в заданные дискретные моменты времени t_1, t_2, ... под влиянием сигналов, появляющихся на его входах в эти моменты. Наиболее употребительна так называемая *синхронная* организация работы элементов памяти, при которой моменты их возможных *переключений* (изменений состояния) следуют друг за другом через один и тот же фиксированный промежуток времени $\Delta t = \text{const}$, называемый *тактом*. Эти моменты определяются обычно с помощью импульсов, вырабатываемых специальным *тактирующим синхрогенератором*. Количество тактовых импуль-

сов, выдаваемых им в течение одной секунды, называется *тактовой частотой.*

Состояние элемента памяти $a(t)$ в любой заданный момент переключения t является однозначной функцией $a(t) = \zeta(a(t - \Delta t), x(t))$ его состояния $a(t - \Delta t)$ в предшествующий момент времени и входного сигнала $x(t)$ в настоящий момент времени. Эта функция называется *функцией переходов* (*переключений*) данного элемента. Разумеется, в реальных схемах переключение состояния происходит не мгновенно, а с некоторой задержкой, характеризующей *быстродействие* выбранных элементов памяти. Величина такта выбирается в зависимости от этой задержки и, как правило, существенно меньше ее. Единицей быстродействия служит герц (Гц), соответствующий тактовой частоте величиной в одно переключение в секунду. Частота в один килогерц (кГц) соответствует тысяче переключений в секунду (величина такта 1 мс $= 10^{-3}$ с), частота в один мегагерц (МГц) означает миллион переключений в секунду (величина такта 1 мкс $= 10^{-6}$ с) и, наконец, один гигагерц (ГГц) соответствует миллиарду переключений в секунду (величина такта 1 нс $= 10^{-9}$ с).

Выходной сигнал $y(t)$ элемента памяти, как правило, совпадает с его состоянием $a(t)$. Следовательно, в моменты переключений его можно задать функцией переходов $\zeta(a(t - \Delta t), x(t))$. В более общем случае выходной сигнал может задаваться произвольной функцией $y(t) = \psi(a(t - \Delta t), x(t))$, называемой *функцией выходов.*

В современной электронике употребляются в основном *двоичные элементы памяти*, состояние которых представляет собой булеву величину. Иными словами, элемент памяти способен запомнить всего лишь один бит информации. При необходимости запоминания большего количества информации используется *составная память* (запоминающее устройство), состоящая из некоторого множества элементов. В реальных условиях это множество, разумеется, всегда конечно, хотя в теоретических исследованиях бывает удобно рассматривать и бесконечную память (по крайней мере потенциально).

В простейшем случае множество элементов памяти организуется в так называемый *регистр*, т. е. в (конечную) линейно упорядоченную последовательность элементов, называемых *разрядами* (*ячейками*) *регистра.* Разряды нумеруются последовательными натуральными числами 1, 2, ..., n. Число n этих разрядов называется *длиной регистра.*

Состояния a_i отдельных разрядов составляют (булев) вектор a, называемый *состоянием регистра.* Входные и выходные сигналы отдельных разрядов рассматриваемого регистра (также предполагаемые булевыми) составляют соответственно входной x и выходной y (векторные) сигналы данного регистра.

Заметим еще раз, что в подавляющем большинстве случаев $y = a$.

Обычная *последовательностная схема*, называемая также *конечным автоматом*, составляется из регистра памяти R и двух комбинационных схем S_1 и S_2, соединенных между собой так, как это показано на рис. 1.3. Как видно из этого рисунка, выходы (выходные каналы) x схемы S_1 подключены к входам регистра R, а выходы v схемы S_2 совпадают с выходами всей последовательностной схемы. Что же касается входов схем S_1 и S_2, то они являются комбинированными, т. е. они представляют собой объединение входных каналов u всей последовательностной схемы с выходами y регистра R.

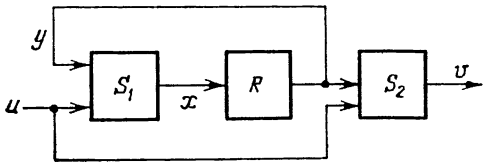

Рис. 1.3

Заметим, что описанную (последовательностную) схему можно рассматривать как комбинационную схему S_2, перестраиваемую *выходными сигналами регистра* R. В наиболее наглядном (хотя и заведомо упрощенном) виде такую перестройку можно представить себе следующим образом. Состояния регистра R, подающиеся на его выходные каналы, с помощью специальной комбинационной схемы K — так называемого *коммутатора* — переключают входной сигнал u на различные комбинационные схемы $S_2', S_2'', \ldots, S_2^{(m)}$, имеющие общий выход v и составляющие (вместе с коммутатором K) схему S_2 рис. 1.3. Схема подобной коммутации условно изображена на рис. 1.4. Как явствует из этого рисунка, конечный автомат может реализовать столько различных функций $v = f(u)$, задаваемых соответствующими комбинационными схемами, сколько различных состояний N может иметь память этого автомата. При двоичном регистре длины n имеем, очевидно, $N = 2^n$. Разумеется, не исключено, что различным состояниям регистра R соответствуют одинаковые схемы $S_2^{(i)}$, так что в общем случае $m \leqslant N = 2^n$.

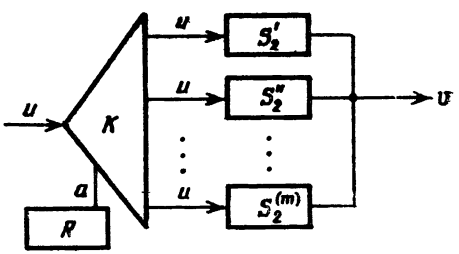

Рис. 1.4

Условность подобного представления заключается прежде всего в том, что в схеме с *чисто двоичными* сигналами нельзя переключить сигнал u на один из выходов, а на других выходах не иметь ничего (это был бы третий вид сигнала, отличный как

от 0, так и от 1). Кроме того, в подавляющем большинстве случаев схемы $S_2^{(i)}$ нецелесообразно строить отдельно одну от другой, так как при этом, вообще говоря, возрастает общее число используемых логических элементов. Однако эти условности не меняют главного — сделанных оценок для числа различных комбинационных схем, реализуемых конечным автоматом. Кроме того, при некоторых реализациях двоичных сигналов (например, импульсами различной полярности) в электронных схемах естественным образом реализуется и третий вид сигнала, а именно, отсутствие каких-либо импульсов. В этом случае предложенная интерпретация фактически теряет свою условность и может быть реализована практически.

1.7.1. Виды элементов памяти. Простейшим *элементом памяти* является схема, задерживающая входной сигнал на один микротакт, прежде чем подать его на выход. Функции переходов и выходов этого элемента, называемого *элементом задержки*, имеют вид $a(t)=x(t)$, $y(t)=a(t-\Delta t)$. Задержка в таком элементе специально планируется и, как правило, она значительно больше, чем в обычных (комбинационных) логических элементах. Необходимость подобного увеличения задержки в элементах памяти легко понять, взглянув на рис. 1.3. Схема S_1 на этом рисунке содержит в общем случае *обратные связи*, т. е. цепи логических элементов, передающие сигналы от выходов регистра R к его входам. Нормальное функционирование таких связей в обычных условиях требует, чтобы вызываемая ими суммарная задержка сигналов обратной связи не превышала времени такта работы элементов памяти. Хотя в схемах обратной связи по возможности избегают употребления длинных последовательно соединенных цепей логических элементов, величина такта обычно не менее чем на один десятичный порядок превосходит величину задержки, приходящуюся на один логический элемент. Таким образом, даже в экспериментальных схемах 1980 г. величина такта для наиболее быстродействующих полупроводниковых интегральных схем памяти измерялась по крайней мере единицами наносекунд (частота переключений в сотни мегагерц), а в лучших коммерческих схемах эта величина по крайней мере еще на порядок хуже.

Те же самые ограничения на быстродействие накладываются не только на элементы задержки, но и на другие элементы памяти, употребляющиеся на практике. Одним из наиболее распространенных типов элементов памяти являются так называемые (двоичные) *триггеры*. Триггер имеет два входных канала x_1 и x_2. При передаче единичного сигнала по первому каналу триггер устанавливается в состояние $a=1$, а по другому каналу — в состояние $a=0$. При нулевых сигналах по обоим каналам триггер сохраняет свое состояние.

Одновременная передача единичных сигналов по обоим входным каналам исключается путем специальных методов конструирования схемы обратной связи в конечном автомате. Выходной сигнал y_1 триггера совпадает с его текущим состоянием. Часто наряду с выходом y_1 триггер снабжается инверсным выходом $y_2 = \neg y_1$.

Еще один тип элемента памяти — *двоичный счетчик* — меняет свое состояние на противоположное с приходом единичного входного сигнала и сохраняет его при нулевом входном сигнале. Выходной сигнал этого элемента совпадает с его состоянием.

Заметим, что в качестве элементов памяти мы рассматриваем пока лишь такие элементы, входы и выходы которых могут непосредственно включаться в произвольные комбинационные схемы. Память такого рода мы будем называть *открытой*, в противоположность *закрытой памяти*, содержимое которой перед переработкой в комбинационных схемах нуждается в предварительном переносе его в открытую (регистровую) память. Пример памяти закрытого типа дает запись сигналов на магнитной ленте, а в большинстве случаев и на других магнитных носителях.

Присоединяя к полному набору логических (комбинационных) элементов открытые элементы памяти любого из перечисленных выше типов, мы получаем *полную систему* элементов для последовательностных схем. Под полнотой здесь понимается возможность построения из такой системы элементов последовательностной схемы, задающей любое отображение последовательностей входных сигналов на последовательности выходных сигналов, которое вообще может быть задано дискретным двоичным преобразователем информации с *конечной памятью*.

Заметим еще, что любой конечный автомат может рассматриваться как элемент с соответствующими функциями переходов $a(t) = f(a(t - \Delta t), u(t))$ и выходов $y(t) = g(a(t - \Delta t), x(t))$. Если при этом автомат не имеет памяти, то он вырождается в комбинационную схему, которая в случае необходимости может рассматриваться как логический элемент. Тем самым создается возможность для построения иерархии последовательно усложняющихся систем элементов, называемых, начиная со второго уровня, обычно уже не элементами, а *блоками*. Этот прием позволяет заметно упростить и сделать гораздо более наглядным проектирование сложных преобразователей дискретной информации путем так называемого *блочного синтеза*.

1.7.2. Автоматные отображения. Считая входные и выходные сигналы последовательностной схемы двоичными кодами букв некоторых абстрактных алфавитов X и Y, мы получаем возможность интерпретировать результаты работы такой схемы как некоторое *отображение* f слов в алфавите X (входных слов авто-

мата) в слова в алфавите Y (выходные слова автомата). Оно носит название *автоматного отображения* и зависит не только от самой схемы автомата, но и от начального состояния этого автомата (предполагается, что перед подачей очередного входного слова автомат устанавливается в это состояние).

От произвольных *алфавитных отображений* (слов алфавита X в слова алфавита Y) автоматные отображения отличаются рядом свойств, среди которых наиболее очевидно свойство равенства длин входных и выходных слов. Условие равенства длин вопросов и ответов, характерное для автоматного отображения в чистом виде, представляется на первый взгляд довольно серьезным ограничением. Его, однако, нетрудно устранить весьма несложным приемом с помощью выделения в алфавитах X и Y специальных *пустых* букв (знаков *пробела*), не несущих никакой содержательной нагрузки. Добавляя эту букву к началу и концу слов, можно всегда уравнять слова первоначально не равной длины.

Автором показано, как с помощью этого приема можно задать любую конечную *таблицу соответствия* между словами в алфавитах X и Y с помощью конечного автомата. Процедура построения автомата, индуцирующего заданное алфавитное отображение (вообще говоря, с точностью до пустых букв), называется процедурой *синтеза автомата*. Обратная процедура называется *анализом автомата*. Разработаны также многочисленные процедуры *минимизации*, позволяющие реализовать (по заданному отображению) автомат минимальной сложности. Поскольку комбинационные схемы представляют собой частный случай последовательностных схем (автоматы без памяти), то все указанные процедуры применимы также и к ним с соответствующими упрощениями.

Заметим, что разработанные методы синтеза автоматов позволяют в принципе построить конечный автомат, реализующий любую наперед заданную (конечную) систему диалогов (вопросов и ответов). В эту систему могут, в частности, войти диалоги, имитирующие любые процессы обучения и адаптации автоматов к изменяющимся условиям. На практике, однако, уровень формализации, используемый в теории конечных автоматов, оказывается слишком бедным для фактической реализации автоматов большой сложности. Алгоритмы синтеза и минимизации конечных автоматов оперируют с отдельными их состояниями. Когда число таких состояний измеряется миллионами (т. е. при числе разрядов более двадцати), алгоритмы синтеза автоматов становятся практически трудно реализуемыми даже при условии их полной автоматизации.

Для построения сложных преобразователей дискретной информации требуется дальнейшая специализация их внутреннего

строения и в первую очередь структуризация памяти и систем преобразуемых данных. Эта специализация позволяет реализовать процедуры блочного синтеза, о которых уже говорилось выше. Заметим, что подобная специализация призвана раскрывать лишь внутреннюю структуру преобразователя, который на любом уровне специализации представляет собой конечный автомат, поскольку на практике объем используемой в устройстве памяти всегда конечен. В теории, однако, удобно вводить преобразователи с бесконечной памятью, которые, разумеется, нельзя рассматривать как конечные автоматы.

1.8. Структуры данных и структурированная память

До сих пор мы говорили о данных как о некоторой аморфной массе, лишенной какой-либо внутренней организации. На практике в большинстве случаев данные образуют ту или иную *структуру. Структурирование данных* задается прежде всего с помощью различного рода *отношений порядка* (*упорядоченности*). Простейший вид упорядоченности задается простой нумерацией данных с помощью последовательных целых чисел. Идентификаторы упорядоченных таким образом данных образуются обычно путем приписывания какому-либо фиксированному идентификатору (скажем, x) специального *идентификатора индекса*, значения которого пробегают заданный набор целых чисел.

Возникающий таким образом идентификатор x_i (или $x[i]$), где i пробегает целые числа от m до n, идентифицирует упорядоченный набор данных, называемый обычно *одномерным массивом*. Двухиндексный идентификатор $x_{i,j}$ (или $x[i, j]$) идентифицирует *двумерный массив*, трехиндексный — *трехмерный массив* и т. д. В упорядоченных таким образом массивах возникает естественное *отношение следования*: так, следующим по индексу j для элемента $x_{i,j}$ будет элемент $x_{i,j+1}$, а предыдущим — элемент $x_{i,j-1}$. Если индекс j пробегает значения от p до q, то для p не существует предыдущего, а для q — следующего значения индекса.

Если границы изменений индексов задаются константами, то говорят, что соответствующий массив — *прямоугольный*, поскольку в двумерной целочисленной решетке область возможных значений индексов в этом случае представляет собой прямоугольник (хотя термин «прямоугольный» применяется для массивов любой размерности). Области значений индексов могут задаваться и более сложными способами. Например, соотношения $p \leqslant i \leqslant j \leqslant q$ задают так называемый *треугольный массив*. Массивы обычно предполагаются *однородными*, т. е. состоящими из элементов одного и того же типа. Одномерные однородные массивы (x_i) принято называть *векторами*, двумерные (x_{ij}) — *матри-*

цами, а составляющие их элементарные данные — их *компонентами* или *элементами*.

В общем случае индексы могут пробегать не обязательно последовательно все целые числа с шагом, равным единице, а любые их конечные подмножества, в том числе с переменной величиной шага. Дальнейшее обобщение выводит возможные значения индексов из класса целых чисел, заменяя числа элементами алгебр специального вида — так называемых *полугрупп*.

Помимо однородных массивов, на практике часто приходится иметь дело с данными различных типов. Рассмотрим, например, содержание обычной анкеты, применяемой при учете кадров. Одни графы этой анкеты (профессия) заполняются словами, другие (год рождения) — числами, третьи (имеете ли награды) — булевыми величинами (да, нет). В безбумажной информатике каждая совокупность данных (вообще говоря, различных типов), характеризующая тот или иной объект или явление, называется *записью*. Кадровая анкета представляет собой один из возможных видов подобной записи. Совокупность записей (обычно одного и того же типа) носит наименование *файла*. В частности, совокупность анкет на всех сотрудников какого-либо учреждения составляет *кадровый файл* этого учреждения.

Если запись в целом характеризует тот или иной объект, то отдельные ее позиции представляют собой различные свойства этого объекта, или, как обычно принято говорить, его *атрибуты*. Если запись состоит из n атрибутов, то каждый файл записей такого типа задает *n-местный предикат* (*n-местное отношение*) $f(x_1, x_2, \ldots, x_n)$ на множестве всевозможных значений этих атрибутов. Этот предикат считается *истинным* для значений атрибутов $x_1 = a_1$, $x_2 = a_2$, $\ldots$, $x_n = a_n$, если в заданном файле существует запись $\langle a_1, a_2, \ldots, a_n \rangle$, и *ложным* в противном случае. Таким образом, всякий набор файлов задает некоторую модель данных в смысле § 1.5.

Записи — это подмножества множества данных длины n, имеющие специальный вид. В общем случае структура данных может быть задана в виде произвольной системы N подмножеств множества M (упорядоченных или нет). Подмножества системы N получают при этом некоторые имена и включаются в рассмотрение в качестве *составных данных* наряду с исходными (элементарными) данными множества M.

Например, любая книга без рисунков может рассматриваться как упорядоченная последовательность букв (в число которых включаются знаки препинания и другие специальные символы, включая символ пробела). Эти буквы объединяются в слова, которые можно рассматривать как составные объекты первого уровня, слова — в предложения, предложения — в параграфы, а параграфы — в главы.

В структурированной таким образом книге объект (данное) каждого уровня (за исключением самого верхнего) входит в один и только один объект следующего (более высокого) уровня. Данные с такой структуризацией принято называть *деревьями* или (простыми) *иерархическими структурами*. Составной объект самого верхнего уровня называется *корнем*, а (элементарные) объекты самого нижнего уровня — *листьями* этого дерева (рис. 1.5).

Аналогично структурированию данных может производиться *структурирование памяти* в преобразователях дискретной информации. Уже для простейшего случая конечных автоматов сведенная в один регистр память обычно предполагается упорядоченной. В частности, в регистре выделяются самый младший и самый старший (булев) разряды этого регистра.

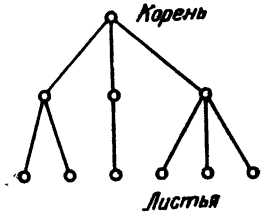

Рис. 1.5

При различных детализациях понятия конечного автомата его память разбивается на некоторое (конечное) множество регистров, каждый из которых получает свое *имя* или *номер*. *Запоминающие устройства* (сокращенно ЗУ) большого объема, как правило, разбиваются на *ячейки* одной и той же длины (обычно от одного до нескольких десятков байтов). Каждой ячейке присваивается некоторый целочисленный *адрес*. Обычно для нумерации ячеек используются неотрицательные целые числа, начиная с нуля. ЗУ могут собираться из отдельных *блоков*, образуя тем самым иерархические структуры.

1.8.1. Виды памяти и ее характеристики. Прежде всего следует различать *открытую* и *закрытую память*. В первом случае выходные каналы всех элементов памяти могут непосредственно подключаться к входам любых комбинационных схем. Во втором случае, прежде чем использовать в комбинационных схемах информацию, содержащуюся в той или иной ячейке ЗУ, ее необходимо прочесть, т. е. перевести ее (считать из памяти) в некоторый открытый регистр. Разумеется, предварительно нужная информация должна быть записана в соответствующую ячейку ЗУ.

Запоминающие устройства с *произвольным доступом* характеризуются одним и тем же (обычно весьма малым) *временем доступа* в произвольную ячейку независимо как от ее адреса, так и от предыдущих обращений в память. Под временем доступа здесь понимается время, необходимое для установления соединения данной ячейки ЗУ (задаваемой своим адресом) с некоторым (фиксированным) открытым регистром. Добавляя к этому времени время, необходимое для фактического перемещения ин-

формации из ЗУ в регистр или обратно, получают соответственно наименования времени *цикла чтения* и *цикла записи*.

Перечисленные характеристики определяют *быстродействие* ЗУ. При прочих равных условиях это быстродействие уменьшается с увеличением *объема* ЗУ. Поскольку байт представляет собой слишком малую единицу измерения, объем ЗУ обычно выражается в *килобайтах* (Кбайт или просто К), или *мегабайтах* (Мбайт). Поскольку адресация большинства ЗУ осуществляется в двоичной системе счисления, в качестве килобайта удобно брать не 1000 байт, а $1К = 1024 = 2^{10}$ байт. Соответственно, под мегабайтом обычно понимается величина 2^{20} байт.

В ЗУ *последовательного доступа* доступ осуществляется последовательно в порядке роста или убывания адресов ячеек (как при прокручивании магнитофона). Поэтому после обращения в ячейку с адресом n следующее соединение будет установлено либо с $(n+1)$-й, либо с $(n-1)$-й ячейкой. Последовательные ЗУ характеризуются (помимо объема) также скоростью передачи информации, которая в этом случае чаще всего одинакова как при записи, так и при считывании. Зная эту скорость, можно вычислить время доступа в любую ячейку в той или иной конкретной ситуации.

Например, при ячейках в 1 байт и скорости передачи 1 Мбайт/с после ячейки с нулевым адресом доступ в ячейку с адресом 100 000 будет осуществлен через 0,1 с, а в ячейку с адресом 3 млн.— через 3 с.

ЗУ с *прямым доступом* комбинируют черты устройств с произвольным и последовательным доступом. Обладая в принципе возможностью перескока от одного адреса к любому другому (как это имеет место при перемещении головки дискового проигрывателя), они вместе с тем гораздо быстрее открывают ячейки с последовательными адресами. Поэтому в их характеристику, помимо максимального времени произвольного доступа, входит, как и в последовательных ЗУ, независимая от этого времени скорость передачи информации.

При построении преобразователей дискретной информации часто оказывается удобным использовать ЗУ *магазинного типа*, называемые также *стеками*. Слова, запоминаемые в таком ЗУ, ведут себя так же, как патроны, закладываемые в обойму (магазин) пистолета: при вводе (записи) нового слова уже заполненные слова сдвигаются на один шаг в глубь магазина, давая место «новичку». При выводе (чтении) процесс протекает в обратном направлении. Таким образом, реализуется принцип: последнее (слово) из записанных читается первым. Стеки обычно реализуются в виде комплекса регистров, из которых самый верхний регистр (содержащий последнее из записанных данных) является открытым, а остальные — закрытыми.

Для ряда приложений очень удобна так называемая *ассоциативная память*. Запись в нее осуществляется в произвольную ячейку (выбираемую самим ЗУ). Доступ к нужному элементу данных осуществляется не по адресу, а по запоминаемому вместе с этим элементом *признаку*. Конструктивно такая память выполняется в виде матрицы x_{ij} с булевыми первичными ячейками. Вторичные (составные) ячейки (адресуемые индексом i) образуют строки этой матрицы, тогда как столбцы (адресуемые индексом j) несут в себе соответствующие разряды признаков всех слов-строк, запомненных в ЗУ. Для реализации, помимо обычного (адресного) доступа по строкам матрицы, также ассоциативного доступа, достаточно реализовать доступ по второму индексу — адресу столбца матрицы (с соответствующей схемой анализа выбранного столбца).

Более подробно о различных видах памяти, используемых в современной информатике, говорится в гл. III.

1.9. Процессоры

Процессором называется устройство, способное выполнять некоторый заданный набор операций над данными в структурированной памяти и вырабатывать значение заданного набора логических условий над этими данными.

Стандартная схема процессора состоит из двух устройств, называемых обычно *арифметико-логическим устройством* (АЛУ) и *устройством управления* (УУ). В схему АЛУ включается структурированная память, состоящая, как правило, из регистров, к которым может добавляться один или несколько стеков. С помощью специальных комбинационных схем в структурированной памяти может осуществляться тот или иной набор преобразований.

Как уже отмечалось выше, преобразования (операции), задаваемые комбинационными схемами, на сегодняшнем этапе развития микроэлектроники предпочитают делать достаточно простыми. Поэтому операции, выполняемые АЛУ за один такт синхронизирующего генератора, называются *микрооперациями*, а соответствующий их выполнению такт — *микротактом*. Выбор той или иной микрооперации осуществляется путем подачи кода этой микрооперации на специальный управляющий вход АЛУ.

Как правило, в состав микроопераций АЛУ включается так называемая *очистка* регистров и стеков, т. е. обращение их содержимого в нуль. Среди других часто встречающихся микроопераций отметим различного рода *пересылки данных* между регистрами и стеками, *сдвиги* (вправо или влево) двоичного кода на регистрах, а также операцию *накапливающего суммирования*. Для этой операции выделяются специальные регистры (с соот-

ветствующими комбинационными схемами), называемые *накапливающими сумматорами* (аккумуляторами). При передаче кода некоторого числа x в аккумулятор, содержавший число y, происходит суммирование (с учетом знаков) этих чисел, а сумма $x+y$ замещает в аккумуляторе его первоначальное содержимое y.

Помимо описанных *внутренних микроопераций*, в АЛУ реализуются также *внешние микрооперации*, осуществляющие *прием и выдачу* данных, т. е. *обмен данными* между АЛУ и внешним миром.

Наконец, АЛУ может формировать дополнительно выходные сигналы в виде набора так называемых *логических условий*, выражающих те или иные свойства содержимого памяти АЛУ.

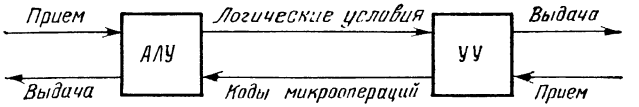

Рис. 1.6

Примеры таких свойств: «коды в регистрах A и B одинаковы», «число, содержащееся в регистре A, отрицательно», «младший разряд двоичного кода в регистре A равен нулю» и т. д.

Устройство управления процессора представляет собой конечный автомат, входными сигналами которого служат наборы логических условий, формируемых АЛУ, а выходными сигналами — коды микроопераций АЛУ. Кроме того, УУ может иметь также каналы обмена с внешним миром. Тем самым схема процессора получается такой, как на рис. 1.6. Каналы выдачи и приема УУ и АЛУ обычно объединяются в каналы обмена процессора с внешним миром.

Через канал приема в УУ поступают коды *команд* (инструкций) на выполнение тех или иных *операций процессора*, а через канал выдачи — запросы на очередные команды, либо так называемый сигнал *останова*, который прекращает передачу в процессор синхронизирующих сигналов и тем самым прекращает его работу. Операция процессора обычно не сводится к одной микрооперации (хотя это в принципе и не исключено), а индуцируется некоторой последовательностью микроопераций. Такие последовательности определяются так называемыми *микропрограммами*, встраиваемыми или запоминаемыми в УУ. В случае запоминаемых микропрограмм их можно оперативно менять, настраивая процессор на различные наборы операций (команд). Принято говорить в таком случае, что процессор имеет *мягкую архитектуру* (т. е. перестраиваемую).

1.9.1. Режимы работы процессоров. Прежде всего можно выделить процессор в отдельное законченное устройство, вводя в него код операции и исходные данные вручную. Выходные данные (результат операции) выдаются на специальное *устройство отображения* для непосредственного восприятия человеком. При этом процессор превращается в *калькулятор.*

При подобном *ручном режиме* использования процессора автоматизируется только часть труда по обработке информации. Трудоемкие операции поиска информации, выбора нужных для ее обработки команд, ввод информации и соответствующей ей команды в калькулятор, а также фиксация промежуточных результатов выполняются вручную (обычно на бумаге). Использование калькуляторов сохраняет в основном традиционную (бумажную) форму обработки информации и не может обеспечить высокую скорость обработки данных, особенно для достаточно длинных последовательностей команд.

Автоматический режим работы процессора может осуществляться двумя различными способами. В первом из них процессор сам извлекает последовательно необходимые команды из *программы* (последовательности команд, запомненных в быстродействующей памяти — внешней по отношению к процессору). Все необходимые данные (исходные, промежуточные и выходные) также хранятся в этой памяти. Команды, помимо кода операции, содержат адреса данных, необходимых для их исполнения. Извлекая из памяти очередную команду и анализируя ее, процессор тут же извлекает необходимые для ее выполнения данные. В этом случае принято говорить, что процессор управляется *потоком команд.* Отрезок времени, в течение которого происходит выполнение команды, называется *циклом.*

Другой способ состоит в том, что данные, поступающие в АЛУ процессора из какого-нибудь внешнего источника, запоминаются в нем, анализируются и вызывают (с помощью УУ) выборку из быстродействующей памяти последовательности команд для их обработки. В этом режиме, называемом *режимом реального времени*, процессор управляется *потоком данных.* Быстродействие процессора должно быть достаточно велико, чтобы обработать поступившие данные (и выдать результат обработки) до поступления новой, очередной порции данных.

1.9.2. Микропрограммирование. Построение УУ делается особенно наглядным, когда в его состав включается специальная *управляющая память*, в которой запоминаются микропрограммы всех индуцируемых им операций. *Микропрограмма* представляет собой конечную последовательность правил — так называемых микрокоманд (микроинструкций). Каждая микрокоманда содержит код некоторой микрооперации и номер микрокоманды (или адрес хранящей ее ячейки управляющего ЗУ), которая должна

быть выполнена после исполнения данной микрокоманды. Если указание о таком *скачке* (переходе) в микрокоманде отсутствует, то управление передается следующей по порядку микрокоманде микропрограммы. В конце выполнения микропрограммы при ручном управлении осуществляется останов, а при автоматическом — выбор следующей операции и переход к соответствующей новой микропрограмме.

В качестве примера рассмотрим микропрограмму умножения двух целых неотрицательных чисел на АЛУ с тремя регистрами a, b, c, из которых последний представляет собой накапливающий сумматор. Основной фрагмент требуемой микропрограммы можно представить четырьмя следующими микрокомандами:

1. Очистить регистр c (т. е. заслать в него число нуль).
2. Если младший разряд числа в регистре b равен 1, то передать число из регистра a в регистр (накапливающий сумматор) c; в противном случае передать в c нуль.
3. Сдвинуть влево на один разряд содержимое регистра a.
4. Сдвинуть вправо на один разряд содержимое регистра b.

Легко видеть, что выписанная микропрограмма выполняет один шаг обычного умножения «в столбик» в двоичной системе счисления. Микрокоманда 2 умножает множимое (содержимое регистра a) на младший разряд множителя (содержимого регистра b), а микрокоманды 3 и 4 готовят множимое и множитель к следующему шагу.

Чтобы получить из приведенного фрагмента полную микропрограмму, можно поступить двумя различными путями. Во-первых, повторить n раз микрокоманды 2, 3, 4 (где n — максимальное число значащих разрядов во множимом или в множителе), во-вторых, использовать прием образования цикла: повторять выполнение микрокоманд 2, 3, 4, пока в регистре b будет ненулевое число; иначе окончить выполнение микропрограммы.

Заметим, что при n-разрядных множимом и множителе регистры a и c, чтобы не потерять старших разрядов произведения, должны иметь разрядность $2n$.

Объем памяти УУ определяется общим количеством микрокоманд в микропрограммах всех команд, выполняемых процессором. На первый взгляд кажется, что это число просто равно сумме длин всех микропрограмм. На самом деле оказывается возможным экономить память за счет объединения общих участков в микропрограммах для различных операций. В случае сложных микропрограмм этот процесс можно повторить несколько раз: наряду с самыми мелкими общими частями на следующем шаге из них строятся общие куски более высокого уровня и т. д. Подобная *ступенчатая* организация микропрограмм (реализованная впервые в ЭВМ МИР-1) позволяет существенно экономить память.

1.10. Алгоритмы и алгоритмические языки

Только что рассмотренные микропрограммы представляют собой частный случай алгоритмов. *Алгоритмом* принято называть любую конечную систему правил преобразования информации (данных) над любым конечным алфавитом.

Для описания как самой системы правил, так и перерабатываемой с их помощью информации служат так называемые *алгоритмические языки*. Любой алгоритмический язык задается над некоторым фиксированным конечным алфавитом, с помощью букв которого задаются структурированные данные и правила их переработки. Буквы алфавита представляют собой первичный тип *элементарных данных*, а составляемые из них слова могут рассматриваться как одномерные массивы.

Разумеется, помимо букв основного (первичного) алфавита, в алгоритмическом языке могут задаваться и другие типы элементарных данных. Так, в языках, описывающих информационные процессы в современных ЭВМ,— так называемых *машинных языках* — первичным является двоичный алфавит. Что же касается типов элементарных данных, то к ним обычно относятся числа заданного формата в двоичной системе счисления, а также буквы байтового алфавита, составляющие *вторичный алфавит* ЭВМ.

Помимо элементарных данных, в алгоритмические языки могут включаться различного рода структуры данных, в частности массивы, файлы и т. д. В последние годы все большее распространение получают языки, в которых имеются средства для определения, в случае необходимости, новых типов данных, например комплексных чисел.

Следующей составной частью алгоритмических языков являются различного рода *операции над данными*. Так, если в языке определяется тип данных «вещественное число» (реально — приближение вещественных чисел рациональными числами с некоторой заданной степенью точности), то для него обычно определяются четыре арифметические операции, а также операция возведения в степень. Для типа данных «булевы» определяются операции дизъюнкции, конъюнкции и отрицания, а иногда и некоторые дополнительные операции. С помощью операций строятся алгебраические *формулы*, называемые в прикладной теории алгоритмов обычно *выражениями* (арифметическими, булевыми и т. д.).

Третья составная часть языка — определенные на множестве структурированных данных *логические условия*. Для любых типов данных в языки, как правило, вводятся отношения равенства и неравенства $a = b$, $a \neq b$, представляющие собой элементарные логические условия. Для типов данных «вещественное чис-

ло» или «целое число» обычно используются бинарные отношения в виде различного рода «направленных» неравенств ($>$, $<$, $\geqslant$, $\leqslant$).

В развитых алгоритмических языках операции и отношения в арифметических и булевых выражениях могут комбинироваться. Например, выражение «a^2, если $a < 1$, иначе a^3» может трактоваться как арифметическое выражение, равное a^2 при $a < 1$ и a^3 при $a \geqslant 1$. Аналогично, отношения равенства и неравенства могут определяться не только между элементарными данными, но и между целыми выражениями, например, $a + b \geqslant x - y$.

С помощью введенных выражений можно из одних данных получать другие. Для окончательного оформления преобразования данных, задаваемых теми или иными выражениями, остается лишь найти место для получаемых результатов в общей системе рассматриваемых данных. В случае машинных языков это достигается путем записи (засылки) результата операции в ту или иную ячейку структурированной памяти. Ячейка при этом обычно задается своим адресом. Соответственно, исходные данные также берутся из каких-либо ячеек памяти, так что идентификаторы переменных в формулах для машинных языков интерпретируются как адреса ячеек памяти, содержащих текущие значения этих переменных.

В так называемых *процедурно-ориентированных* и *проблемно-ориентированных языках* понятие памяти не вводится явно. Включение тех или иных данных (получаемых в результате операций) в общую систему данных получается с помощью *присвоения* получаемых значений некоторым идентификаторам. Таким образом, возникает понятие так называемого *оператора присваивания*, схема которого имеет вид: ⟨идентификатор⟩ := ⟨выражение⟩. Здесь собственно операция присваивания (значение идентификатора в левой части становится равным значению правой части) обозначается специальным символом :=, введенным специально для того, чтобы не путать операцию присваивания с отношением равенства. Например, $x := x + 1$ означает, что идентификатор (переменная) x меняет свое значение, увеличивая его на 1.

В то же время выражение $x = x + 1$, эквивалентное отношению $0 = 1$, представляет собой просто тождественно ложное булево выражение, т. е. булеву константу «ложь».

В общем случае в понятие оператора алгоритмического языка входит еще и указание на то, какой *следующий* оператор нужно выполнять после него. Произвольный алгоритм, называемый в случае машинных языков *программой*, представляет собой конечную упорядоченную последовательность операторов, которые обычно (если не указано противное) выполняются в порядке их следования. Если же по той или иной причине необхо-

димо изменить порядок выполнения операторов, то в текущем операторе указывается *метка* оператора, который должен быть выполнен после него. Вместо специальной метки может использоваться просто *номер* оператора в алгоритме (программе).

В чистом виде операция изменения порядка следования операторов представлена так называемым *оператором безусловного перехода*, обозначаемым в большинстве языков английским выражением go to L («идти на L»), где L — некоторая метка.

Для обработки массивов в алгоритмические языки включаются специальные *операторы цикла*. Обычная форма оператора цикла имеет вид:

для x_i от $i = m$ шаг k до $i = n$ выполнить A

где A — оператор, i — идентификатор индекса, k, m, n — целые числа (в некоторых языках — арифметические выражения). Оператор A согласно этому выражению должен быть применен к элементам x_i массива с индексами $i = m$, $m + k$, $m + 2k$, ..., n. Например, оператор цикла:

для x_i от $i = 0$ шаг 2 до $i = 1000$ выполнить $x_i = x_i + 1$

увеличивает на единицу значения всех элементов массива $x_i (0 \leqslant i \leqslant 1000)$ с четными индексами. Аналогично определяются операторы цикла и в случае многомерных массивов, например:

для x_{ij} от $i = m$ шаг k до $i = n$, от $j = p$ шаг q до $j = r$ выполнить A

С помощью специальных *операторных скобок*, в качестве которых во многих языках употребляется пара английских слов **begin**, **end** (начало, конец), в алгоритме можно выделять отдельные *блоки*, внутри которых могут определяться свои системы обозначений данных. Они задаются специальными *описаниями*, помещаемыми в начале блока. Блок может рассматриваться как сложный, так называемый *составной оператор*, который может получить *собственное сокращенное обозначение — имя* составного оператора — и использоваться наряду с обычными операторами при построении сложных алгоритмов. В случае машинных языков такие блоки получают наименование *макрооператоров* или *подпрограмм*.

Еще двумя (особыми) типами сложных операторов являются так называемые условные операторы и операторы итерационного цикла.

Условный оператор чаще всего имеет вид:

если α, то A, иначе B

При его выполнении прежде всего проверяется логическое условие α. Если оно истинно, то выполняется оператор A; в против-

ном случае — оператор B. Иногда условный оператор имеет и другую (сокращенную) форму:

если α, то A

Он применяет оператор A, если условие истинно, и не выполняет никакого действия, или, как говорят, эквивалентен *пустому оператору*, в противном случае. Если вместо A подставить оператор go to L, то получим *оператор условного перехода*. Он передает работу, или, как обычно говорят, управление, оператору с меткой L, если условие α истинно, и следующему по порядку оператору в противном случае.

Оператор итерационного цикла имеет вид:

пока α, выполнять A

Он означает, что оператор A выполняется до тех пор, пока логическое условие α является истинным. В противном случае управление передается следующему по порядку оператору алгоритма.

Алгоритм считается закончившим свою работу, если выполнен его последний оператор, за которым не следует никакого другого оператора.

1.10.1. Полнота алгоритмических языков. Алгоритмические языки, которые имеют все перечисленные выше средства, обладают так называемым свойством *полноты*. Под этим понимается возможность построить в этом языке (L) алгоритм K, выполняющий с точностью до кодирования работу любого наперед заданного алгоритма M в произвольном алфавите A. Слова «с точностью до кодирования» понимаются в том смысле, что буквы алфавита A кодируются каким-либо (фиксированным) способом с помощью элементарных данных рассматриваемого алгоритмического языка L, после чего алгоритм K любые кодированные этим способом исходные данные D переводит в кодированные этим же способом данные $M(D)$, где через $M(D)$ обозначены данные на выходе алгоритма M.

Поскольку в этом определении участвует тот или иной тип элементарных данных, то более точно говорить не просто о полноте языка, а о *полноте над данным типом элементарных данных*. Используя теорию так называемых *нормальных алгоритмов* Маркова, можно показать, что алгоритмический язык L полон над некоторым типом T элементарных данных, если в нем может быть построен алгоритм, который для любой тройки векторов (одномерных массивов) x, a, b может определить, входит ли массив a в качестве подмассива в массив x, заменить первое (слева) из таких вхождений на массив b и в зависимости от успеха или неуспеха поиска такого вхождения передать управление в две разные точки алгоритма.

Например, в двоичном алфавите при $x = 011110111$, $A = 111$, $B = 01$ A входит в x в качестве подмассива три раза: $x = 0(111)10111$ — самый левый из них выделен скобками. После замены этого вхождения массивом B получаем массив $0(01)10111$ или, после удаления скобок, 00110111.

1.11. Алгоритмические системы

Алгоритмические языки представляют лишь *средства для записи* алгоритмов. Для их фактического исполнения требуется добавить некоторую *интерпретирующую систему*. В качестве такой системы может, разумеется, выступать человек, снабженный карандашом и бумагой, для фактического выполнения предписываемых алгоритмом преобразований. Выступая в качестве интерпретатора записей в алгоритмическом языке, человек превращает этот язык в алгоритмическую систему. В автоматическом режиме исполнение различных операций и нахождение значений логических условий алгоритмического языка возлагаются на процессор с соответствующей системой операций и условий. Однако процессор может исполнять лишь отдельные операции. Для исполнения *программ*, т. е. произвольных (конечных) последовательностей операций, его необходимо дополнить *запоминающим устройством* (ЗУ).

Запоминающее устройство предназначено для запоминания как всех видов данных (входных, промежуточных и выходных), так и программ их обработки. Соединяя определенным образом ЗУ с процессором, получают автоматическую интерпретирующую систему S для алгоритмического языка L, операции и условия которого встроены в процессор. Язык L вместе с системой S составляют автоматическую *алгоритмическую систему*, которую мы будем называть *идеализированным компьютером*.

Идеализация, о которой здесь идет речь, состоит прежде всего в том, что мы игнорируем пока операции ввода и вывода данных из компьютера. Кроме того, имеется в виду то обстоятельство, что длина программ и объем данных, хотя и предполагаются конечными, могут быть как угодно велики. Поэтому ЗУ в идеализированном компьютере предполагается бесконечным или, более точно, содержащим бесконечное число ячеек. Что касается самих ячеек, то они будут предполагаться в дальнейшем конечными. Их размер должен позволять помещать в них элементарные данные всех встречающихся в языке типов (по одному на ячейку), включая *коды* всех операторов этого языка — команды идеализированного компьютера.

Интерпретируемый таким компьютером алгоритмический язык в принципе может быть любым (не обязательно машинно-ориентированным). Для этого, однако, может дополнительно потребо-

ваться, например, установление взаимно однозначного соответствия между множеством ячеек ЗУ и множеством идентификаторов рассматриваемого языка.

1.11.1. Компьютер фон Неймана. Дальнейшее уточнение понятия идеализированного компьютера предусматривает специальную *машинную ориентацию* интерпретируемых им алгоритмических языков и простейшую (линейную) структуризацию ЗУ. Мы ограничимся здесь лишь такими ЗУ, у которых все ячейки имеют одну и ту же конечную длину и нумеруются последовательными целыми числами, начиная с нуля. Коды, с которыми оперируют ЗУ и процессор, являются двоичными кодами, причем принимается, что в одной ячейке может храниться либо код *операнда* (элементарного данного), либо код *машинной команды* (оператора языка).

Код машинной команды состоит из *кода операции* и *адресной части*, в которой указываются адреса операндов и, возможно, адрес следующей команды. Конечность кода команды предопределяет и конечность множества кодируемых в ней адресов. Поэтому непосредственно адресуемая процессором память компьютера всегда ограничена реальной длиной кода адреса: при длине кода адреса, равной n, может адресоваться максимум 2^n ячеек памяти.

Для расширения объема адресуемой памяти может использоваться так называемый *базовый регистр*, на котором с помощью специальных команд могут формироваться старшие разряды адреса. При адресации ЗУ к содержимому этого регистра присоединяется (в качестве младших разрядов) значение адреса, указываемое в исполняемой команде. Этот прием позволяет в принципе адресовать память неограниченного объема.

Полнота машинного языка (или, иначе говоря, полнота системы команд) подобного компьютера обеспечивается при наличии в нем возможности, во-первых, организовать двусторонний обмен (чтение — запись) между ячейкой ЗУ и фиксированным регистром процессора (обычно аккумулятором), различать соседние ячейки и выполнять условный переход (на команды в любых заданных ячейках ЗУ) по условию равенства (или неравенства) кодов в любой паре ячеек ЗУ. Необходима также специальная команда останова для окончания работы машины.

В случае бесконечной памяти выполнение перечисленных условий позволяет реализовать в подобном идеализированном компьютере любые алгоритмы (с точностью до кодирования). На практике, разумеется, память компьютера будет всегда конечной. С целью хотя бы частичной компенсации этого недостатка к подобной *оперативной памяти* (ОЗУ) с произвольным доступом добавляют *внешнюю память* (ВЗУ) с последовательным или прямым доступом. Кроме того, компьютер снабжается устройст-

вами для *ввода* и *вывода* информации. Система команд при этом пополняется командами обмена между внешней памятью и устройствами ввода-вывода, с одной стороны, и оперативной памятью — с другой. Идеализированный компьютер с указанными ограничениями и добавлениями носит наименование *компьютера* или *машины фон Неймана* по имени американского математика, предложившего такую структуру машины для автоматической обработки информации (рис. 1.7). Штриховой линией показаны пути передачи команд управления на устройства ввода-вывода и ВЗУ.

Конечная память уже не дает возможности реализации на машине фон Неймана любых алгоритмов. Однако это может быть

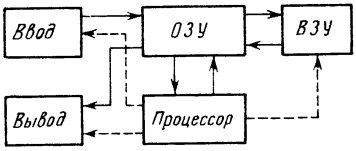

Рис. 1.7

сделано в предположении, что внешняя память может наращиваться неограниченно. Поэтому компьютер фон Неймана с полной системой команд называется обычно *универсальным компьютером.*

Заметим, что при описании компьютера фон Неймана нигде не использовалось то обстоятельство, что его компоненты являются электронными устройствами. В принципе он может быть построен на элементах любой природы (механической, химической, биологической и др.). Однако на практике лишь электроника дала сегодня возможность построить *реально действующие* универсальные компьютеры фон Неймана. Кроме того, оказывается, что универсальность по отношению к *вычислительным* алгоритмам обеспечивает (с точностью до кодирования) универсальность по отношению к *произвольным* алгоритмам. Поэтому за компьютерами закрепилось не вполне отражающее их сущность наименование *электронных вычислительных машин* (ЭВМ), к которому иногда добавляется термин «цифровые», чтобы отличать их от электронных вычислительных машин, работающих с аналоговой информацией.

1.11.2. Система обработки данных. Первые ЭВМ строились по простейшей схеме фон Неймана. Однако вскоре требования практики потребовали значительно усложнить структуру, или, как сейчас принято говорить, *архитектуру* ЭВМ. В их состав включаются сейчас не только *центральные процессоры* с универсальным набором команд, но и различного рода специализированные процессоры, предназначенные прежде всего для освобож-

дения центрального процессора от операций ввода-вывода. Наличие таких процессоров позволило подключать к ЭВМ большое число разнообразных *периферийных внешних устройств*. К их числу относятся устройства внешней памяти, устройства ввода-вывода, устройства сопряжения с каналами связи.

Для управления всем этим сложным хозяйством разрабатываются специальные системы программ — так называемые *операционные системы*, позволяющие гибко менять конфигурацию системы, эффективно загружать оборудование и облегчить написание и отладку *прикладных программ* пользователей. Программная («мягкая») часть системы (software) вместе с «твердой» частью (hardware), т. е. собственно с оборудованием, составляют сложную информационную систему обработки данных, для которой лишь по традиции сохраняется старое название ЭВМ.

Глава II

ПРЕОБРАЗОВАТЕЛИ ИНФОРМАЦИИ

2.1. Преобразователи формы представления информации

Простейшими преобразователями информации являются *аналого-цифровые* и *цифро-аналоговые преобразователи* (АЦП и ЦПА), в задачу которых входит лишь преобразование формы представления информации. *Аналоговая информация* наиболее часто представляется в виде угла поворота вала. Другие часто встречающиеся формы представления информации (в виде величины давления, температуры, магнитного поля, механических колебаний, силы света, уровня радиации и др.) с помощью простых приборов легко преобразуются к трем перечисленным основным формам. Поэтому большинство АЦП и ЦПА строятся применительно именно к этим трем формам, хотя необходимость простых аналого-цифровых и цифро-аналоговых преобразований возникает иногда применительно и к другим формам представления аналоговой информации (прежде всего это касается частоты и фазы электрических синусоидальных колебаний).

Не имея возможности рассказать о всем многообразии современных АЦП и ЦПА, ограничимся рассмотрением лишь отдельных примеров. Опишем, например, *время-импульсный* преобразователь напряжения в двоичный код. Он состоит из генератора пилообразного напряжения (рис. 2.1), генератора импульсов постоянной частоты и специальной схемы сравнения, фиксирующей моменты времени, когда напряжение равно 0 и измеряемому напряжению U. Специальный счетчик считает импульсы между этими двумя моментами времени. Тем самым на счетчике возникает цифровой код измеряемого напряжения. Точность подобного преобразователя определяется ча-

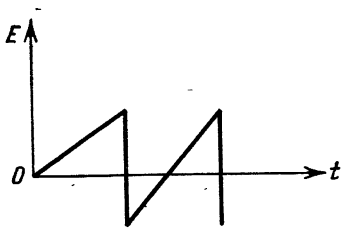

Рис. 2.1

стотой следования импульсов, стабильностью этой частоты, а также точностью выработки пилообразного напряжения. В лучших образцах точность преобразования достигает 0,2—0,1%. Скорость работы преобразователя достигает 10 000 преобразований в секунду, но чаще всего колеблется в пределах от 1000 до 5000.

При преобразованиях угла поворота вала в цифровой код можно использовать счет меток (или непосредственное считывание кодов) на специальных кодовых дисках, насаженных на этот вал. Обратное преобразование может быть осуществлено с помощью так называемых *шаговых двигателей.*

Один из простейших преобразователей двоичного кода в напряжение использует так называемый *генератор калиброванных импульсов.* Вырабатываемые им импульсы по амплитуде различаются в 2, 4, 8 и вообще в 2^n раз. Подавая калиброванные импульсы, соответствующие разрядам двоичного кода, в которых стоит единица, на специальный конденсатор, мы заряжаем этот конденсатор до напряжения, пропорционального данному двоичному коду. Скорость работы такого преобразователя может быть сделана достаточно большой, однако его точность невелика (ошибка до 10—15%). Другие типы преобразователей могут обеспечить несколько большую точность.

Следует заметить, что при прочих равных условиях при аналого-дискретных и дискретно-аналоговых преобразованиях увеличение скорости приводит к уменьшению точности. Имеются аналого-дискретные преобразователи, обеспечивающие до 400—800 тыс. преобразований в секунду с точностью 0,4% и 25—50 тыс. преобразований в секунду с точностью 0,1%.

2.2. Функциональные дискретные преобразователи

Если требуется задать любую функцию $y = f(x)$, в которой x и y — некоторые двоичные коды (булевы векторы), то, как уже отмечалось в гл. I, эта задача в принципе может быть решена с помощью некоторой *комбинационной схемы.* На практике наряду со специальными комбинационными схемами, являющимися частью конструкции более сложных устройств, получили достаточно широкое применение *схемы общего применения.*

Примером таких схем является так называемый *дешифратор.* Он имеет n двоичных входов и $N = 2^n$ двоичных выходов. Когда на вход подается некоторый сигнал $x = \langle x_1, x_2, \ldots, x_n \rangle$, то возбуждается (выдает единичный сигнал) один и только один выход коммутатора. Дешифраторы находят применение для управления сигналом кода этого канала (открытия и закрытия каналов передачи данных), в качестве которого обычно принимается номер этого канала в двоичном представлении. Поэтому дешифраторы носят также и другое название — *коммутаторы.*

По мере развития электроники и безбумажной информатики появляются все новые типы функциональных дискретных преобразователей. Мы ограничимся, однако, лишь одним приведенным примером.

Как уже отмечалось в гл. I, для построения любой комбинационной схемы достаточно элементарных схем трех типов, реализующих логические функции конъюнкции, дизъюнкции и отрицания. Соответствующие схемы принято называть схемами *совпадения*, *разделения* и *инверсии сигнала*. Теоретически можно свести универсальный набор элементов к одному-единственному типу. Однако мы этим заниматься здесь не будем. Более того, учитывая реальные свойства полупроводниковых схем (являющихся сегодня основой электроники), целесообразно несколько изменить подход к выбору элементной базы. Один из возможных подходов берет за основу схему ключа.

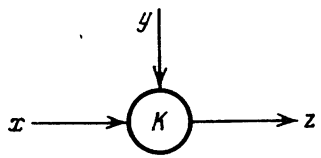

Рис. 2.2

2.2.1. Ключ. Очень часто в схемах современных ЭВМ в качестве основного логического элемента употребляется так называемый *ключ* (рис. 2.2). Нужно различать два типа ключей. Первый из них представляет собой по существу логическую схему совпадения $z = x \wedge y$. Однако, учитывая некоторые особенности физической реализации подобных схем, удобно пользоваться несколько иной трактовкой их работы. Особенность, о которой идет речь, состоит в том, что более слабым сигналом y можно управлять пропусканием через ключ более сильного сигнала x.

В ключе первого типа единичный сигнал на входе y пропускает сигнал x на выход (открывает ключ), т. е. $z = x$ при $y = 1$, а нулевой сигнал запирает ключ, т. е. $z = 0$ при $y = 0$. Если подать на вход x сигнал от постоянного сильного источника, тождественно равный единице, то ключ будет выступать просто как усилитель слабого сигнала y; при этом реализуемая им функция тривиальна, а именно, $z \equiv y$.

В ключе второго (инверторного) типа положение меняется на обратное. Сигнал $y = 0$ открывает, а сигнал $y = 1$ запирает ключ. При $x = 1$ будем иметь $z \equiv \neg y$. Таким образом, данный тип ключа может рассматриваться как усилитель — *инвертор*.

2.2.2. Функционально полные системы элементов. Если использовать два указанных вида ключа как двухвходовые элементы, то с их помощью можно построить любую комбинационную схему (поскольку в составе элементов имеются схемы «и» и «не»). Однако с целью упрощения получаемых схем в систему дополнительно включаются схемы «и» и «или», причем обычно не в двухвходовом, а во многовходовом исполнении. Схемы этого

типа могут быть реализованы на полупроводниковых диодах и резисторах, как это показано на рис. 2.3 и 2.4.

Если сигнал 1 представляется высоким напряжением ($+E_2$), а сигнал 0 — низким напряжением ($-E_1$), то на схеме рис. 2.3 единичный сигнал (высокое напряжение) появляется тогда и только тогда, когда хотя бы на один из входов $x_1, x_2, \ldots, x_n$ подано высокое напряжение (единичный сигнал). Таким образом,

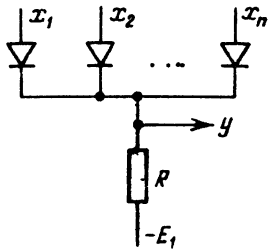

Рис. 2.3

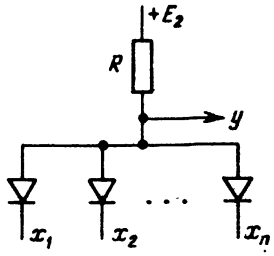

Рис. 2.4

эта схема есть схема «или». Аналогично, нетрудно видеть, что на выходе y схемы рис. 2.4 единичный сигнал (высокое напряжение) появляется тогда и только тогда, когда высокое напряжение (единичный сигнал) будет подан на все входы $x_1, x_2, \ldots, x_n$. Таким образом, эта схема представляет собой схему «и».

Для реализации инвертора и ключей обычных диодов и сопротивлений оказывается уже недостаточно. Требуются так называемые активные элементы, в качестве которых сегодня в основном используются *транзисторы* (полупроводниковые триоды). Один из типов таких транзисторов, называемых *полевыми* транзисторами, представляет собой по существу обычный ключ (см. рис. 2.2): при подаче на его управляющий вход y высокого напряжения ключ отпирается и пропускает ток от входа x к выходу z. При низком напряжении на входе y транзистор запирается и не проводит тока. Схема инвертора в таком случае может быть задана так, как изображено на рис. 2.5. Для сигнала на выходе y, очевидно, $y = \neg x$.

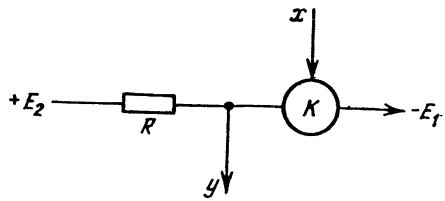

Рис. 2.5

2.2.3. Синтез функциональных преобразователей. Простейший метод синтеза, т. е. построения схемы функционального преобразователя, состоит в том, что заданная векторная функция

$y=(y_1, \ldots, y_m)=f(x_1, \ldots, x_n)$ разбивается на отдельные компоненты $y_i=f_i(x_1, \ldots, x_n)$. Каждая из таких функций $y(x_1, \ldots, x_n)$ очень простым способом может быть представлена в виде формулы специального вида, а именно дизъюнкции конъюнкций K_i вида $\tilde{x}_1 \wedge \tilde{x}_2 \wedge \ldots \wedge \tilde{x}_n$, где $\tilde{x}_k$ равно либо x_k, либо $\neg x_k$. Далее, используя тождественные преобразования в булевой алгебре (подобно тому, как это имеет место в обычной алгебре), приводят ее к виду, в котором все операции соответствуют выбранной системе элементов.

Описанный метод оказывается на практике малоэкономичным, поскольку схемы, реализующие отдельные функции f_i, могут иметь общие участки. К настоящему времени разработаны достаточно эффективные методы синтеза и минимизации комбинационных схем. Эти методы воплощены в автоматизированных системах проектирования и успешно применяются при разработках новых ЭВМ.

2.3. Конечные автоматы

Как уже отмечалось в гл. I, с позиций теории конечных автоматов на практике удобно рассматривать лишь устройства с небольшим объемом памяти (порядка двух десятков битов). Однако даже при таком ограничении находится много самостоятельных применений схем конечных автоматов. Это, прежде всего, различного рода *контроллеры* для управления относительно несложными устройствами, счетчики сигналов, счетчики-дешифраторы и др.

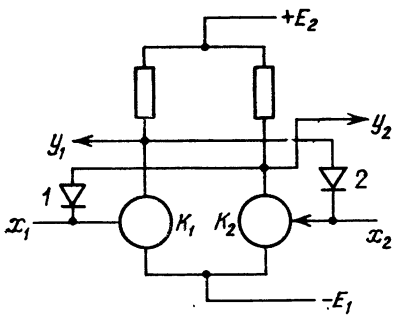

Рис. 2.6

Для построения конечных автоматов достаточно к полному набору комбинационных элементов присоединить элементы памяти одного какого-либо типа. Один из таких элементов — триггер — нетрудно получить, соединяя два инвертора, как это показано на рис. 2.6 (в несколько упрощенном виде). Подавая на вход постоянный единичный сигнал, легко заметить, что изображенная на рисунке схема, называемая обычно *триггером*, может находиться в одном из двух устойчивых состояний. Если ключ K_1 открыт, то сигнал на его выходе (y_1) равен 0. Поступая (через диод 2) на вход ключа K_2, этот сигнал запирает K_2. Следовательно, сигнал на его выходе (y_2) равен 1. Поступая (через диод 1) на вход ключа K_1, этот сигнал поддерживает его в открытом состоянии. Ввиду симметричности

схемы возможно и обратное — когда ключ K_2 открыт и своим выходным сигналом запирает ключ K_1.

Если триггер находится в первом состоянии (т. е. ключ K_1 открыт), то, подавая на вход ключа K_2 сигнал $x_2 = 1$, мы отпираем этот вентиль, в результате чего триггер переходит (или, как обычно говорят, перебрасывается) во второе состояние. Обратный переход совершается подачей сигнала $x_1 = 1$ на вход ключа K_1. В случае, когда триггер должен сохранить свое состояние, на входы подаются нулевые сигналы.

2.4. Процессоры

Определение и принципы работы процессоров уже были изложены в гл. I. Наша задача — описать более подробно различные типы процессоров, употребляемые в современной практике.

2.4.1. Арифметические процессоры. *Арифметическими* называются процессоры, предназначенные в первую очередь для автоматизации вычислительных операций той или иной степени сложности. Для сложных научных, проектно-конструкторских и планово-экономических расчетов наиболее употребительны процессоры, выполняющие полный набор арифметических операций (сложение, вычитание, умножение и деление) над многозначными числами в двоичной системе счисления параллельным способом. Выбор двоичной системы и параллельного способа выполнения операций (над всеми разрядами одновременно) обусловлен тем, что при этом достигаются (при прочих равных условиях) наибольшая скорость вычислений и наилучшее использование оборудования (прежде всего — памяти).

Заметим, что для человека двоичное представление чисел неудобно. Поэтому при любой обработке числовых данных исходные и заключительные данные должны быть представлены в десятичной системе счисления. Тем самым в процесс обработки на основе двоичного процессора включаются дополнительно два процесса перевода чисел из одной системы счисления в другую и обратно. Поэтому, чтобы использование двоичной системы оправдало себя, нужно, чтобы после перевода в двоичную систему над данными производилось достаточно большое число операций.

Во многих системах обработки данных (например, операций с банковскими счетами, начислении зарплаты и др.) количество вычислительных операций для обработки каждой очередной порции данных настолько мало, что переводы из одной системы счисления в другую не оправдывают себя последующей экономией на вычислениях. В этом случае предпочтительнее использовать процессоры, работающие с двоично-десятичными кодами, причем, как правило, доля арифметических операций в общем цикле

обработки данных в указанных применениях настолько мала, что арифметические операции целесообразно выполнять последовательно разряд за разрядом.

Таким образом, мы очертили первое разделение арифметических процессоров на двоичные параллельные и последовательные десятичные (точнее, двоично-десятичные) процессоры. В 60-е годы они назывались соответственно процессорами для научных и деловых расчетов. Сегодня эти названия уже устарели, поскольку в число деловых расчетов включаются сложные планово-экономические расчеты, не уступающие по сложности вычислений самым трудоемким научным задачам.

Двоичные параллельные процессоры в свою очередь делятся на процессоры с *фиксированной* и с *плавающей запятой*. Для двоичных параллельных процессоров с фиксированной запятой, включаемых в состав больших и средних ЭВМ, выбирается достаточно большая разрядность представления чисел (32 двоичных разряда и более). Точность в 32 двоичных разряда примерно соответствует десяти десятичным разрядам. Если эта точность недостаточна, то в ЭВМ включаются режимы работы с двойной точностью, когда процессор работает сначала с одной половиной каждого числа, а затем — с другой (на том же оборудовании). Скорость вычислений при этом значительно уменьшается, причем, к сожалению, не в два, а гораздо большее число раз.

Двоичные процессоры с плавающей запятой используют, как правило, более длинные коды (48 или даже 64 двоичных разряда). Для них также могут быть организованы режимы, использующие двойную точность представления чисел. Последовательные десятичные процессоры в принципе могут работать с числами произвольной длины. Однако на практике точность представления чисел ограничивается некоторой величиной, хотя обычно и достаточно большой.

Поскольку арифметические операции бинарны, то они (включая результат операции) имеют дело с тремя операндами. Поэтому для построения АЛУ арифметических процессоров в принципе достаточно трех регистров, из которых один обычно представляет собой накапливающий сумматор. Однако при вычислениях по сложным формулам промежуточные результаты с целью повышения скорости не выдаются вовне, а сохраняются внутри процессора. Для этой цели (а также и для некоторых других целей) арифметические процессоры средних и больших ЭВМ часто снабжаются дополнительными регистрами. Иногда эти регистры объединяются в стеки. Помимо арифметических операций, в арифметические процессоры встраиваются и другие операции, обеспечивающие их алгоритмическую универсальность.

2.4.2. Процессоры мини-ЭВМ. Помимо больших и средних ЭВМ, большое распространение получили так называемые *мини-*

ЭВМ, или *миникомпьютеры*, предназначенные в первую очередь не для вычислений с большой точностью, а для работы с *реальной измерительной аппаратурой*, поставляющей данные ограниченной точности. Поскольку с этими данными приходится производить немало вычислений, мини-ЭВМ снабжаются обычно параллельными двоичными процессорами небольшой разрядности (обычно 16 разрядов и даже меньше). Вообще, производится максимально возможное упрощение схем, выполняющих арифметические операции, особенно операции с плавающей запятой. Благодаря этому снижаются стоимость и размеры процессора и всей ЭВМ в целом, что делает этот тип ЭВМ наиболее распространенным на практике. Разумеется, подобные процессоры снабжаются универсальным (алгоритмически полным) набором команд, что позволяет, в частности, реализовать с их помощью выполнение поразрядных вычислений в двоично-десятичной арифметике.

2.4.3. Интеллектуальные процессоры. Первые ЭВМ строились в соответствии с так называемыми *принципами фон Неймана*. Одним из этих принципов являлась максимальная простота процессоров и, как следствие, максимальная простота набора операций. В состав таких операций включались обычно лишь простейшие арифметические операции, логическая операция сравнения двоичных кодов и некоторые простые операции управления. Это было необходимо, поскольку элементная база ЭВМ первого поколения, основанная на вакуумных лампах, не могла позволить строить более сложные процессоры. Положение изменилось уже во втором поколении ЭВМ при переходе на полупроводники и при появлении относительно дешевых устройств памяти для хранения микропрограмм.

Еще в 1959 г. автор высказал мысль о необходимости в связи с совершенствованием элементной базы ЭВМ отступать от принципов фон Неймана и, в частности, строить процессоры со сложными наборами операций. Эта идея впервые была реализована в ЭВМ МИР-1, выпускавшейся серийно с 1965 г. В набор операций процессора МИР-1 были введены операции вычисления простейших элементарных функций $\sin x$, $\ln x$ и др., операция $n!$ (факториал) для целых чисел n, операции вычисления (с заданной степенью точности) бесконечных сумм, определенного интеграла и т. п.

В ЭВМ МИР-2, выпускавшейся серийно с 1969 г., в состав операций, выполняемых процессором, были введены тождественные преобразования алгебраических формул, символическое (формульное) дифференцирование и интегрирование и т. п. Память для хранения микропрограмм как в МИР-1, так и в МИР-2 была организована ступенчатым способом, описанным в п. 1.9.2.

Благодаря высокому уровню машинного языка в ЭВМ класса МИР резко облегчалось общение с ними. Вместо «тупого» исполнителя, когда, задавая программу работы, требовалось «растолковывать» подробнейшим образом каждый шаг, у пользователя появился гораздо более «интеллектуальный» помощник, способный понимать его, что называется, с полуслова.

Во второй половине 60-х годов идея интеллектуализации процессоров была воспринята американской фирмой Burroughs и в настоящее время является общепризнанной линией развития вычислительной техники.

Заметим, что идея интеллектуализации процессоров, решая свою основную задачу — упрощение общения пользователя с ЭВМ, способствует в то же время и решению другой важнейшей задачи — увеличения производительности ЭВМ. Действительно, во-первых, использование крупных операций значительно уменьшает внешние обмены процессора, чем существенно повышается общее быстродействие ЭВМ. Во-вторых, при обычной программной (вне процессора) интеллектуализации ЭВМ программист, разрабатывая программы, должен пользоваться уже готовым набором операций, предоставленных ему разработчиком процессора. Если же такие программы переводятся в процессор (на уровень микропрограмм) и разрабатываются *вместе* с процессором, у разработчика появляются дополнительные возможности их оптимизации. Например, вместо выполнения операции деления на 2 при вычислениях какой-либо функции можно просто сдвинуть соответствующее двоичное число на регистре на один разряд вправо.

В результате во многих случаях удается получить значительное увеличение производительности. Так, ЭВМ МИР-2 успешно конкурировала в скорости выполнения ряда аналитических преобразований с большими ЭВМ с программной реализацией таких преобразований, которые превосходили МИР-2 по номинальному быстродействию в сотни раз. Еще один дополнительный эффект интеллектуализации процессора — значительная экономия памяти ЭВМ, необходимой для хранения программ и промежуточных данных.

2.4.4. Магистральные процессоры. При выполнении последовательности команд, поступающих извне, процессор выполняет целый ряд операций, повторяющихся от команды к команде: он должен воспринять извне код очередной операции, адреса, по которым хранятся операнды, и адрес, по которому должен быть заслан результат операции, принять и разместить на регистрах операнды, выполнить операцию, отослать вовне ее результат и подготовиться к восприятию следующей команды (выработать адрес, по которому за ней надо обратиться).

Поскольку все эти составные части процесса выполнения любой команды встроены в одну цепочку, напрашивается мысль об

их выполнении на конвейере. Такой *конвейер* представляет собой ряд соединенных друг с другом в одну цепочку *специализированных процессоров*, каждый из которых выполняет лишь свою часть работы, после чего передает обрабатываемую команду следующему специализированному процессору. Сам же он, освободившись от работы по исполнению своей части команды, получает от предыдущего процессора следующую команду для исполнения соответствующей ее части. Темп выполнения команд при этом существенно ускоряется.

Рассмотренный *конвейерный метод* выполнения команд представляет собой частный случай *магистрального метода*. В отличие от только что рассмотренного простого конвейера, в *магистрали* на отдельные составные части раскладывается также выполнение основной операции. Поскольку такие разложения для разных операций, вообще говоря, различны, в магистральном методе используется обычно несколько конвейерных линий с гибко перестраиваемыми связями между ними.

Особенно большой эффект подобное распараллеливание основной операции дает в случае сложных операций, характерных для интеллектуальных процессоров, причем, чем сложнее операции, тем, как правило, больший эффект может быть получен.

В идеале на магистральном процессоре последовательность команд может обрабатываться в темпе следования отдельных микроопераций или в темпе — один рабочий такт процессора (микротакт) на одну команду (при достаточной скорости внешних обменов).

На практике подобный идеал полностью недостижим, поскольку команды не всегда выполняются в порядке их расположения в программе, однако увеличение быстродействия на один (десятичный) порядок в магистральных процессорах достигается относительно легко. Поэтому магистральный принцип широко применяется практически во всех современных ЭВМ сверхвысокой производительности, рассчитанных на широкий круг применений. Учитывая относительную дороговизну магистральных процессоров, в более дешевых ЭВМ малой, средней и даже высокой производительности принцип магистральной обработки обычно не применяется.

2.4.5. Специализированные процессоры. Если ЭВМ предназначается для работы, в которой часто приходится использовать определенные виды сложных операций, в ее состав включают специализированные процессоры, предназначенные для быстрого выполнения этих операций. Наибольшее распространение получали *векторные* и *матричные процессоры*, в состав которых включается обычно несколько десятков *элементарных процессоров* для одновременного выполнения тех или иных операций над отдельными компонентами вектора или матрицы.

Один из первых матричных процессоров (Иллиак IV) был выпущен фирмой Burroughs. Показывая высокую скорость при обработке векторов и матриц (порядка 200 млн. элементарных скалярных операций в секунду), этот процессор снижает, однако, свое быстродействие почти на два порядка при работе с обыкновенными скалярными величинами. Впрочем, подобный недостаток (сильная зависимость быстродействия от класса решаемых задач) присущ всем специализированным процессорам.

Наряду с векторно-матричными процессорами достаточно широкое распространение получили процессоры, ориентированные на задачи спектрального анализа сигналов (быстрое преобразование Фурье), задачи интерполяции и на ряд других применений.

2.4.6. Периферийные процессоры. Рассмотренные до сих пор типы процессоров применяются в современных ЭВМ в качестве так называемых *центральных процессоров*. Для их разгрузки от более простых операций, прежде всего от операций управления обмена данными оперативной памяти с внешней памятью и устройствами ввода-вывода, широко используются так называемые *периферийные процессоры*, именуемые часто также *каналами* или *канальными процессорами*. Как правило, такие обмены являются *групповыми*, т. е. обмен происходит не элементарными данными, а группами (массивами, записями или файлами). Получив от центрального процессора указание о начальных адресах группового обмена, например, адрес начала записи на магнитной ленте и адрес первой ячейки ОЗУ (оперативной памяти), с которой следует начать записывать в ОЗУ последовательные атрибуты этой записи, периферийный процессор, далее, всю работу по передаче записи в ОЗУ выполняет самостоятельно или, как принято говорить, в *автономном режиме*.

Выполняя передачу, периферийный процессор может изменять *форматы данных*, например, принимать данные побайтово, а передавать побитово или, наоборот, параллельными (многобайтовыми) кодами. В случае, когда такой процессор работает с медленными устройствами, он может передавать или принимать данные по одной физической линии вперемежку от разных устройств. Тогда эти данные должны сопровождаться некоторыми условными кодами (именами устройств). В задачу периферийного процессора, называемого в этом случае *мультиплексором*, входит расшифровка этих кодов при приеме данных и их формирование при передаче данных. Стандарты форматов данных и физических форм представления сигналов, употребляемые при взаимодействии канальных процессоров и подсоединяемой к ним периферии, носят специальное наименование *интерфейсов*.

Канальные процессоры специального вида употребляются также для сопряжения ЭВМ или их вводно-выводных устройств с

телеграфными и телефонными каналами. Более подробно о канальных процессорах говорится в последующих главах.

К числу специальных периферийных процессоров можно отнести и устройства первичной обработки информации, поступающей от различного рода датчиков в современных контрольно-измерительных системах. Поскольку характер и последовательность обработки в разных системах отличаются между собой, для таких процессоров используются специальные *модульно-наборные конструкции*. На стандартных платах монтируются комбинационные схемы и простейшие специализированные процессоры, осуществляющие те или иные требуемые преобразования. На таких же платах монтируются преобразователи формы информации, позволяющие сопрягать дискретную часть системы с аналоговыми датчиками.

Подбирая нужные для обработки платы (модули) и располагая их в требуемом порядке на специальных стандартных конструкциях — сборках, получают требуемый *процессор первичной обработки* (*предпроцессор*). В сборках имеется система шин, соединяющих в единое целое вставляемые в сборку модули (снабженные стандартными разъемами). Для одного из конструктивных стандартов, широко используемых в контрольно-измерительных системах для ядерной физики, системы Камак, применяется связующая система, состоящая из 24 информационных шин.

Заметим, что описанный способ компановки предпроцессоров можно рассматривать как один из частных случаев магистрального принципа обработки. Тем самым удается получать высокие скорости обработки, необходимые для экспериментальных исследований в ядерной физике и в ряде других случаев. В тех же случаях, когда требования к скорости обработки (равно как и к максимальной точности, измеряемой числом информационных шин) не столь высоки, в мировой практике употребляется другой побайтовый стандарт — интерфейс предпроцессоров, предложенный первоначально американской фирмой Hewlett Packard (так называемый *приборный интерфейс* Международной электротехнической комиссии).

2.4.7. Микропроцессоры. Успехи микроэлектроники позволили создать процессоры, полностью размещаемые на одном кристалле. Вначале разрядность таких процессоров была очень низкой (4 двоичных разряда), что фактически не позволяло использовать их как отдельные устройства. По мере увеличения уровня интеграции (сначала до нескольких сотен, а затем и до тысяч вентилей на кристалл) оказалось возможным повысить разрядность однокристальных микропроцессоров до 8, а потом и до 16 двоичных разрядов.

В ряде случаев точность в 16 двоичных разрядов (соответствующих примерно 5 десятичным разрядам) оказывается доста-

точной для построения ЭВМ. Это относится прежде всего к так называемым *управляющим* ЭВМ, обрабатывающим информацию, поступающую с различного рода аналоговых датчиков. Набор операций, хотя и несложных, достаточно разнообразен, чтобы обеспечить эффективные применения таких процессоров на широком круге задач.

2.4.8. Оценка производительности процессоров. Производительность процессоров определяется обычно количеством операций (команд), которые они могут выполнить за одну секунду. Однако время выполнения различных операций может довольно сильно разниться. Поэтому при определении производительности указывают *среднее быстродействие* на *смеси* различных операций, взятых в определенных пропорциях друг к другу.

Если $p_1, \ldots, p_n$ — относительные доли этих операций в смеси, а $t_1, \ldots, t_n$ — время выполнения каждой из этих операций (в секундах), то среднее быстродействие $\mathscr{P}$ вычисляется по формуле

$$\mathscr{P} = \sum_{i=1}^{n} p_i \Big/ \sum_{i=1}^{n} p_i t_i. \qquad (1.1)$$

Одна из наиболее употребительных смесей, основанная на статистике для многих научных задач и называемая *смесью Гибсона* 1, принимает для операций сложения и вычитания с фиксированной запятой $p_1 = 33\%$, с плавающей запятой $p_2 = 7{,}3\%$; для умножения с фиксированной запятой $p_3 = 0{,}6\%$, с плавающей запятой $p_4 = 4{,}0\%$; для деления с фиксированной запятой $p_5 = 0{,}2\%$, с плавающей запятой $p_6 = 1{,}6\%$ *). Доля операций других типов составляет 53,3%.

2.5. Центральные процессоры ЕС ЭВМ

Единая система ЭВМ (ЕС ЭВМ) стран социалистического лагеря, начиная со второй половины 70-х годов, составляет основную техническую базу безбумажной информатики в нашей стране. Разработка ЕС ЭВМ производилась с учетом соглашений и тенденций, принятых и установившихся в мировой вычислительной технике. В состав системы входят ЭВМ различной производительности, начиная от малой (ЕС-1010 и ЕС-1012) и кончая высокой (ЕС-1060 и ЕС-1065). Система непрерывно пополняется новыми усовершенствованными моделями. Модульная структура и высокий уровень совместимости модулей, взятых из разных моделей, позволяют пользователю наращивать число модулей, заменять одни модули другими, более совершенными [61].

*) Время выполнения операций сложения и вычитания у ЭВМ обычно одно и то же.

Малые машины ЕС-1010 и ЕС-1012, выпускаемые Венгерской народной республикой, стоят несколько особняком. Имея небольшое быстродействие (порядка 5—7 тыс. оп./с) и относительно малые размеры, они относятся скорее к классу мини-ЭВМ. Они предназначены в первую очередь для использования в качестве машин-сателлитов для обслуживания удаленных абонентских пунктов старших моделей ЕС ЭВМ.

Все остальные модели ЕС ЭВМ построены в принципе одинаково и отличаются друг от друга главным образом количественными показателями, в частности производительностью *центральных процессоров*.

Термин «центральный» добавляется к процессору потому, что в состав ЕС ЭВМ входят также специализированные периферийные процессоры, которые рассматриваются в следующих разделах книги. Центральный процессор (сокращенно ЦП) ЕС ЭВМ составляется из трех отдельных АЛУ: основного АЛУ, выполняющего операции двоичной арифметики с фиксированной запятой, и двух дополнительных АЛУ — для двоичной арифметики с плавающей запятой и для десятичной арифметики.

Основное АЛУ имеет в своем составе сумматор и 16 32-разрядных двоичных регистров. Оно оперирует с целыми двоичными числами максимум с 31 двоичным разрядом, что соответствует не менее 9 десятичным разрядам. Один двоичный разряд используется для представления знака числа. В набор операций, выполняемых на этом АЛУ, кроме четырех арифметических операций и операций обмена между регистрами и (внешним по отношению к ЦП) оперативным запоминающим устройством, входят еще некоторые логические операции и операции управления. Подробнее о них будет рассказано в гл. V.

Арифметико-логическое устройство для чисел с плавающей запятой, кроме сумматора, располагает четырьмя 64-разрядными двоичными регистрами. Представление числа — либо 32-, либо 64-разрядными двоичными кодами. Первый разряд отводится под знак числа, следующие 7 — под характеристику p, а последние 24 или 56 разрядов — под его мантиссу m. Представляемое таким образом число считается равным ± 0, $m \times 16^{p-64}$. Таким образом могут быть представлены числа в диапазоне (по абсолютной величине) от $0{,}54 \times 10^{-78}$ до $0{,}72 \times 10^{76}$. АЛУ выполняет 4 арифметические операции над такими числами и, разумеется, операции обмена с регистрами.

Арифметико-логическое устройство для десятичных чисел выполняет 4 арифметические операции над десятичными числами, которые могут иметь максимально 32 значащие цифры. Это же устройство обычно используется для преобразований из двоичной системы в десятичную и наоборот. Операции выполняются последовательно, разряд за разрядом. Числа могут иметь пере-

менную длину (не превосходящую, разумеется, 32 разрядов). Устройство позволяет выполнять (побайтово) некоторые операции над произвольными (буквенно-цифровыми) словами переменной длины (не превосходящей 256 символов).

Хотя мы назвали АЛУ для десятичных чисел дополнительным, оно, как правило, включается во все модели, поскольку операции над словами широко применяются в современной безбумажной информатике.

Приведем теперь сравнительные характеристики быстродействия центральных процессоров различных моделей ЕС ЭВМ, сведя их для удобства сравнения в таблицу. Мы указываем в ней среднее время (в микросекундах) выполнения следующих основных операций: сложения (и вычитания) с фиксированной запятой (±ф. з.), сложения (и вычитания) с плавающей запятой (±п. з.), умножения с фиксированной запятой (×ф. з.), умножения с плавающей запятой (×п. з.), деления с фиксированной запятой (:ф. з.) и деления с плавающей запятой (:п. з).

Тип ЭВМ	$\frac{\pm}{\text{ф.з.}}$	$\frac{\pm}{\text{п з.}}$	$\frac{\times}{\text{ф.з.}}$	$\frac{\times}{\text{п.з.}}$	$\frac{:}{\text{ф.з.}}$	$\frac{:}{\text{п.з.}}$
ЕС-1022	3,3—6,0	14—18	23—42	26	65—68	60
ЕС-1033	1,4—2,7	4,5	8,5	9,5	14,6	17,7
ЕС-1040	1,6—1,9	2,8—4,2	6,7—7,6	6—12,1	10,6	7,6—16,1
ЕС-1050	0,65	1,4	2	2—3,2	8,3	7,2—12
ЕС-1060	$\leqslant$0,32	$\leqslant$1,75	$\leqslant$2,00	$\leqslant$2,10	$\leqslant$5,05	$\leqslant$4,40
ЕС-1065	0,140	0,288	0,288	0,576	—	3,152

2.6. Цифровые вычислительные машины

Как уже отмечалось в предыдущей главе, современные ЭВМ представляют собой систему различных устройств, связанных между собой физически и управляемых специальной (программной) *операционной системой*. Наборы устройств и вид операционной системы зависят от *мощности* ЭВМ, а также от ее назначения и режима использования. Следует подчеркнуть, что *мощность* ЭВМ определяется не только производительностью центральных процессоров, которая может быть полностью реализована лишь при надлежащих комплектах и характеристиках всех остальных устройств и, прежде всего, оперативной и внешней памяти.

По мощности принято различать *супер*-ЭВМ, *большие* и *средние* ЭВМ, *мини*-ЭВМ и *микро*-ЭВМ. По назначению — *универсальные* ЭВМ для оборудования вычислительных центров (ВЦ) общего назначения и ЭВМ, входящие в состав так называемых *программно-технических комплексов*, ориентированных на те или

ные *классы применений* (например, для решения планово-экономических задач в масштабе предприятия). По режимам использования различают *пакетный режим*, *режим реального времени* и так называемый *режим разделения времени*, включающий обычно *режим диалога*.

Не вдаваясь пока в характеристику назначений ЭВМ (это будет сделано в последующих главах), рассмотрим более подробно вопрос о мощности и режимах работы ЭВМ.

2.6.1. Супер-ЭВМ. Так принято называть уникальные ЭВМ или скорее *вычислительные системы*, предназначенные для решения особо сложных научно-технических, планово-экономических задач, а также задач обработки информации и управления для особо сложных систем реального времени (например, глобальных систем противосамолетной и противоракетной обороны).

Производительность таких систем исчисляется сегодня десятками (а для специализированных систем сотнями) миллионов операций в секунду (по Гибсону). Соответственно, их стоимость исчисляется многими миллионами рублей (порядка 10 млн. руб. и выше).

Центральные процессоры супер-ЭВМ выполняются на самых быстродействующих (и потому самых дорогих) элементах и используют, как правило, конвейерный принцип обработки команд. Для подпитки процессоров наряду с ОЗУ используется сверхоперативная память (СОЗУ) небольшого объема, но очень большого быстродействия. В систему включаются высокопроизводительные периферийные процессоры. Емкость внешней памяти измеряется обычно многими сотнями миллиардов байтов, а оперативной — десятками миллионов байтов (до 256 Мбайт).

Общее количество информации, которая подвергается обработке в различных частях системы за одну секунду,— так называемая *пропускная способность* — достигает у супер-ЭВМ 20 млрд. байт в секунду и более. Подобная производительность достигается благодаря использованию принципа *многопроцессорной обработки*. Обычно супер-ЭВМ включают в себя несколько (4, 8 и более) универсальных высокопроизводительных центральных процессоров, а кроме того, возможно, и различного рода специализированные процессоры. Все устройства, включаемые в состав супер-ЭВМ, как правило, имеют экстремальные характеристики по всем показателям (скорость, объемы памяти, набор возможностей управления и т. д.).

Супер-ЭВМ имеются сегодня лишь в наиболее технически развитых странах, а их количество исчисляется единицами, в лучшем случае десятками.

Недавно автором был найден новый принцип организации работы универсальных многопроцессорных систем, названный *принципом макроконвейера*. Подобно тому, как уже описанный выше

конвейер осуществляет параллельное выполнение отдельных микроопераций, макроконвейер делает то же самое на уровне целых программных блоков. Специальные средства организации структур данных и *структурирования* памяти, а также гибкая система коммутации позволяют строить *универсальные* многопроцессорные системы со многими десятками, сотнями и даже тысячами процессоров.

2.6.2. Большие и средние ЭВМ. Мы объединяем большие и средние машины в один класс, поскольку сегодня они имеют много общего в *архитектуре* (составе аппаратуры и принципах организации их совместной работы). Такие ЭВМ широко применяются для автоматизации различных информационных процессов: научных и планово-экономических расчетов, проектно-конструкторской работы, испытаний сложных объектов, организационного и технологического управления сложными объектами (цехами, предприятиями, отраслями народного хозяйства) и др.

Производительность больших ЭВМ исчисляется сегодня миллионами, а средних — сотнями тысяч операций в секунду. То же самое имеет место в отношении объемов оперативной памяти: для больших ЭВМ — миллионы, а для средних — сотни тысяч байтов. Соответственно (примерно на порядок) разнятся и объемы внешней памяти (сотни миллионов и даже миллиарды байтов для больших машин, десятки или сотни миллионов — для средних). *Конфигурация* (состав оборудования) ЭВМ может меняться в значительных пределах в зависимости от назначения. В установках со средней производительностью центрального процессора объем внешней памяти и производительность устройств ввода-вывода ЭВМ могут достигать значений, характерных для больших ЭВМ, и наоборот.

К этим двум классам относятся широко распространенные сегодня машины ЕС ЭВМ. Их процессоры уже были описаны выше. Что же касается архитектуры ЕС ЭВМ (характерной для большинства ЭВМ такого же класса), то она представлена на рис. 2.7. Кружками на этом рисунке представлено *периферийное оборудование*, т. е. *устройства ввода-вывода* и *устройства внешней памяти*, а треугольниками — *контроллеры* этих устройств.

Конфигурация периферийного оборудования и каналов может варьироваться в зависимости от назначения машины. Максимально (обычно лишь в старших моделях) в систему может быть включено 8 каналов (из них один мультиплексор). Число же различных периферийных устройств может исчисляться сотнями.

Все обмены между периферийным оборудованием и оперативной памятью выполняются через каналы. Центральный процес-

сор лишь инициирует каждый такой обмен и получает сигнал о его окончании. Кроме того, в ЕС ЭВМ имеются еще так называемые адресные шины *прямого доступа* (показанные на рис. 2.7 штриховыми линиями). Они могут использоваться для подключения источников сигналов, требующих немедленной обработки центральным процессором. Такая ситуация может возникнуть, например, при объединении нескольких ЭВМ в так называемый *многомашинный комплекс*. При этом основные информационные обмены могут осуществляться через каналы с помощью специальных устройств, называемых *адаптерами* (*канал — канал*), как это показано на рис. 2.8.

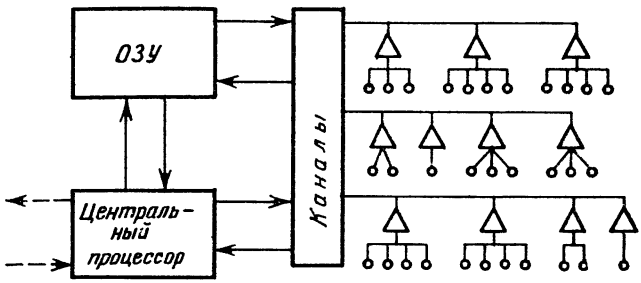

Рис. 2.7

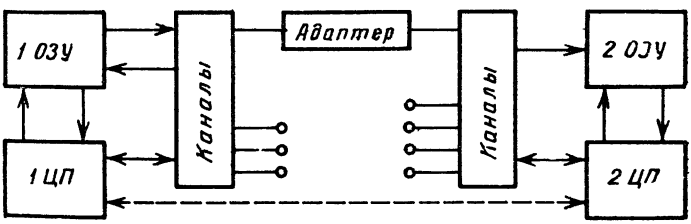

Рис. 2.8

Что же касается шин прямого доступа (на рисунке показаны штрихом), то они используются в основном для передачи *управляющих* сигналов.

2.6.3. Мини-ЭВМ. К классу мини-ЭВМ принято относить малоразрядные (16 двоичных разрядов и менее) машины небольших габаритов и невысокой стоимости. Что же касается скорости выполнения арифметических операций, то у современных мини-ЭВМ она практически (если не считать меньшей разрядности) сравнялась с соответствующим показателем для средних машин. Кроме того, не имея, как правило, специальной скорост-

ной аппаратуры для операций с плавающей запятой, мини-ЭВМ уступают в скорости их выполнения машинам среднего класса, снабженным такой аппаратурой.

Мини-ЭВМ уступают машинам среднего класса и по большинству других показателей (объем оперативной и внешней памяти, конфигурация периферийного оборудования и т. д.). Имея дело с меньшим числом периферийных устройств, мини-ЭВМ, как правило, не имеют специальных канальных процессоров, используя более простые виды *интерфейсов* (сопряжения) с периферией. Если у больших и средних ЭВМ периферийное оборудование (и даже его контроллеры) обычно выполняется в виде отдельных шкафов, мини-ЭВМ, используя специальную малогабаритную периферию, зачастую обходятся одним шкафом (не считая пульта управления и некоторых устройств ввода-вывода).

Мини-ЭВМ находят широкое применение для научно-технических и планово-экономических расчетов невысокой сложности, а также для автоматизации экспериментальных исследований, управления технологическими процессами и др.

Наибольшее распространение в нашей стране (и в целом в социалистическом лагере) получили мини-машины серии СМ ЭВМ.

Архитектура, присущая современным мини-ЭВМ, наиболее ярко выражена в машинах СМ-3 и СМ-4. В основу этой архитектуры положен так называемый принцип *общей шины*. Общая шина включает в себя провода для параллельной передачи адреса, данных и управляющих сигналов. К этой шине подключаются центральные процессоры, оперативное ЗУ и все периферийные устройства. Каждое периферийное устройство, как и ОЗУ, снабжается так называемым *буферным регистром*, который включается в *общее поле памяти* ЭВМ и адресуется точно так же, как и ячейки ОЗУ. Кроме того, у каждого устройства имеется однобитовый регистр — так называемый *флажок*. Когда флажок поднят (в регистр заслана единица) — это означает, что устройство готово к передаче данных. При передаче данных одновременно передается код адреса, который воспринимается всеми устройствами и анализируется специальными схемами. В результате этого анализа данные (из буферного регистра) воспринимаются и передаются только тем устройством, адрес буферного регистра которого был передан по адресным шинам.

В минимальный комплекс машины СМ-3 входят (кроме процессора с быстродействием порядка 200 тыс. оп/с) ОЗУ емкостью 16 К, внешняя память (на магнитных дисках и лентах) емкостью в несколько Мбайт и четыре устройства ввода-вывода.

СМ-4 комплектуется двумя центральными процессорами и соответственно расширенными возможностями оперативной и внешней памяти.

У ЭВМ СМ-1 и СМ-2 интерфейс с периферийным оборудованием осуществляется с помощью специальных согласователей ввода-вывода (упрощенных каналов), к каждому из которых может подсоединяться до 16 периферийных устройств. К центральному процессору подсоединяется до трех таких согласователей. Для подключения удаленной периферии (на расстоянии нескольких километров) используются специальные расширители интерфейсов (см. § 4.9). Интерфейс машин СМ-1 и СМ-2 с периферией носит наименование *интерфейс* 2К.

В комплекс СМ-1 включается один центральный процессор, а в комплекс СМ-2 — 2 процессора. Максимальные объемы ОЗУ у них равны 64 и 256 Кбайт соответственно.

2.6.4. Микро-ЭВМ. Успехи микроэлектроники сделали возможным появление универсальных микро-ЭВМ, на порядок более дешевых и малогабаритных по сравнению с мини-ЭВМ. Они обычно выполняются в настольном варианте, приближаясь по размерам к обычной пишущей машинке. Собственно центральный процессор и ОЗУ (объемом от 4 до 64 К) занимают совсем небольшой объем. Необходимость некоторого увеличения объема вызывается периферийным оборудованием. Пока не разработана специальная микропериферия, микро-ЭВМ пользуются наиболее малогабаритными из периферийных устройств мини-ЭВМ.

Состав и возможности периферийного оборудования у микро-ЭВМ, как правило, значительно меньше, чем у мини-ЭВМ. То же самое обычно имеет место и в отношении набора команд (сохраняющего тем не менее алгоритмическую универсальность). Работая, как и мини-ЭВМ, с короткими кодами, микро-ЭВМ перекрывают диапазон скоростей от нескольких тысяч до нескольких сот тысяч операций в секунду.

Основная область применений микро-ЭВМ — автоматизация экспериментальных исследований и управлений технологическими процессами. В конце 70-х годов большое распространение получила практика встраивания микро-ЭВМ в сложные измерительные приборы и технологическое оборудование. Что же касается микропроцессоров, а также специализированных комбинационных и последовательностных схем, то они находят применение и в оборудовании гораздо меньшей сложности.

С микро-ЭВМ не следует путать *электронные калькуляторы*, которые сегодня выпускаются в «карманном» исполнении (в виде блокнота), встраиваются в часы и т. п. Простые карманные калькуляторы выполняют четыре арифметических действия. Более сложные калькуляторы снабжены операциями вычисления элементарных функций.

2.6.5. Режимы работы ЭВМ. Первые ЭВМ решали одну задачу за одно обращение. В настоящее время широкое распространение получил так называемый *пакетный метод обработки.*

Суть его состоит в том, что в ЭВМ вводится не одна задача (программа), а целый *пакет задач*. Используя то обстоятельство, что в большинстве современных ЭВМ (за исключением самых малых) различного рода обмены могут выполняться одновременно с работой центрального процессора, ЭВМ одновременно работает с несколькими программами (старшие модели ЕС ЭВМ — до 15 программ). Подобный режим, называемый *режимом мультипрограммирования*, обеспечивает лучшую загрузку оборудования и увеличение пропускной способности ЭВМ.

Операционная система (ОС) ЭВМ осуществляет переключения центрального процессора с одной задачи на другую не только по их окончании, но и в том случае, если данные для решаемой задачи еще не переданы из внешней памяти в оперативную. Одновременно с таким *прерыванием* решаемой задачи формируется задание канальным процессорам найти и передать в ОЗУ недостающие данные. В случае их отсутствия и во внешней памяти, ОС формирует задание на их ввод в ЭВМ.

Второй режим работы ЭВМ — *режим реального времени* — уже был в основном описан ранее (1.9.1). Третий режим — *режим разделения времени* — получает все большее и большее распространение, позволяя использовать большие ЭВМ и супер-ЭВМ для одновременного обслуживания большого числа абонентов. С этой целью операционная система выделяет каждому абоненту *элементарные порции времени*, в течение которого ЭВМ решает его задачу. Этот прием особенно эффективен при *диалоговом* или, как его еще иначе называют, *интерактивном режиме* работы с абонентами. Суть его состоит в обмене абонента с ЭВМ различного рода вопросами и ответами.

Пока абонент обдумывает ответ и формирует новый запрос, высокопроизводительная ЭВМ успевает обслужить большое число других абонентов и дать быстрый ответ на вновь поступивший вопрос. В результате у каждого из абонентов создается ощущение, что он один работает с ЭВМ. При достаточно простых задачах абонентов и большой производительности ЭВМ время ее *реакции* на запросы настолько мало, что многие десятки и даже сотни абонентов получают ответы на свои запросы практически одновременно.

Для эффективной работы *системы разделения времени* ее операционная система строится таким образом, что вся память системы (как оперативная, так и внешняя) представляется абонентам как единая (очень большая) *оперативная память*. Получение такой *виртуальной* (воображаемой) памяти происходит за счет того, что операционная система осуществляет все время *быстрый обмен* целыми *страницами данных* между внешней и оперативной памятью, запоминая (в специальном участке ОЗУ) текущее расположение страниц. Пользуясь этой информацией,

операционная система автоматически переводит *виртуальный* (так называемый *математический*) адрес, формируемый абонентом, в *истинный* (*физический*) адрес ОЗУ. Разумеется, такой перевод может быть сделан лишь тогда, когда адресуемая абонентом страница находится в ОЗУ. В случае ее отсутствия там дается задание на ее поиск и передачу в ОЗУ. Таким образом, создается *очередь задач* на обслуживание их нужными страницами данных. Разработаны достаточно простые дисциплины такого обслуживания, осуществляющие приемлемый практически уровень минимизации очереди задач. Существенно при этом, чтобы размер страниц был не очень большим, но и не очень маленьким (обычно от 256 до 2048 байт).

2.7. Аналоговые вычислительные машины

Подобно цифровым машинам, машины, оперирующие с аналоговой информацией, также строятся из некоторого набора элементов относительно небольшого числа типов. Основу для построения таких элементов составляют так называемые *операционные усилители*, способные усиливать электрический ток в широких спектральных пределах (в частности, медленно меняющийся). Одним из самых важных аналоговых элементов является *интегратор*. Электрическое напряжение $u(t)$ на входе интегратора связано с напряжением $v(t)$ на его выходе соотношением $\frac{d}{dt}v(t) = u(t)$ или $v(t) = c + \int_0^t u(t)\,dt$. Еще один тип элементов составляют сумматоры аналоговых сигналов. Кроме того, имеются элементы, позволяющие перемножать два аналоговых сигнала, умножать сигнал на константу и др.

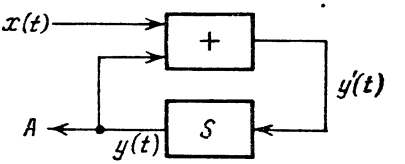

Рис. 2.9

Главное назначение аналоговых машин с таким набором элементов — моделирование решений систем обыкновенных дифференциальных уравнений. Для каждого подобного моделирования нужно по-своему соединить заданные элементы. Например, схема, показанная на рис. 2.9, состоящая из сумматора (+) и интегратора (S), на выходе A вырабатывает сигнал $y(t)$, который представляет собой решение дифференциального уравнения $y' = y + x(t)$.

В течение многих лет развития аналоговой вычислительной техники составление нужных соединений элементов производилось вручную на специальной коммутационной панели. В современных аналоговых машинах используются автоматические

электронные коммутаторы, управляемые обычно универсальными цифровыми машинами. В результате возникает своеобразный *гибрид*, состоящий из цифровой и аналоговой частей. Подобные *гибридные* ЭВМ широко применяются при проектировании автоматических систем управления. При этом аналоговая часть обычно используется для моделирования *объекта управления*, а цифровая — для собственно управляющей системы. Современные аналоговые машины выполняются в микроэлектронном исполнении и имеют поэтому небольшие размеры.

Существуют аналоговые машины для решения дифференциальных уравнений с частными производными и ряда других математических задач. Главными недостатками аналоговой вычислительной техники является, во-первых, малая точность расчетов, во-вторых (и это еще важнее),— отсутствие универсальности, легко достижимой в случае цифровых машин. Эти недостатки делают аналоговые машины в лучшем случае лишь вспомогательным средством в общем арсенале средств безбумажной информатики.

Глава III

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

3.1. Запоминающие устройства с произвольным доступом

Как уже отмечалось в гл. I, *запоминающие устройства с произвольным доступом* представляют собой совокупность *ячеек*, в каждой из которых может храниться двоичное слово определенной длины (одной и той же для всех ячеек). Ячейки нумеруются последовательными целыми числами $n = 0, 1, 2, \ldots$ Номера ячеек принято называть *адресами*.

В ЗУ включается так называемый *регистр адреса* и специальный *коммутатор*, открывающий *доступ* в ячейку, адрес которой помещен в этот регистр. После открытия ячейки с ней можно выполнить одну из двух операций: *запись* или *считывание* информации.

Полный *цикл записи* состоит из следующих операций: 1) прием и запись на адресный регистр ЗУ адреса ячейки, с которой предстоит работать (рабочей ячейки); 2) прием и запись на специальный *регистр операнда* ЗУ (регистр числа) слова, которое нужно записать; 3) собственно запись, происходящая по сигналу «запись», передаваемому на специальный вход ЗУ. При записи старое содержимое рабочей ячейки стирается и вместо него запоминается переданное слово.

Полный *цикл чтения* состоит из следующих операций. Как и прежде, это 1) прием на адресный регистр ЗУ адреса открываемой ячейки; 2) по сигналу «чтение», передаваемому на специальный вход ЗУ, передача из выбранной ячейки ее содержимого в регистр операнда ЗУ; 3) передача выбранного слова в выходной канал ЗУ. При конкретной реализации ЗУ с произвольным доступом во многих случаях оказывается, что в процессе чтения происходит разрушение информации в выбранной ячейке. Поэтому параллельно с последней операцией выдачи из ЗУ выбранного слова осуществляется его повторная запись в выбранную ячейку.

Поскольку повторная запись требует определенного времени, собственно цикл чтения, от момента передачи адреса и сигнала

чтения в ЗУ до момента получения на выходных шинах выбранного операнда, занимает меньше времени, чем *полный цикл обращения* к ЗУ (с учетом повторной записи). Имеются конструкции ЗУ с произвольным доступом, обеспечивающие *неразрушающее считывание информации*. Подчеркнем, что во всех случаях после считывания информации она остается неизменной и готовой для последующих считываний.

3.1.1. Ферритовые ЗУ. В качестве основного элемента для хранения информации в таких ЗУ используются миниатюрные колечки, сделанные из специального магнитного материала — феррита (имеющего петлю гистерезиса, близкую к прямоугольной). Намагниченность колечка в одном направлении отождествляется с записью на нем единицы, а в противоположном направлении — нуля. Колечки монтируются в виде квадратной матрицы, как это показано на рис. 3.1.

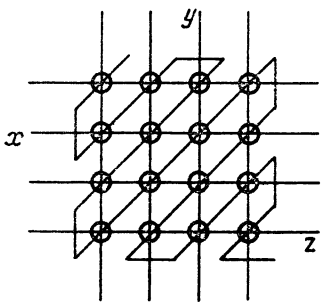

Рис. 3.1

Если выбрать один из горизонтальных (x) и один из вертикальных проводов (y) и пропустить через них ток, то при подходящем выборе величины тока i он способен перемагнитить лишь тот сердечник, который находится на пересечении проводов x и y. Меняя направление тока, можно записать в соответствующий сердечник как нуль, так и единицу. При чтении в сердечник по проводам x и y направляется ток, устанавливающий его в нуль. Если до этого в нем была записана единица, то при перемагничивании сердечника в проводе z наводится электрический импульс. Если же в сердечнике был записан нуль, то такого импульса не возникает.

Набирая пакет из n матриц и производя коммутирование их горизонтальных и вертикальных проводов одновременно, получим *модуль* ЗУ с n-разрядными двоичными ячейками. Разумеется, такой модуль должен, помимо пакета матриц, содержать еще коммутатор, регистры адреса и операнда, а также соответствующие схемы управления.

Если через m обозначить число вертикальных (или горизонтальных) проводов, то в построенном таким образом модуле будет m^2 ячеек. Для удобства адресации m выбирается обычно в виде степени двойки: $m = 2^k$, так что число ячеек в модуле равно 2^{2k}. На практике употребляются модули в 4096 ($k = 6$), 16384 ($k = 7$) и 65336 ($k = 8$) ячеек.

Заметим, что матричный принцип организации доступа к ЗУ облегчает коммутирование: для m^2 ячеек необходимо коммутировать лишь $2m$ групп проводов. В случае необходимости не-

сколько модулей могут объединяться для получения блока ЗУ большего объема. При этом необходим дополнительный коммутатор для коммутирования модулей. Что же касается регистров адреса и операнда, то их достаточно иметь по одному на блок.

Быстродействие ферритового ЗУ будет тем больше, чем меньше размер сердечников. Кроме того, на быстродействие влияет также размер коммутатора, так что с ростом *объема* ЗУ (т. е. числа ячеек) его быстродействие, вообще говоря, уменьшается. В настоящее время в ферритовых ЗУ употребляются ферритовые сердечники, внутренние диаметры которых составляют доли миллиметра. При средних объемах ЗУ (от нескольких десятков до нескольких сотен Кбайт) они обеспечивают время цикла порядка одной микросекунды.

Например, ферритовое ЗУ ЕС-3207, которым комплектуется отечественная ЭВМ средней производительности ЕС-1033, состоит из 8 модулей по 64 Кбайт каждый, что дает общую емкость в 512 Кбайт. Время цикла такого ЗУ 1,2 мкс, а время выборки 0,7 мкс.

Ферритовые ЗУ описанного типа (наиболее распространенного) разрушают информацию при считывании, так что после выборки операнда в цикл встраивается еще операция восстановления содержимого ячейки, из которой происходило чтение. Важным преимуществом ферритовых ЗУ является то обстоятельство, что содержащаяся в них информация сохраняется при временных отключениях электрического питания.

3.1.2. Полупроводниковые ЗУ. Полупроводниковые ЗУ представляют собой по существу некоторое множество обычных триггерных регистров с соответствующими схемами коммутации и управления. В эпоху ЭВМ второго, а тем более первого поколения, когда логические элементы были дороги, подобные регистровые ЗУ могли строиться лишь на небольшие объемы (обычно не более одного-двух десятков слов). То же самое имело место и для периода становления ЭВМ третьего поколения, использовавших интегральные схемы малого уровня интеграции. Лишь в эпоху схем большой интеграции, объединяющих на одном кристалле сотни и даже тысячи логических элементов, полупроводниковые ЗУ стали экономически выгодными.

Преимущество полупроводниковых ЗУ — однородность схемных решений в коммутационно-управляющей и в собственно запоминающей части. В отличие от ферритовых ЗУ, они открывают гораздо бо́льшие возможности для микроминиатюризации, увеличения быстродействия, уменьшения потребления энергии и, главное, для автоматизации их изготовления, а значит, и снижения стоимости. Поэтому, несмотря на свои определенные недостатки, полупроводниковые ЗУ уверенно вытесняют ферритовые,

Недостатки, о которых здесь идет речь,— это, во-первых, потеря информации при временных отключениях питания, а во-вторых, относительная сложность схем коммутации.

Последний недостаток (приводящий к снижению быстродействия) становится особенно ощутимым для ЗУ большого объема. Поэтому такие ЗУ (так называемая *массовая память* объемом в несколько Мбайт) и сегодня пока еще продолжают выполняться на ферритовых сердечниках. Что же касается ЗУ средних и особенно малых объемов (десятки и сотни Кбайт), то полупроводниковые ЗУ по всем показателям, кроме устойчивости к отключениям питания, превосходят ферритовые и уверенно их вытесняют.

3.1.3. Другие типы ЗУ с произвольным доступом. В эпоху ЭВМ первого поколения определенное распространение получили ЗУ на электронно-лучевых трубках. Основой для памяти здесь служит принцип выбивания электронным лучом электронов с поверхности экрана и образования на нем соответствующего поля распределения зарядов — так называемого *потенциального рельефа*. Громоздкость, небольшая емкость, необходимость постоянного поддержания созданного рельефа привели к тому, что этот вид памяти вышел из употребления.

Сегодня появление специальных ячеистых экранов, использование лазеров и голографии позволяют избежать многих недостатков электронно-лучевых ЗУ. Одна из иностранных фирм объявила о выпуске в начале 80-х годов ЗУ на электронно-лучевых трубках с ячеистыми экранами емкостью в 32 Мбайт на одну трубку. Скорость обмена: 4 Мбайт/с при чтении и 1,5 Мбайт/с при записи. Время доступа — менее 0,1 мс, время хранения информации без регенерации — несколько часов. Такие усройства могут составить серьезную конкуренцию дискам с фиксированными головками (см. ниже). Однако пока такие ЗУ получили лишь ограниченное распространение и главным образом там, где требуется так называемая *долговременная память*, т. е. память, в которую информация записывается раз и навсегда, обеспечивая лишь быстрое считывание этой информации.

3.1.4. Оперативное и сверхоперативное запоминающие устройства. Возможность в любой момент считать пли записать информацию в любую ячейку памяти делает ЗУ с произвольным доступом естественным партнером центрального процессора в качестве оперативного хранилища информации. Это привело к закреплению за ними наименования *оперативного запоминающего устройства* (ОЗУ).

При очень большом быстродействии процессора время цикла для обычных ОЗУ (порядка 1 мкс и менее) может являться серьезным ограничивающим фактором для скорости работы системы ОЗУ — процессор.

Одним из выходов из этого положения является организация независимой подпитки процессора от нескольких параллельно работающих блоков ОЗУ. Однако подобное решение приводит к усложнению управления и реализации специальных способов размещения информации по отдельным блокам ОЗУ.

Другой выход заключается в построении *двухуровневой памяти*, когда относительно медленное ОЗУ большого объема подпитывает не процессор, а специальное так называемое *сверхоперативное* ЗУ (СОЗУ), представляющее собой память с произвольным доступом относительно небольшого объема (порядка не более нескольких Кбайт), но существенно более быструю, чем основное ЗУ (время цикла порядка 0,1—0,2 мкс). СОЗУ выполняются сегодня, как правило, на полупроводниковой микроэлектронной основе. Их задача — непосредственная подпитка информацией особо быстродействующих процессоров.

3.1.5. Долговременные запоминающие устройства. ЗУ с произвольным доступом могут строиться так, чтобы информация в них записывалась раз и навсегда и в процессе работы могла только считываться. Подобные ЗУ называются *долговременными* ЗУ (ДЗУ). По конструкции они, как правило, значительно проще, чем ОЗУ, а кроме того, обычно и быстрее.

В недавнем прошлом наиболее распространенными были ферритовые ДЗУ. От уже описанных выше (п. 3.1.1) *двусторонних* ферритовых ЗУ (т. е. с возможностью как считывания, так и записи) они отличаются тем, что информационный провод z (см. рис. 3.1) проходит только через те ферритовые кольца, в которых должна храниться 1, и минует кольца, которые соответствуют нулям.

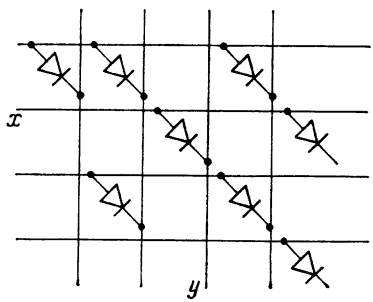

Рис. 3.2

В этом случае упрощение конструкции относительно невелико (исключаются лишь цепи, необходимые для записи информации). В настоящее время ферритовые ЗУ вытесняются *диодно-матричными* ДЗУ в микроэлектронном исполнении. Основу их составляет матрица, образованная двумя перпендикулярными друг к другу системами проводов (рис. 3.2). Если точке пересечения двух таких проводов (x и y) должна соответствовать единица, то эти провода соединяются в ней полупроводниковым диодом, пропускающим сигнал с x на y. В точках, которым соответствуют нули, провода остаются несоединенными.

Подобные диодные ЗУ много проще по конструкции, чем полупроводниковые ОЗУ, и уже сегодня могут содержать несколь-

ко десятков тысяч бит информации на одном кристалле (с перспективой дальнейшего увеличения).

Оказывается возможным строить также *репрограммируемые* ДЗУ, в которых информацию можно перезаписывать, хотя и не с такой скоростью, как это делается в ОЗУ.

3.2. Запоминающие устройства с последовательным доступом

Подобно ЗУ с произвольным доступом, ЗУ с последовательным доступом также разбивается на отдельные ячейки. Отличие между ними состоит в том, что в ЗУ с последовательным доступом невозможен прямой скачок от ячейки с адресом m к ячейке с адресом n. Для выполнения такого перехода нужно пройти все ячейки с промежуточными адресами. Ясно, что подобное свойство ухудшает качество ЗУ с точки зрения пользователя. Однако ЗУ с последовательным доступом гораздо дешевле ЗУ с произвольным доступом, и именно в этом заключается их основное преимущество.

Сегодня ЗУ с последовательным доступом выполняются в основном на магнитных лентах, аналогичных магнитофонным. Для получения такой ленты на пластмассовую основу наносится магнитное покрытие, которое намагничивается в одном направлении. В процессе работы лента протягивается под блоком *записывающих головок*, которые могут изменить направление намагничивания.

Используются два основных метода записи. Первый метод — *без возврата к нулю* (БВН): единицы представлены изменением направления намагничивания, нули — отсутствием такого изменения. При втором методе, называемом *фазовым потенциальным* (ФП), для бита, совпадающего по значению с предыдущим, дважды меняется направление намагничивания. Если же бит отличается по значению от предыдущего, то направление намагничивания меняется только один раз.

Второй метод более сложен в реализации (и несколько дороже). Однако он позволяет быстрее и проще исправлять большинство ошибок, возникающих из-за мелких дефектов ленты. Кроме того, он позволяет осуществить большую плотность записи. Обычно при методе без возврата к нулю плотность записи (для одной головки) составляет 32 бит на мм длины ленты, а при фазовом потенциальном методе — 64 бит на мм.

Наибольшее распространение получили магнитные ленты шириной 1/2 дюйма (1,27 см). Запись информации на таких лентах идет параллельно по 9 дорожкам девятью записывающими головками. Вслед за записывающими головками ставятся 9 *считывающих головок*, которые осуществляют чтение только что за-

писанной информации. На 8 дорожках осуществляется при этом побайтовая запись нужной информации. Девятая дорожка используется для целей контроля записанной информации по так называемому *признаку четности.* Значение *контрольного бита* для каждого 9-битового *кадра записи* выбирается так, чтобы число единиц в кадре было четным. Если при считывании окажется, что это свойство нарушено, то налицо ошибка, которая должна быть исправлена.

Ошибки могут вызываться постоянными дефектами ленты или временными (приставшие пылинки). Если контрольное считывание и проверка четности показывают наличие ошибки, обычно производится попытка повторной записи соответствующего кадра на то же место в расчете на то, что причина ошибки — временный дефект. Если же и повторная запись окажется ошибочной, то дефектное место пропускается и запись осуществляется в новом месте.

Поскольку *лентопротяжный механизм* обладает инерцией, запись и считывание идут обычно не отдельными кадрами, а блоками из многих кадров. *Межблочный промежуток* выбирается в зависимости от скорости движения ленты и инерционных свойств механизма, с тем чтобы на этом промежутке можно было осуществить остановку ленты и ее разгон до полной скорости. Для обычных больших лентопротяжных механизмов длина такого промежутка составляет около 1,5 см (0,6 дюйма). Размеры самих блоков могут варьироваться от некоторого минимального значения (равного обычно 18 кадрам).

В конце каждого блока записывается обычно *контрольный кадр*, который формируется так, чтобы число единиц в каждой дорожке блока было нечетным. Наличие двух видов контроля позволяет в случае наличия в блоке единственной ошибки точно определить ее местоположение (дорожку и кадр) и, следовательно, исправить ее.

Используются и некоторые дополнительные методы контроля правильности записываемой и считываемой с ленты информации.

Для идентификации начала и конца ленты на ее обратной стороне наклеиваются два кусочка алюминиевой фольги, которые идентифицируются фотоэлектрическим способом. Тем самым можно реализовать *обратную перемотку* ленты для ее установки в начальное положение, а также сигнализировать о необходимости остановки ленты при достижении ее конца. Алюминиевые метки, о которых идет речь, наносятся на некотором расстоянии от начала и конца ленты, так что при многократных перемотках от начала к концу и обратно лента не соскакивает с бобин лентопротяжки.

Длина ленты для обычных больших лентопротяжек обычно имеет стандарт 0,75 км. При плотности записи 64 бит/мм это

дает максимальную информационную емкость ленты 48 Мбайт. Истинная информационная емкость из-за наличия дефектных участков и межблочных промежутков будет, конечно, значительно меньше. Максимальная скорость движения ленты в больших лентопротяжках сегодня составляет 5 м/с, но обычно не превышает 2—2,5 м/с.

При плотности записи в 64 бит/мм это означает, что максимальная скорость *информационного обмена* (чтения и записи на ленту) может достигать 320 Кбайт/с.

Используя еще более совершенные методы записи, можно поднять информационную плотность до 250 бит/мм и выше и соответственно повысить скорость обмена. Однако в ЗУ на магнитных лентах, обычно употребляемых на практике, такие плотности не употребляются из-за удорожания аппаратуры и резкого повышения требований к качеству ленты. Поэтому чаще всего довольствуются скоростями информационного обмена от 30 до 180 Кбайт/с. Многие специалисты считают, что повышение плотности записи свыше 250 бит/мм вряд ли будет когда-либо рентабельно.

3.2.1. Ленточные котроллеры. При работе с магнитной лентой, помимо основных операций записи и чтения, требующих взаимодействия с другими устройствами, имеется большое число операций вспомогательного характера, которые могут выполняться под автономным (местным) управлением. К их числу относятся уже упоминавшиеся операции обратной перемотки ленты, фиксации начала и конца. Кроме того, в больших лентопротяжках имеется операция разгрузки, при которой лента полностью перематывается на сменную бобину, после чего эта бобина может быть снята и заменена другой.

Операция захвата ленты второй бобиной после установки сменной бобины в современных лентопротяжках обычно тоже автоматизируется. Операции контроля правильности записи и чтения и повторной записи или чтения блоков, в которых обнаружены ошибки, были описаны ранее.

Поиск необходимого блока или зоны ленты после получения информации о них может также выполняться под автономным управлением.

Все перечисленные операции выполняются специальными электронными *схемами управления*. Определенная часть этих схем конструктивно совмещается с лентопротяжным механизмом, другие могут быть полностью или частично вынесены в отдельные устройства, называемые *ленточными контроллерами*. Обычно такой контроллер обслуживает несколько лентопротяжек (чаще всего от 4 до 8). Одной из главных задач, выполняемых таким контроллером, является задача согласования (синхронизации) скоростей информационных обменов между лентопротяжками и

каналами, по которым информация передается в другие устройства.

3.2.2. Лентопротяжки и контроллеры ЕС ЭВМ. В рамках программы СЭВ разработано несколько типов лентопротяжек, работающих на стандартных бобинах. Внешний диаметр такой бобины 267 мм, максимальная длина ленты 750 м (при стандартной ширине 12,7 мм). Снятая с лентопротяжки бобина для защиты от пыли и механических повреждений помещается в специальный контейнер с внешним диаметром 296 мм.

Лентопротяжка ЕС-5017, разработанная в ГДР, имеет скорость ленты в рабочем режиме 2 м/с, плотность записи информации 8 и 32 бит/мм.

При максимальной плотности записи скорость информационного обмена составляет 64 Кбайт/с. Для перемотки ленты (без записи и считывания) используется повышенная скорость (5 м/с). Благодаря этому полное время перемотки стандартной ленты составляет лишь 2,5 мин.

Разработанная в ЧССР лентопротяжка ЕС-5022 имеет те же параметры, что и ЕС-5017, за исключением скорости движения ленты в рабочем режиме, повышенной до 4 м/с. Тем самым обеспечивается максимальная скорость информационного обмена до 128 Кбайт/с. Имеются и другие, еще более производительные лентопротяжки. Как правило, считывание (в отличие от записи) можно осуществлять на лентопротяжках ЕС ЭВМ при движении ленты как в прямом, так и в обратном направлениях. Применительно к выпускаемым лентопротяжным механизмам, называемым также *ленточными накопителями*, разработаны устройства управления ими (в качестве отдельных устройств): ЕС-5515 (ЧССР), ЕС-5517 (СССР), ЕС-5519 (ПНР) и др. Все они допускают подключение до 8 накопителей на одно устройство и обеспечивают необходимую скорость передачи данных.

3.2.3. Кассетные накопители на магнитных лентах. Такие накопители используются главным образом в составе мини-ЭВМ. Как следует из их названия, главной их отличительной особенностью является использование вместо бобин *кассет*, подобных употребляемым в бытовых магнитофонах. Они отличаются удобством в эксплуатации и небольшими габаритами, благодаря чему обычно встраиваются в один шкаф с другими устройствами мини-ЭВМ. Кассетные накопители отличаются от бобинных меньшей информационной емкостью, меньшей скоростью движения ленты и соответственно меньшей скоростью информационного обмена.

В качестве примера приведем характеристики кассетного накопителя СМ-5300 (НРБ), предназначенного для работы в составе унифицированного ряда мини-ЭВМ: СМ ЭВМ. При плотности записи 32 бит/мм этот накопитель имеет скорость движе-

ния ленты порядка 0,32 м/с, что (при байтовом кадре) обеспечивает скорость информационного обмена порядка 10 Кбайт/с.

Емкость одной кассеты составляет около 10 млн. бит, т. е. немногим более одного Мбайта (против 20 Мбайт и более в бобинных накопителях).

Другое устройство памяти кассетного типа (УВПК) использует более высокую скорость движения ленты. С целью упрощения и удешевления в нем используется однодорожечный способ записи, что приводит, разумеется, к еще большему снижению скорости информационного обмена (до 4 Кбайт/с) и информационной емкости. Поскольку, однако, в мини-ЭВМ требования дешевизны, малогабаритности и простоты превалируют над требованиями производительности, даже такие низкие показатели скорости и емкости во многих случаях оказываются вполне достаточными. В состав УВПК включаются 2 кассетных накопителя, контроллер и автономный блок питания. Он имеет небольшие размеры и вес (до 35 кг).

Заметим, что в состав СМ ЭВМ могут включаться также и обычные бобинные накопители со стандартной лентой ЕС ЭВМ.

3.2.4. Лентотеки. Возможность съема и автономного хранения бобин и кассет с записанной на них информацией приводит к созданию специальных хранилищ такой информации — *лентотек*. Ленты в контейнерах или кассеты размещаются в таких лентотеках в строгом порядке в специальных шкафах, где обеспечиваются необходимые условия защиты от пыли, а также от колебаний температуры и влажности воздуха. Важным элементом защиты является также экранировка лентотеки от магнитных полей, способных испортить хранимую информацию.

Помимо физической защиты обеспечиваются необходимые защитные меры организационно-юридического характера. Сюда входят строгий учет, проверка выдачи, возврата и поступления материалов в лентотеку, персональная ответственность персонала лентотеки за сохранность информации. В большинстве случаев общедоступная информация требует специальной защиты от ее искажения пользователями. С этой целью бобина с такой информацией снабжается специальным ключом (выполняемым обычно в виде кольца), удаление которого блокирует запись на эту бобину. Чтение же при этом остается возможным. Сотрудники лентотеки, владеющие такими ключами, и только они могут (по мере надобности) изменять информацию на таких защищенных бобинах.

3.2.5. Другие виды ЗУ с последовательным доступом. Помимо магнитной ленты, ЗУ с последовательным доступом можно осуществить и на других физических принципах. В качестве примера можно привести, например, *акустические линии задержки*, сыгравшие определенную роль в развитии вычислительной техники.

В них роль дорожки играло проволочное кольцо, по которому распространялись звуковые волны, несущие информацию.

Большие надежды по замене накопителей на магнитной ленте и других электромагнитных устройствах с движущимися частями возлагаются на так называемую *память на магнитных доменах*. Дело в том, что в некоторых магнитных материалах могут создаваться устойчивые миниатюрные области постоянного намагничивания, способные под действием электромагнитного поля перемещаться по поверхности материала, сохраняя заданную намагниченность (величину и направление).

Эти области, называемые *магнитными доменами*, становятся носителями информации, изменяемой и читаемой специальными неподвижными головками, подобно тому, как это делается в обычной магнитной записи. Однако, в отличие от такой записи, под головками движется не материальное тело, обладающее инерцией, а безынерциальное структурированное магнитное поле (в соответствии с записанной информацией). Таким образом, создается как бы безынерциальная лентопротяжка, способная практически мгновенно (с точностью до времени переходных процессов в ускоряющих полях) останавливаться и набирать скорость. Наличие такой возможности резко упрощает управление.

К сожалению, трудности технологического порядка пока не позволяют ЗУ на магнитных доменах конкурировать с устройствами, основанными на традиционной магнитной записи, по объемам запоминаемой информации и по удельной стоимости хранения в них информации (в расчете на один байт). Поэтому пока такие ЗУ используются в ограниченном объеме и главным образом в различного рода специализированных преобразователях информации.

То же самое относится к так называемым *приборам с зарядовыми связями* (ПЗС), где переносящее информацию движение осуществляется не магнитными доменами, а электрическими зарядами.

Ряд иностранных фирм анонсировали выпуск в начале 80-х годов ЗУ на приборах с зарядовой связью емкостью в несколько сот Мбайт со скоростью информационного обмена 75 Мбайт/с и стоимостью примерно 0,005 цента на бит хранимой информации. В них используются большие интегральные схемы памяти с плотностью несколько сот кбит на квадратный сантиметр.

Примерно тех же параметров по плотности и стоимости (и даже лучших) предполагается достичь и в ЗУ на магнитных доменах, хотя скорость информационного обмена у них будет гораздо ниже, чем в ЗУ на ПЗС. Впрочем, скорости обмена зависят не только от физических принципов, на которых строится ЗУ, но и от степени запараллеливания доступа (одновременно ко многим «дорожкам»).

3.3. Запоминающие устройства с прямым доступом

Главный недостаток ЗУ с последовательным доступом — длительное время ожидания подхода под головки нужной ячейки памяти. Даже при условии убыстренной перемотки это время в неблагоприятных случаях измеряется многими десятками секунд, что совершенно не соответствует огромным скоростям работы электронных схем. Вместе с тем перспективы создания дешевых ЗУ с произвольным доступом большого объема (на десятки Мбайт) пока еще достаточно отдаленны.

Поэтому возникает необходимость в относительно дешевых ЗУ большого объема со способом доступа, промежуточным между произвольным и последовательным. При таком способе, называемом *прямым*, в одном устройстве как бы объединяется большое число ЗУ с последовательным доступом, каждое из которых имеет относительно небольшой объем и потому относительно небольшое время доступа. Среднее время доступа в таких устройствах при объемах, сравнимых с объемами ЗУ на магнитных лентах, может быть сокращено в сравнении с ними на 3 порядка и даже более. Рассмотрим различные типы устройств, реализующих идею прямого доступа.

3.3.1. Магнитные барабаны. *Магнитный барабан* представляет собой вращающийся цилиндр с магнитным покрытием, с размещенными вдоль его оси записывающими и считывающими головками. Каждая головка (являющаяся одновременно как записывающей, так и считывающей) работает на свою дорожку. Выбор нужной дорожки (пары головок) производится с помощью быстродействующего коммутатора. После выбора нужной дорожки максимальное время ожидания подхода под головку нужного места дорожки равно времени оборота барабана вокруг своей оси. Даже для барабанов больших размеров скорость в 3000 об/мин не является чрезмерно высокой. Для барабанов меньших размеров она повышается до 6000 об/мин и более. При 3000 об/мин барабан совершает один оборот за 0,02 с. Таким образом, максимальное время доступа к информации на барабане будет 0,02 с, а среднее время — 0,01 с (временем коммутации головок в этом расчете можно пренебречь).

Поскольку барабан вращается с постоянной скоростью, инерционные эффекты в процессе его работы не играют роли (за исключением моментов начального запуска и выключения). Поиск нужного адреса (места на дорожке) производится с помощью специальной дорожки, на которой записаны синхронизирующие импульсы. Считывающая головка считывает эти импульсы и направляет их на *циклический счетчик*, показание которого становится нулевым после завершения полного оборота. Каждому показанию счетчика соответствует определенный угол пово-

рота барабана и, следовательно, определенный адрес на дорожке.

Объединяя дорожки (вернее, соответствующие головки), можно осуществлять *параллельную* запись и считывание отдельных байтов и слов (по разряду на каждую дорожку).

Объем памяти, реализуемый на больших магнитных барабанах, колеблется в пределах от нескольких сот кбайт до нескольких Мбайт. Удельная стоимость хранения информации достаточно высока. Если анонсированные показатели ЗУ на магнитных доменах и ПЗС будут достигнуты, то магнитным барабанам конкурировать с ними будет уже невозможно. Впрочем, и сейчас магнитные барабаны практически уступили свое место магнитным дискам с фиксированными головками.

3.3.2. Магнитные диски с фиксированными головками. Недостатком магнитных барабанов является сравнительно малая поверхность, приходящаяся на единицу объема. Гораздо лучшими показателями с этой точки зрения обладают *магнитные диски*, особенно, если треки (дорожки для записи и считывания информации) располагаются на обеих поверхностях диска (1 трек = = 3625 байт). В дисках с фиксированными головками, как и в случае барабанов, каждый трек имеет собственную (неподвижную) головку для записи и считывания информации.

На мировом рынке имеются однодисковые ЗУ с фиксированными головками емкостью свыше одного мегабайта. Бóльшая емкость достигается за счет того, что на один шпиндель насаживается несколько дисков, вращающихся одновременно. Как и в случае барабанов, переключение трактов выполняется быстродействующим коммутатором, так что время доступа определяется практически лишь скоростью вращения шпинделя.

Как и в случае магнитных барабанов, за ускорение доступа приходится платить увеличением удельной (в расчете на 1 бит) стоимости хранения информации. Эта стоимость не менее чем на два порядка превосходит удельную стоимость хранения информации на магнитных лентах даже при условии, что лента постоянно закреплена на лентопротяжке (одна лента на одно устройство). С учетом возможности смены лент (много лент на устройство) разрыв в стоимости будет существенно бóльшим.

3.3.3. Магнитные диски с подвижными головками. Высокие удельные стоимости хранения информации на барабанах и дисках с фиксированными головками обуславливаются прежде всего наличием большого числа головок и сложностью их коммутирования. Всего этого можно избежать, используя *блок подвижных головок*, как это изображено на рис. 3.3. Головки блока специальным шаговым механизмом (гидравлическим или электродинамическим) устанавливаются с большой точностью (до 0,4—0,8 мк) на ряд находящихся друг под другом дорожек. Эта груп-

па дорожек составляет так называемый *цилиндр*. Переключение цилиндров осуществляется специальным адресным устройством, управляющим шаговым механизмом, так что коммутировать приходится лишь ограниченное число пар головок, равное числу используемых (рабочих) поверхностей *пакета дисков*.

Блок головок может быть полностью выведен из пакета дисков. Это позволяет строить ЗУ со *сменными пакетами дисков*, быстро снимаемыми и вновь устанавливаемыми на шпиндель *дисковода*. Подобно бобине с лентой, сменный пакет дисков помещается в специальный контейнер и в таком виде может транспортироваться и храниться в дискотеках, оборудованных точно так же, как и описанные выше лентотеки.

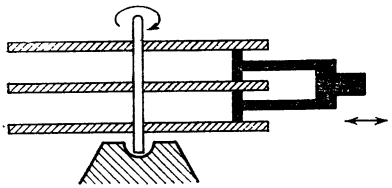

Рис. 3.3

В мировой практике в настоящее время известны ЗУ на сменных магнитных дисках емкостью до 400 Мбайт на шпиндель. Скорость информационного обмена у таких устройств существенно превосходит соответствующий показатель для ЗУ на магнитных лентах, достигая значений в несколько мегабайт в секунду.

Плотность записи на дисках превышает в среднем плотность записи на ленту. Принцип записи — также другой. Линейная скорость движения магнитного слоя под головками в случае дисков измеряется десятками метров в секунду, что на порядок превосходит соответствующий показатель у лучших ЗУ на магнитных лентах.

Время переключения цилиндров (дорожек) в ЗУ на сменных магнитных дисках имеет порядок 0,1 с. Тем самым обеспечивается уменьшение среднего времени доступа по сравнению с ЗУ на магнитных лентах не менее чем на три порядка. Правда, удельная стоимость хранения информации у дисковых ЗУ с подвижными головками на порядок больше (а у дисковых ЗУ с фиксированными головками — на два порядка), чем соответствующий показатель у ленточных ЗУ. Стоимость пакета дисков в 10—12 раз выше стоимости бобины с магнитной лентой. Две внешние поверхности пакета дисков, которые наиболее подвержены повреждениям, как правило, не используются для записи информации. Поэтому число m рабочих поверхностей при n дисках выражается формулой $m = 2n - 2$.

В отличие от магнитных лент, в ЗУ на магнитных дисках обычно избегают параллельной записи информации на несколько дорожек. Все разряды каждого байта и каждого слова располагаются вдоль дорожки один за другим. Правда, в связи с задачей дальнейшего уплотнения записи информации и соответствен-

ного увеличения числа дорожек на дисках, в последнее время начали использоваться так называемые *интегральные головки*. Они представляют собой блок микроминиатюрных головок с соответствующими электронными схемами коммутации. Тем самым в принципе достигается возможность параллельного считывания с группы рядом расположенных дорожек. Сегодня это число достигает пятнадцати, но по мере совершенствования интегральных головок может быть увеличено. Впрочем, принятые ныне формы управления накопителями на магнитных дисках заставляют использовать даже интегральные головки лишь в режиме последовательного доступа к обслуживаемым ими дорожкам.

3.3.4. Дисковые ЗУ ЕС ЭВМ. Для ЕС ЭВМ разработаны дисковые ЗУ со сменными дисками емкостью 7,25, 29, 100 и 200 Мбайт. Наибольшее распространение пока получили дисковые накопители емкостью в 7,25 Мбайт, производимые СССР и НРБ. Советский дисковый накопитель ЕС-5056 имеет 6 дисков, 10 рабочих поверхностей. На каждой поверхности имеются 200 рабочих дорожек и 3 запасные. Они нумеруются от 0 у наружного края диска до 202. На каждую дорожку записывается 3625 байт информации, что при общем числе дорожек, равном 2000, дает информационную емкость 7,25 Мбайт. Скорость вращения диска 2400 оборотов в минуту (40 об/с), чем обеспечивается скорость информационного обмена 156 Кбайт/с (с одной дорожкой).

Минимальное время поиска цилиндра 0,025 с, максимальное 0,15 с. Среднее время доступа к информации 0,09 с (с учетом поиска на выбранной дорожке). Время смены пакета дисков — одна минута.

Накопитель на сменных магнитных дисках ЕС-5061 (НРБ) имеет емкость 29 Мбайт. Четырехкратное увеличение емкости достигнуто за счет удвоения числа рабочих поверхностей и удвоения плотности записи. Количество дорожек на поверхности и скорость вращения шпинделя остаются теми же, что и для ЕС-5056. Времена доступа и поиска цилиндра также имеют примерно те же значения. Максимальная плотность записи (на внутренних дорожках) достигает 90 бит/мм. Скорость информационного обмена с одной дорожкой равняется 312 Кбайт/с.

В накопителях емкостью в 100 Мбайт плотность записи и скорость информационного обмена увеличиваются по сравнению с 29-мегабайтными дисками почти в два раза. Кроме того, удваивается число дорожек на каждой рабочей поверхности. Накопитель на магнитных дисках емкостью 200 Мбайт разработан в НРБ. Его код в системе ЕС ЭВМ есть ЕС-5067, а код соответствующего сменного 200-мегабайтного пакета — ЕС-5267. В пакете 12 дисков с 19 рабочими поверхностями (одна поверхность

используется для управления сервоприводом блока головок, остальные для отладочных целей). В качестве способа записи использована модифицированная частотная модуляция с максимальной плотностью 159 бит/мм. На каждой рабочей поверхности имеются 808 основных и 7 запасных дорожек (минимальный шаг между дорожками менее 70 мк). Минимальное время переключения цилиндров (дорожек) — 0,01 с, максимальное — 0,055 с. Скорость вращения — 3600 об/мин, скорость информационного обмена — 200 Кбайт/с.

Как и в случае магнитных лент, многие операции с накопителями на магнитных дисках могут выполняться под автономным управлением. Сюда относится поиск дорожки и нужного байта на дорожке, контроль информации и др. Кроме того, обмен других устройств с ЗУ на лентах и дисках осуществляется обычно побайтно (т. е. параллельно 8—9 разрядами), тогда как считывание и запись на диск ведется побитово.

Для выполнения сопряжения накопителей на магнитных дисках с другими устройствами и управления автономными операциями служит *устройство управления магнитными дисками.* В ЕС ЭВМ к каждому такому устройству можно подключать до 8 накопителей.

3.3.5. Мини-диски. Для мини-ЭВМ используются обычно упрощенные дисковые накопители, имеющие один диск (2 поверхности) на шпиндель. Накопители могут иметь как сменяемые, так и несменяемые диски. Так, накопитель СМ-5402, предназначенный для работы в составе СМ ЭВМ, включает два диска: сменяемый (помещенный в удобную кассету) и фиксированный. На каждой из четырех поверхностей этих двух дисков размещается по 200 дорожек (плюс 4 запасные). На каждом диске размещается около 2,5 Мбайт (6144 байт на дорожке). За счет достаточно высокой плотности записи (87 бит/мм) скорость *передачи данных* (информационного обмена) составляет 300 Кбайт/с. Достаточно высокими являются и скорости *позиционирования* подвижного блока головок: минимальное время (при переходе на соседний цилиндр) равно 0,01 с, а максимальное — 0,08 с. Скорость вращения диска 2400 об/мин.

3.3.6. Гибкие диски. Особенно удобными для пользования являются минидисковые накопители на *гибких магнитных дисках*, называемых иногда также *дискетами.* По внешнему виду такой диск похож на обычную гибкую грампластинку. Помещаемый в специальный конверт из плотной бумаги (с отверстиями для вала дисковода и блока головок), он весьма удобен для транспортировки и хранения. О технических характеристиках накопителей на гибких магнитных дисках см. в [61].

3.3.7. Жесткие диски. В последние годы наряду с гибкими дисками появились их аналоги, выполненные по так называемой

винчестерской технологии. Диаметр таких дисков 200 мм, а емкость — до 20 Мбайт. Подобная емкость достигается за счет большой плотности записи (2400 бит/мм). Дисковод для такого диска приближается по размерам к пишущей машинке. При емкости более 10 Мбайт жесткие диски дешевле гибких дисков равной емкости.

3.3.8. Архивные ЗУ кассетного типа. Такие ЗУ стали выпускаться в последние годы иностранными фирмами. Информационными модулями в них являются цилиндрические кассеты, содержащие внутри бобину с широкой (порядка 7 см) магнитной лентой небольшой длины (порядка 20 м). Благодаря высокой плотности записи (250 бит/мм) и большому числу дорожек информационная емкость каждого такого модуля достаточно велика: 8 Мбайт на кассету.

Благодаря малой длине ленты время доступа к данным сокращается до нескольких секунд. Очень важно, что ЗУ такого типа включают в себя *автоматизированную лентотеку*. В такой лентотеке кассеты помещаются в автоматически (с помощью ЭВМ) адресуемые гнезда, соединенные пневмопочтой с одной или несколькими лентопротяжками. Выбранная кассета автоматически передается и устанавливается на соответствующую лентопротяжку, а находившаяся на ней ранее кассета (также автоматически) возвращается в свое гнездо.

Автоматизированная лентотека, выпускаемая американской фирмой Control Data, содержит 2000 гнезд, обеспечивая тем самым суммарную информационную емкость в 16 млрд. байт. Имеются сообщения об автоматизированных лентотеках емкостью в 500 млрд. байт, что соответствует примерно полумиллиону 500-страничных бумажных томов среднего формата, т. е. хорошей библиотеке.

Описанные ЗУ занимают как бы промежуточное положение между устройствами памяти последовательного и прямого доступа.

В качестве постоянной (долговременной) архивной памяти используются оптические ЗУ большого объема. В одном из выпускаемых в США устройств такого рода информация представляется в виде выжигаемых лазерным лучом микронных отверстий в полистироловых металлических полосках, наносимых на поверхность стеклянных дисков и считываемых фотоэлектрическим путем. На каждом диске имеются 10 таких полосок, а 8 дисков набираются в пакет, как это имеет место и в случае обычных магнитных дисков.

Благодаря очень большой плотности записи максимальная емкость пакета очень велика — порядка 16 млрд. байт. Время выборки имеет тот же порядок, что и для магнитных дисков (0,1 с). Удельная стоимость хранения информации в таких уст-

ройствах фантастически низка (порядка трех миллионных долей цента на бит) в сравнении с хранением обычной бумажной информации.

Одно время определенное развитие получили архивные накопители на *магнитных картах*. По существу, они могут рассматриваться как ленточные архивные ЗУ с очень короткими лентами. Их преимуществом является практически прямой доступ к информации, после того как нужная карта извлечена из архива и помещена (с помощью вакуумного присоса) на вращающийся барабан (под блоком магнитных головок). Однако это преимущество фактически утрачивается в результате гораздо большего времени, нужного для извлечения требуемой карты из архива.

С практической точки зрения более целесообразно, чтобы времена доступа в архив и к информации на уже извлеченном из архива носителе имели один и тот же порядок, что и имеет место в архивных ЗУ кассетного типа.

3.3.9. Обращение к внешней памяти. Электромеханические устройства памяти (магнитные барабаны, диски и ленты) включаются в состав ЭВМ в качестве так называемой *внешней памяти*. Главное назначение этого вида памяти — осуществлять непрерывную подпитку оперативного ЗУ данными, подлежащими обработке, и своевременно освобождать его от уже обработанных данных.

В современных ЭВМ передача данных между внешними ЗУ и ОЗУ осуществляется обычно через специальные упрощенные процессоры, называемые *каналами* или иногда *периферийными процессорами*.

Получая от центрального процессора программу канала, канал выполняет ее в автономном режиме (независимо от центрального процессора). В его задачу входит прием и проверка информации от выбранного внешнего ЗУ, формирование ее в соответствии с размерами ячеек ОЗУ и фактическая ее передача в последовательные ячейки ОЗУ. При записи информации во внешнее ЗУ операции передачи и формирования данных выполняются в обратной последовательности. В назначение канала входит, кроме того, обмен управляющей информацией с контроллерами внешней памяти и отработка их обращений в центральный процессор.

Поскольку устройства внешней памяти допускают достаточно быстрый обмен данными, на время передачи данных канал полностью занимается одним устройством внешней памяти. Такой режим использования канала называется *монопольным*, а каналы, работающие в таком режиме,— *селекторными каналами*. Стандартный способ представления сигналов в канале и управления ими принято называть *интерфейсом*.

3.3.10. ЗУ с другими видами доступа. Помимо ЗУ с произвольным, последовательным и прямым доступом, в современных ЭВМ применяется память еще с двумя видами доступа: *ассоциативным* и *стековым*.

Принципы работы ассоциативных и стековых ЗУ были изложены в гл. I, и здесь мы их повторять не будем. Отметим лишь, что к началу 80-х годов успехи микроэлектроники полупроводниковых ЗУ позволили получать блоки ассоциативной памяти и стеки, выполненные в виде отдельных чипов.

3.3.11. Место ЗУ различных типов в безбумажной информатике. При построении ЭВМ и автоматизированных систем обработки данных ЗУ различных типов образуют естественную иерархию по мере роста их объема и увеличения времени доступа: СОЗУ, ОЗУ, массовая (промежуточная) память, ЗУ на магнитных барабанах и магнитных дисках с фиксированными головками, ЗУ на сменных магнитных дисках с подвижными головками, ЗУ на магнитных лентах (включая архивные ЗУ). Гибкие диски в больших системах используются главным образом как средства ввода-вывода информации и средства для обмена информацией между различными системами.

Плотность расположения информации на поверхности стандартных магнитных лент колеблется в пределах 125—500 байт/см^2, что почти на 2 порядка больше, чем для обычных бумажных документов. Информационная (поверхностная) плотность для дисковых ЗУ и современной полупроводниковой памяти еще выше, причем для полупроводниковых ЗУ предел еще далеко не достигнут. Оптические архивные ЗУ и устройства памяти на магнитных доменах могут обеспечить плотности порядка 100 000 байт/см^2. Почти таких же показателей ожидают достигнуть в ближайшем будущем также в ЗУ на приборах с зарядовыми связями. Показатели объемной плотности для конструктивно законченных ЗУ, конечно, заметно хуже, но и они начинают обгонять соответствующие показатели для бумажных документов.

Удельная стоимость хранения информации для ЗУ всех типов, кроме магнитных лент и особенно архивной оптической памяти, пока еще заметно больше, чем в случае обычных бумажных документов. Для стандартных бобин магнитной ленты удельная стоимость хранения информации в 1980 г. по мировым ценам была примерно та же, что и у бумажных документов, а у архивных оптических ЗУ — существенно меньше.

Решающее значение, однако, имеет то обстоятельство, что, в отличие от обычных бумажных архивов, информация, хранящаяся в ЗУ любого из перечисленных типов, несравненно динамичнее, поскольку она доступна для ЭВМ, а следовательно, и для быстрых машинных методов обработки. Насколько

важно преимущество динамичности информации, видно хотя бы из живучести такого носителя информации, как *перфокарты* (см. гл. IV). Хотя показатели удельной стоимости и удельной плотности хранения информации на перфокартах в десятки раз хуже по сравнению с обычным бумажным представлением, динамичность представления информации позволяла перфокартам в системах организационного управления не только успешно конкурировать, но и вытесять обычные бумажные документы.

Не следует забывать также, что безбумажная информатика находится в самом начале своего развития и имеет большие перспективы дальнейшего снижения стоимости хранения и повышения оперативности обработки информации, чего нельзя сказать об информатике бумажной.

Если смотреть лишь на критерий удельной стоимости (забыв пока об оперативности), то наилучшие показатели сегодня имеет способ хранения документов в виде сильно уменьшенных фотокопий — *микрофильмов* и *микрофишей*.

В обоих случаях носителем информации служит специальная фотопленка (мелкозернистая или беззернистая). В случае микрофильмов эта пленка вытянута в длину (при малой ширине), в случае микрофишей имеет форму, более близкую к квадрату (чаще всего 9×12 см). В настоящее время микрофильмирование обеспечивает сокращение линейных размеров на 2 порядка и более. При такой плотности на одном микрофише размера 9×12 см можно уместить порядка 10 тыс. страниц текста или примерно 20 Мбайт информации.

Имеется возможность и дальнейшего увеличения плотности, не менее чем на порядок, за счет перехода от оптических к электронно-лучевым методам записи и применения специального (беззернистого) светочувствительного материала. Такая технология позволяет уже сегодня разместить фотокопии всех страниц Большой советской энциклопедии на площади одной стороны спичечного коробка. Правда, при таком уменьшении размеров сильно усложняется и удорожается считывающая и особенно записывающая аппаратура. Поэтому уменьшение линейных размеров в 100 раз (и соответственно площади в 10 тысяч раз) дает сегодня наиболее удовлетворительное решение проблемы.

Архивы на микрофишах имеют размеры, в тысячи раз меньшие размеров соответствующих бумажных архивов, и, кроме того, допускают относительно простую и недорогую автоматизацию доступа.

Автоматизация позволяет путем набора на специальном пульте номера микрофиши и страницы на ней обеспечить быстрый поиск этой страницы и ее высвечивание на экране специального проектора в виде, увеличенном до нормальных размеров. В случае необходимости (с помощью нажима на специальную кнопку)

может быть получена бумажная копия высвеченной страницы (обычных размеров). Более сложна задача поиска в архиве документа по содержанию. Ее решение требует введения в контур управления архивом ЭВМ. Более подробно этот вопрос трактуется ниже (в гл. VI).

Ввиду особой дешевизны и компактности архивов на микрофишах в современных системах обработки данных, оперирующих с большими массивами, находит широкое применение *двухконтурный способ* организации архивов. Полные тексты документов находятся во втором контуре в виде микрофишей. Краткое же содержание и необходимые для поиска признаки этих документов помещаются в первый контур, использующий магнитные диски и магнитные ленты.

Глава IV

УСТРОЙСТВА ВВОДА, ВЫВОДА И ПОДГОТОВКИ ДАННЫХ

Устройства ввода и вывода предназначены, как это видно из их названия, для ввода и вывода информации в ЭВМ. В ряде случаев такой ввод осуществляется через *промежуточные носители информации.* Для нанесения информации на эти носители используют автономные устройства, называемые *устройствами подготовки данных.* Мы разберем вначале устройства ввода и вывода информации с помощью промежуточных носителей, классифицируя их по видам таких носителей.

4.1. Перфокарты

Стандартная *перфокарта* представляет собой прямоугольник из плотного тонкого картона (толщиной 0,18 мм) размером примерно 19 на 8. Она разбивается на 80 колонок, расположенных поперек карты. В каждой колонке имеются 12 позиций для пробивания в них прямоугольных отверстий с помощью специального устройства.

Комбинация одного или двух отверстий представляет десятичные цифры, латинские и русские буквы и другие символы стандартного базисного алфавита (см. § 1.2), так что всего в 80 колонках могут быть представлены 80 символов базисного алфавита. Поскольку каждый такой символ кодируется внутри ЭВМ 8-разрядным двоичным кодом, то информационная емкость перфокарты весьма низка — всего 80 байт на карту или 0,5 байт/см^2. Это примерно на порядок меньше, чем у обычных (машинописных) бумажных документов (5—6 байт/см^2). Поскольку перфокарта толще бумажного листа, объемная плотность записи информации на перфокартах в несколько десятков раз меньше по сравнению с обычными бумажными документами (а удельная стоимость соответственно больше). В ценах 1980 г. цена перфокарт для хранения 1 млн. байт информации составляет более 10 р., а цена бумаги для хранения того же объема информации 40—50 к.

Живучесть перфокарт, несмотря на столь плохие информационные характеристики, объясняется главным образом историческими причинами. Дело в том, что перфокарты и работающая с ними простейшая счетно-сортировочная техника появились еще в конце прошлого столетия, задолго до изобретения ЭВМ. До появления ЭВМ перфокарты были практически единственным способом представления информации, который с помощью специальной *электромеханической сортировки* позволял осуществлять автоматизированный поиск и группировку по признакам в больших массивах информации. Дополняя сортировку простейшими электромеханическими вычислителями — так называемыми *табуляторами*, а также устройствами для пробивки и сверки перфокарт, получаем комплект так называемых *счетно-аналитических машин*.

На протяжении более полувека такие машины служили фактически единственным средством для автоматизации бухгалтерского учета, обработки переписей и других видов относительно несложной обработки больших массивов информации. Накопленный здесь опыт и навыки, с одной стороны, и наличие готового перфорационного оборудования — с другой, привели к широкому использованию перфокарт для ввода информации в ЭВМ. К этому следует добавить наглядность представления информации на перфокартах и возможность простого визуального ее контроля. Поэтому, хотя по современным меркам перфокарты уже безнадежно устарели, они продолжают пока еще достаточно широко использоваться.

Как уже отмечалось в предыдущей главе, живучесть перфокарт является убедительным свидетельством тех преимуществ, которые дает представление информации в *машинно-ориентированной форме*. Несмотря на гораздо более плохие информационные характеристики по сравнению с ориентированной на человека бумажной формой документов, перфокарты вот уже почти столетие успешно конкурируют с ними благодаря возможности автоматизации обработки информации и динамизму информации.

4.1.1. Устройства подготовки данных на перфокартах. Комплект устройств подготовки данных на перфокартах включает в себя прежде всего *клавишный перфоратор*, набор клавиш которого соответствует базисному (96-буквенному) алфавиту (или одному из его сокращенных вариантов). Нажатием на соответствующие клавиши оператор осуществляет пробивку отверстий в колонках перфокарты, автоматически передвигающейся слева направо. В одних перфораторах такая пробивка осуществляется сразу после нажатия на клавишу, в других происходит запоминание набранной информации и одновременная быстрая пробивка всей перфокарты. Во многих современных перфораторах

пробитая информация автоматически выпечатывается вверху перфокарты в расшифрованном (т. е. в обычном буквенно-цифровом) виде. Примером такого *расшифровывающего перфоратора* может служить устройство ЕС-9011. Имеются и отдельные расшифровщики уже пробитых карт, например устройство ЕС-9014.

Второе устройство комплекта — *контрольник*. Он устроен так же, как и перфоратор, но не пробивает отверстий, а сравнивает отверстия на уже пробитой перфокарте с теми, которые соответствуют нажатой клавише. Задача контрольника — фиксировать все несовпадения при вторичном наборе и тем самым осуществлять контроль правильности пробивок. Контрольники находили широкое применение в эпоху счетно-аналитических машин и на ранних стадиях развития ЭВМ. В настоящее время в связи с развитием других средств контроля их употребление резко сократилось.

Скорость набивки перфокарт зависит от навыка работы на клавиатуре и во всяком случае не превосходит скорости печати на пишущей машинке. Поэтому подготовка данных на перфокартах представляет собой весьма трудоемкую операцию, требующую больших затрат ручного труда.

4.1.2. Устройство ввода с перфокарт. В состав устройства входят, во-первых, два *кармана* для колод перфокарт. В *подающий карман* колода перфокарт закладывается человеком, в *приемном кармане* карты накапливаются автоматически после прочтения их устройством. После завершения считывания колода удаляется из приемного кармана человеком.

Из подающего кармана перфокарты автоматически, одна за другой, поступают в *считывающее устройство*, которое сейчас строится в основном на фотоэлектрическом принципе. Специальный электронный *блок сопряжения* преобразует получаемые со считывателя сигналы в стандартный код внутримашинного представления базисного алфавита и выдает его в канал ЭВМ (вместе с дополнительной управляющей информацией).

Скорость ввода информации с перфокарт обычно колеблется в пределах от 500 до 1200 карт/мин, т. е. примерно от 700 до 1600 байт/с. В качестве примера приведем показатели перфокарточного устройства ввода ЕС-6012 (СССР): скорость ввода — 500 карт/мин, емкость подающего кармана 1000 карт, емкость приемного канала 1000 карт.

Заметим, что кроме главного стандарта 80-колонной перфокарты имеются еще 45- и 90-колонные перфокарты. Некоторые из перфокарточных устройств ввода способны воспринимать кроме стандартных 80-колонных и другие перфокарты. Так, устройство ЕС-6016 способно воспринимать также 90-колонные, а устройство ЕС-6012 (СССР) — 45-колонные перфокарты. Устройство ЕС-6019 (СССР) имеет скорость считывания 1200 карт/мин.

4.1.3. Устройства вывода на перфокарты. Устройства такого рода представляют собой перфораторы, управляемые ЭВМ. Человек лишь закладывает колоду чистых (непробитых) карт в подающий карман и удаляет пробитые колоды из приемного кармана. Само же перфорирование производится автоматически. Скорость перфорации обычно колеблется от 100 до 250 карт/мин (примерно от 125 до 330 байт/с). Помимо устройства сопряжения с каналом, выходные перфораторы могут снабжаться также буферной электронной памятью для возможности после быстрого приема в буфер нужной информации производить перфорирование в автономном режиме (без непосредственного участия ЭВМ).

Заметим, что в выходных перфораторах ввиду сравнительно высокой скорости их работы (по сравнению с ручными клавишными перфораторами) удобно иметь два выходных кармана: пока один карман наполняется, другой может быть опорожнен.

4.2. Мультиплексные каналы

Перфокарточные и другие устройства с относительно малыми скоростями информационного обмена (по сравнению с внешними ЗУ) невыгодно подключать к описанным в предыдущей главе *селекторным каналам*. Вместо этого используются так называемые *мультиплексные каналы*, работа которых организуется по принципу разделения времени. При этом к каналу подсоединяется одновременно несколько работающих медленных устройств ввода-вывода. Информация по каналу идет с «метками» этих устройств, благодаря чему при ее обработке канал не перепутывает адреса. Благодаря большой скорости работы канал успевает обрабатывать сообщения, циркулирующие между ним и подключенными к нему внешними устройствами, и производить необходимые операции обмена с оперативным ЗУ.

Как и селекторные каналы, мультиплексные каналы имеют свой стандартный интерфейс, который максимально приближен к интерфейсу селекторных каналов ЭВМ того же типа.

4.3. Перфоленты

Перфолента представляет собой длинную ленту из плотной бумаги или пластмассы шириной порядка 2 см. Вдоль ленты организуется от 5 до 8 информационных дорожек, на которых в строго определенных позициях (на дистанции примерно 2,5 мм друг от друга) могут пробиваться (перфорироваться) отверстия. Комбинируя эти пробивки различным образом, можно представлять на ленте различные символы базисного алфавита. В 8-дорожечных лентах на это уходит одна, а в 5-дорожечных — 2 колонки пробивок.

На одном квадратном сантиметре ленты кодируется, таким образом, 1—2 символа. Этот показатель лучше, чем у перфокарт, но хуже, чем у обычных бумажных документов. Кроме того, в отличие от перфокарт, перфоленты хранятся свернутыми в рулоны с большими пустотами в центре, т. е. объемные плотности хранения информации у перфокарт и перфолент примерно одинаковы.

Скорости ввода и вывода информации у перфолент и перфокарт также примерно одни и те же. Цена перфоленты для хранения 1 млн. байт информации составляет 5—6 рублей. Единственным преимуществом перфолент перед перфокартами является относительная компактность и меньшая стоимость работающих с ними устройств считывания и перфорирования. Поэтому перфоленты часто используются в качестве вводных устройств в дешевых мини-ЭВМ и микро-ЭВМ.

Как и для перфокарт, для перфолент имеются свои исторические причины, объясняющие их живучесть. Дело в том, что задолго до появления ЭВМ перфоленты использовались и до сих пор продолжают использоваться в качестве промежуточных носителей информации в телеграфии: набив телеграмму на перфоленту (на телеграфном телетайпе), мы получаем возможность быстро передать ее (с помощью того же телетайпа), не заставляя линию ждать, пока телеграфист наберет следующую букву. В случае неправильного приема возможно осуществить новую передачу без повторного набора и т. д.

В последние годы происходит все большее сращивание ЭВМ с системами связи (для удаленного доступа и обмена информацией между удаленными ЭВМ), а это послужило дополнительным импульсом для продления жизни перфолент. Однако в целом перфоленты, как и перфокарты, в век безбумажной информатики являются анахронизмом и постепенно заменяются (не только в ЭВМ, но и в связи) магнитными носителями информации — магнитными лентами и магнитными дисками (в первую очередь гибкими).

4.3.1. Устройства подготовки данных на перфолентах. Как уже отмечалось выше, в качестве такого устройства может выступать обычный *телетайп*. Разумеется, при использовании его в этом качестве аппаратура, обеспечивающая подключение телетайпа к линии связи, становится ненужной. Второе устройство (еще более комплексное и дешевое) — это обыкновенная пишущая машинка, снабженная *перфоприставкой*. Такие машины, называемые *флексорайтерами*, осуществляют, как, впрочем, и телетайп, одновременно с перфорацией и обычную печать (на бумажный лист или ленту) отперфорированной информации, осуществляя тем самым непосредственный контроль правильности подготавливаемых данных.

Применительно к ЕС ЭВМ и СМ ЭВМ разработано несколько типов устройств подготовки данных на перфоленты, обладающих дополнительными возможностями, помимо простой перфорации ленты. Так, устройство ЕС-9024, помимо перфорирования ленты, обеспечивает также режим ручного контроля уже пробитой ленты (путем повторного набора), режим распечатки содержимого ленты на бланке, режим реперфорации для получения дубликата пробитой ленты, режим сравнения двух перфолент и др. Устройство позволяет осуществлять поиск заданного кода на ленте, останов по коду, автоматический пропуск некоторых кодов (в частности, *забитых*) и др. Скорость работы в режимах сравнения и реперфорации 50 строк/с, в режиме распечатки 10 строк/с.

Дополнительным неудобством перфолент по сравнению с перфокартами является большая трудность исправления неправильно отперфорированных данных. В случае перфокарт для этого достаточно заменить лишь неправильно набитые карты, в случае же перфолент приходится забивать и повторно вводить неправильно набранную информацию (что иногда приводит к необходимости перебивать всю ленту), либо делать разрезы и вклейки, либо исправлять ошибки уже после ввода — в памяти ЭВМ (используя добавочно вводимую информацию).

4.3.2. Устройства ввода с перфоленты. Главными в таких устройствах являются фотосчитывающая головка и механизм протяжки ленты. Этот механизм использует позиционную перфорацию на специальной дорожке (посередине ленты) и может осуществлять обычно как *непрерывную*, так и *стартстопную* протяжку (со считыванием по одной строке за раз). Скорость считывания в непрерывном режиме достигает 1500 строк/с.

Устройства ввода с перфолент имеют обычно не более одной бобины (в отличие от магнитных лентопротяжек), называемой в этом случае *диском намотки* или *подающей бобиной*. В устройствах ЕС ЭВМ емкость такой бобины равняется 285 м перфоленты (порядка 110 тыс. строк). В них используется несколько видов перфолент шириной 17,5 и 25,4 мм с числом дорожек, равным 5, 6, 7 или 8. В некоторых устройствах, предназначенных для протяжки относительно коротких перфолент, подающая бобина вообще отсутствует.

Приведем в виде примера характеристики устройства (механизма) ввода информации с перфолент FS-1501 (BHP). Устройство работает и имеет скорость считывания 1500 строк/с с числом дорожек 5, 6, 7 или 8 в стартстопном режиме. Масса устройства 17,6 кг. Термин «механизм» призван подчеркнуть то обстоятельство, что в устройстве не содержится электронных схем сопряжения с каналами ЕС ЭВМ, а имеется лишь собственно устройство считывания с электронным регистром на один символ.

Сопряжение с каналом осуществляется отдельным устройством. В устройство ввода с перфолент ЕС-6022 (СССР) включен электронный блок сопряжения с каналом. Устройство работает с числом дорожек от 5 до 8 в непрерывном и стартстопном режиме, обеспечивая в обоих случаях скорость считывания 1500 строк/с. Простые безбобинные механизмы ввода с перфоленты обычно обеспечивают возможность считывания информации с *карт с краевой перфорацией.* Такие карты шириной 76,2 или 82,6 мм имеют на краях круглые отверстия, часть из которых прорезана до края. Комбинация отверстий двух типов составляет двоичный код признаков содержимого карты. Полный текст содержимого карты наносится на нее обычным шрифтом. Будучи сложены в специальный ящик, карты допускают ручную выборку по признакам. С этой целью используют длинную спицу, которую продевают через отверстия всех карт в определенной позиции. Поднимая спицу, извлекают из ящика все карты, у которых отверстия в данной позиции не прорезаны до края. Повторяя эту операцию для других позиций, можно извлекать нужные карты по любому многозначному признаку.

Примером устройства (механизма) ввода с перфолент и с карт с краевой перфорацией может служить устройство (механизм) ЕС-6191 (ВНР). Механизм работает с перфолентами с числом дорожек 5—8 и с картами с краевой перфорацией в стратстопном и быстром (квазинепрерывном) режимах, обеспечивая считывание не только при прямой, но и при обратной подаче. Скорость считывания в стартовом режиме (вперед-назад) 40 строк/с, а в режиме быстрого считывания 150 строк/с.

4.3.3. Устройства вывода на перфоленту. Электромеханические перфораторы, управляемые ЭВМ. Примером такого устройства (с блоком стандартного сопряжения с каналом ЕС ЭВМ) может служить устройство ЕС-7022 (СССР). Оно работает в стартстопном режиме и обеспечивает скорость перфорирования до 150 строк/с. Форма выходной информации — 8- либо 5-дорожечная перфолента. Скорость вывода 150 строк/с является типичной для ленточных перфораторов. Повышение этой скорости встречает многие трудности конструктивного характера.

4.4. Читающие автоматы

Как уже отмечалось выше, использование бумажных (хотя и машинно-ориентированных) носителей информации в виде перфокарт и перфолент представляет собой в какой-то мере дань истории. Становление безбумажной информатики постепенно заменяет их магнитными носителями информации. Однако переход к безбумажной информатике не означает полного отказа от традиционных бумажных форм представления информации, ориен-

тированных на человека. Речь идет о переводе в память ЭВМ и на машинные носители подавляющей массы информации.

Наличие ориентированных на человека бумажных документов ставит задачу автоматизации их ввода в ЭВМ. Эта задача решается специальными устройствами, получившими название *читающих автоматов*. Обычная схема работы читающего автомата такова. В *приемный бункер* загружается пачка разброшюрованных документов (по одному листу). В простейших читающих автоматах устанавливается при этом стандарт на формат листов. Более совершенные автоматы работают с листами разного формата (меняющегося в некоторых заданных пределах). В загружаемой пачке эти разноформатные документы должны быть выровнены по одному из углов (удобнее всего по левому верхнему).

Из приемного бункера тем или иным способом (например, с помощью специального рычажного механизма с вакуумными присосами) листы один за одним передаются на считывающий механизм. Он может представлять собой вращающийся барабан с вакуумным присосом, вдоль оси которого осуществляется сканирование помещенного на барабан листа с текстом специальным блоком фотоголовок. Особая электронная схема (достаточно сложная) осуществляет последовательное распознавание и двоичное кодирование напечатанных на листе букв, цифр и других символов. С помощью специального блока сопряжения эта информация передается в канал ЭВМ.

Читающий автомат обычно заранее настраивается на тот или иной печатный шрифт. После такой настройки можно осуществлять считывание текста с листа с достаточной скоростью (колеблющейся в зависимости от типа автомата от нескольких сот до нескольких тысяч символов в секунду).

Надежность распознавания при настройке на заданный шрифт также может быть сделана достаточно высокой. Количество возможных ошибок на миллион распознанных символов у хороших читающих автоматов гораздо меньше, чем при ручном вводе соответствующего текста. В ряде случаев частичной непропечатки или частичного замазывания символа автомат продолжает его правильно распознавать, в то время как человек теряет такую возможность. Описанные результаты были получены уже на первых экспериментальных образцах отечественых читающих автоматов ЧАРС, разработанных еще в конце 60-х годов.

Попытки построить читающий автомат, способный распознавать (без предварительной настройки) любые печатные шрифты, приводят к огромному его усложнению и удорожанию. Задача же автоматического чтения произвольных рукописных текстов не решена в полном объеме до сих пор. Правда, если ограничиться строгими правилами написания символов и писать их раздельно друг от друга, то задача автоматического чтения

подобной (формализованной) рукописной записи решается относительно просто. Применительно к распознаванию десятичных цифр такая задача не только решена, но и широко внедрена в практику автоматического распознавания рукописных цифровых индексов на почтовых конвертах.

Оценивая состояние автоматического ввода бумажных документов в ЭВМ в целом, можно сказать, что при большом числе символов, а также разнообразии шрифтов и форматов бумаги он обходится пока значительно дороже по сравнению с вводом перфокарт, перфолент и других машинно-ориентированных носителей информации. Этим обстоятельством объясняется не слишком широкое распространение читающих автоматов. Правда, успехи микроэлектроники позволят несколько улучшить положение. Однако бумажным документам свойственны и такие трудно устранимые недостатки, как непрочность и неудобство машинного манипулирования. Поэтому при проектировании автоматизированных систем данных стремятся обойти бумажные документы как источник информации для ЭВМ всякий раз, как это оказывается возможным.

4.4.1. Карты с карандашными отметками. Ввод информации в ЭВМ значительно облегчается, если эта информация заранее сформирована таким образом, что человеку, использующему подобную *форматированную карту*, достаточно подчеркнуть нужное или вычеркнуть ненужное. Подобные отметки (делаемые для надежности специальными карандашами) легко считываются относительно простыми *специализированными читающими автоматами*. Их специализация заключается в том, что они лишь устанавливают факт наличия или отсутствия отметки в той или иной позиции. Содержательное же значение этого факта (что означает отметка в данной позиции) заложено в конструкцию устройства или в память ЭВМ, к которой это устройство подключено.

Карты с карандашными отметками используются при массовых анкетированиях и переписях населения. Использовались они, в частности, в последней всесоюзной переписи населения в 1979 г. В мировой практике широко применялись, а кое-где и сейчас применяются так называемые *дуаль-карты*. Они представляют собой по существу обычные перфокарты, только заранее форматированные и потому пригодные для нанесения на них информации с помощью карандашных отметок. Прочтя эти отметки, специальное устройство может осуществить пробивку перфокарт в соответствии с информацией, записанной первоначально с помощью отметок. Тем самым существенно облегчается подготовка исходных данных на перфокартах, ибо исключается ручное перфорирование карт.

Отметки делаются не только быстрее, чем перфорирование, но и, что очень важно, без специального оборудования и без спе-

циально подготовленного персонала. Например, при использовании дуаль-карт на производстве они заполняются обычным производственным и административно-управленческим персоналом. При этом по сравнению с традиционными методами фиксации учетной и управленческой информации их работа существенно упрощается.

4.5. Печатающие устройства

4.5.1. Устройства последовательной печати. Простейшим устройством вывода информации из ЭВМ на печать является обыкновенная электрифицированная пишущая машинка, управляемая ЭВМ через специальный электронный блок *). Конструктивно этот блок либо совмещается с машинкой, либо встраивается в специальный стол, на котором крепится машинка. Помимо управления печатающим механизмом, в функции блока входит также сопряжение устройства с каналами ЭВМ.

Следует подчеркнуть, что обычные рычажные пишущие машинки, к числу которых принадлежит и электрифицированная пишущая машинка Консул (ЧССР), не могут обеспечить скорость печати выше 10—12 зн/с. Для повышения скорости используют другие типы механизмов печати, в которых возвратно-поступательные движения сведены к минимуму.

Одними из первых стали применяться легкие (сменные) *головки*, на которых нанесены символы рабочего алфавита машинки. Перемещаясь и вращаясь, головка обеспечивает подвод под заданную позицию любого из этих символов, после чего она прижимается к красящей ленте и через нее оставляет соответствующий отпечаток на бумаге.

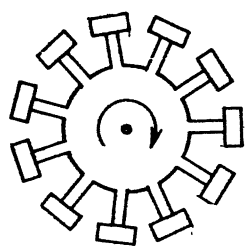

Рис. 4.1

Вместо головки, имеющей две степени свободы движения, можно использовать замкнутую *цепь*, несущую на себе литеры алфавита и перемещаемую с большой скоростью вдоль позиции печати. Как только нужная литера оказывается на этой позиции, осуществляется прижим к ней ролика с бумагой (через красящую ленту), т. е. печать соответствующего символа.

Еще один способ — быстро вращающийся *диск* с гибкими лепестками, на которых помещаются литеры (рис. 4.1); когда нужная литера находится в позиции печати, осуществляется ее прижим к бумаге через красящую ленту.

*) Роль такой машинки может выполнять обыкновенный телетайп.

Перечисленные способы позволяют поднять скорость печати при выводе из ЭВМ до 30—40 зн/с и даже несколько выше.

Еще большим быстродействием обладает так называемый *точечный* (игольчатый) механизм печати. Его основу составляет группа иголок, концы которых образуют прямоугольный точечный растр, соответствующий по размеру печатаемым литерам. В подобной игольчатой *матрице* каждая иголка может двигаться независимо от других с помощью специальных электромагнитных приводов. Прижимаясь через красящую ленту к поверхности бумаги, она оставляет на ней точечный след. Комбинациями таких точек можно представить любой требуемый символ.

Ввиду легкости иголок и малости их перемещений скорость точечных устройств печати достаточно высока, достигает 200 зн/с. Большим удобством устройств такого рода является то, что их алфавит определяется электроникой, так что при смене алфавита никаких переделок в механизме не требуется. К числу достоинств этих устройств относится и их малошумность. В печатающих устройствах со сменяемой головкой, а также цепных и лепестковых, настройка механизма на другой алфавит требует замены всего одной части. Наиболее же трудна замена у рычажных механизмов. В некоторых печатающих механизмах имеется возможность печати двумя цветами за счет оперативной (по ходу печати) автоматической подмены красящей ленты.

Примером устройства с точечной печатью является последовательно печатающий механизм ЕС-7186 (ПНР). Он использует квадратный точечный растр из 49 (7×7) точек и обеспечивает скорость печати 180 зн/с. Встроенная электроника обеспечивает печать 96 символов с возможностью расширения до 128 символов. Устройство имеет буферное ЗУ на 256 байт.

Большие достоинства имеют *струйные печатающие устройства*. В них используется принцип нанесения знаков на обычную бумагу с помощью микроструи красящего вещества (чернил), выдавливаемой из миниатюрного сопла. При выходе из сопла капельки, из которых состоит струя, заряжаются электростатическим зарядом и отклоняются управляемым электрическим полем. Управление полем делается специальной электронной схемой — так называемым *знакогенератором*. Другой способ не предусматривает отклонения струи. Все необходимые для получения различных знаков операции происходят за счет относительного перемещения сопла и листа бумаги (в двух взаимно перпендикулярных направлениях), а также за счет включения и выключения струи в нужные моменты времени. Во втором способе требуется большой объем буферной памяти в устройстве (обычно для запоминания целой строки). Зато очень легко реализуется многоцветная печать за счет установки соответствующего числа сопел.

4.5.2. Устройства строчной печати. При необходимости еще более повысить скорость печати употребляются *алфавитно-цифровые печатающие устройства* (сокращенно АЦПУ) *с параллельной* (строчной) печатью. Как следует из их названия, эти устройства печатают не последовательно буква за буквой, а сразу целую строку.

Одними из первых получили распространение *барабанные* АЦПУ. Основу их механизма составляет длинный (во всю длину строки) барабан. По дорожкам поперек оси барабана располагаются выпуклые литеры, по полному экземпляру литер алфавита на каждую дорожку. Число дорожек равно числу знаков (символов) в строке. Печатаемая строка располагается вдоль образующей барабана. *Прижимные молоточки* действуют независимо друг от друга, приходя в движение, когда нужные буквы алфавита находятся под ними, так что за один оборот барабана вся строка оказывается напечатанной.

В *цепных* АЦПУ литеры насаживаются на замкнутую цепь, движущуюся вдоль строки. Прижимные молоточки в каждой позиции строки ловят момент, когда под ними находится нужная буква. При полном прохождении всего алфавита вдоль строки строка оказывается напечатанной.

АЦПУ *со знаковыми колесами* можно представить себе в виде барабанного АЦПУ, у которого барабан разрезан на отдельные диски (колеса), по одному на каждую позицию строки, вращающиеся самостоятельно. Перфорация служит для подачи бумаги при переходе к печати следующей строки. В подающем кармане может храниться запас ленты длиной до 1000 метров. Ширина ленты может колебаться от 80 до 420 мм, максимальное число символов в строке (число знаковых колес) 128, скорость печати до 900 строк/мин (15 строк/с). Максимальное количество копий (через копировальную бумагу) 4. Переключение на различные форматы бумаги производится вручную. Возможна печать на форматных бланках без краевой перфорации. Управление перемещением бланков производится с помощью перфоленты с 4 дорожками. Устройство имеет буферное ЗУ на 128 байт и снабжается пультом оператора, на котором находится сигнализация (например, сигнал об окончании бумажной ленты и необходимости заправки новой ленты), а также клавиши управления (пуск, холостой ход и т. п.). Специальный электронный блок, придаваемый АЦПУ, обеспечивает автономное управление им, а также сопряжение со стандратным каналом ЕС ЭВМ. Польское АЦПУ ЕС-7033 обеспечивает большую скорость печати (1100 строк/мин).

Однако, вообще говоря, механические АЦПУ практически не допускают увеличения скорости выше 1200—1500 строк/мин (20—25 строк/с). Гораздо большая скорость печати (несколько

сот строк в секунду) обеспечивается в *электрохимических* АЦПУ. В них вдоль строки располагается линия иголок, по нескольку на каждый символ строки. При подаче на иголку электрического импульса в находящейся под ней бумаге, пропитанной специальным химическим составом, возникает изображение точки. Управляя соответствующим образом подачей импульсов на иголки при перемещении бумаги на высоту буквы в ней возникает изображение целой строки букв (собранное из точек).

На электрохимических АЦПУ можно легко варьировать алфавиты, выдавать не только алфавитно-цифровую, но и графическую информацию. Большим недостатком, однако, является использование специальной бумаги и относительно невысокое качество печати. Этих недостатков лишены *струйные* строчные печатающие устройства, которые работают по уже описанному выше принципу, но с большим числом сопел одновременно (на всю строку). Такие устройства сохраняют практически все преимущества электрохимических устройств (включая высокую скорость печати) и вместе с тем лишены их главных недостатков.

4.5.3. Страничные принтеры. В последние годы ряд фирм освоили производство печатающих устройств с использованием лазеров, обеспечивающих скорость вывода свыше 20 тыс. строк/мин. Эти устройства, получившие наименование *страничных принтеров*, отличаются весьма большой гибкостью, позволяя осуществлять печать на бумаге различных форматов с двух сторон разными шрифтами, выводить графическую информацию и т. п. Они допускают использование в качестве копирующих устройств. Некоторые из выпускаемых страничных принтеров допускают многоцветную печать.

4.6. Устройства ввода и вывода графической информации

До широкого применения цифровой вычислительной техники в контрольно-измерительной технике для регистрации информации, поступающей от контрольно-измерительных приборов, использовались различного рода самописцы. С их помощью информация представлялась в виде графиков на бумажных носителях либо на фотопленке. Для ввода таких графиков в ЭВМ используются специализированные графосчитывающие устройства. Они представляют собой по существу один из видов аналого-дискретных преобразователей, рассмотренных в гл. II.

В настоящее время в связи с развитием цифровой измерительной техники подобный способ ввода информации в ЭВМ следует считать устаревшим. При необходимости промежуточной фиксации информации, получаемой от измерительных приборов с цифровым выводом, в настоящее время принято использовать магнитную ленту. Фиксируя не аналоговую, а цифровую информацию,

магнитная лента может быть впоследствии использована в ЭВМ. Кроме того, подобный способ фиксации информации устраняет дополнительные ошибки, которые возникают при представлении и последующем считывании аналоговой информации в виде графиков.

Второй вид устройств, употребляющихся для ввода графической информации,— так называемые *считыватели координат*. Внешне такое устройство представляет собой прямоугольный планшет, по которому вручную можно перемещать специальную головку. Эта головка обычно снабжена лупой с перекрестием. Наведя это перекрестие на ту или иную точку поверхности планшета (путем соответствующего перемещения головки), автоматически измеряют координаты этой точки в осях, совпадающих с двумя сторонами планшета. На планшет помещается чертеж или карта, с которой и производится считывание информации. При движении головки вдоль некоторой линии можно автоматически вычислять и передавать в ЭВМ координаты точек, расположенных на этой линии (их густота зависит от скорости движения головки).

Для реализации автоматического считывания координат используется несколько принципов. Один из самых простых заключается в том, что в головке (выполняемой в этом случае в виде иглы) находится источник ультразвука. Этот звук воспринимается приемниками, расположенными по сторонам планшета. По времени задержки прихода звука на приемники специальное электронное устройство вычисляет координаты излучающего конца иглы.

Большой точности съема координат можно достичь в мозаичных планшетах. Датчики в них размещены под поверхностью планшета, а измеряемой ими величиной может служить, например, емкость между ними и обкладкой, установленной на передвигаемой по планшету головке.

4.6.1. Графопостроители. Устройства для вывода информации из ЭВМ на бумажные носители принято называть *графопостроителями*.

Графопостроитель *планшетного* типа (рис. 4.2) представляет собой планшет, на котором закрепляется (обычно вакуумным присосом) лист чертежной бумаги. *Пишущий узел* (головка) укрепляется на траверсе, которая специальными шаговыми двигателями может перемещаться вдоль длинных сторон планшета. Сам пишущий узел способен перемещаться (также с помощью шагового двигателя) вдоль траверсы, т. е. в направлении, параллельном коротким сторонам планшета. В пишущем узле на специальном поворотном устройстве закреплены чернильные перья с разноцветными чернилами, которые и производят вычерчивание требуемых линий, точек и символов. Чтобы не загружать память

ЭВМ подробной информацией о вычерчиваемых символах, эта информация обычно встраивается в специальный электронный блок так, что, получив от ЭВМ лишь код символа и привязывающие его к определенному месту чертежа координаты, графопостроитель может вычертить его самостоятельно в автономном режиме. Этот же блок осуществляет обычно линейную и круговую *интерполяцию*, так что, например, для вычерчивания отрезка прямой из ЭВМ достаточно передать лишь координаты его начала и конца.

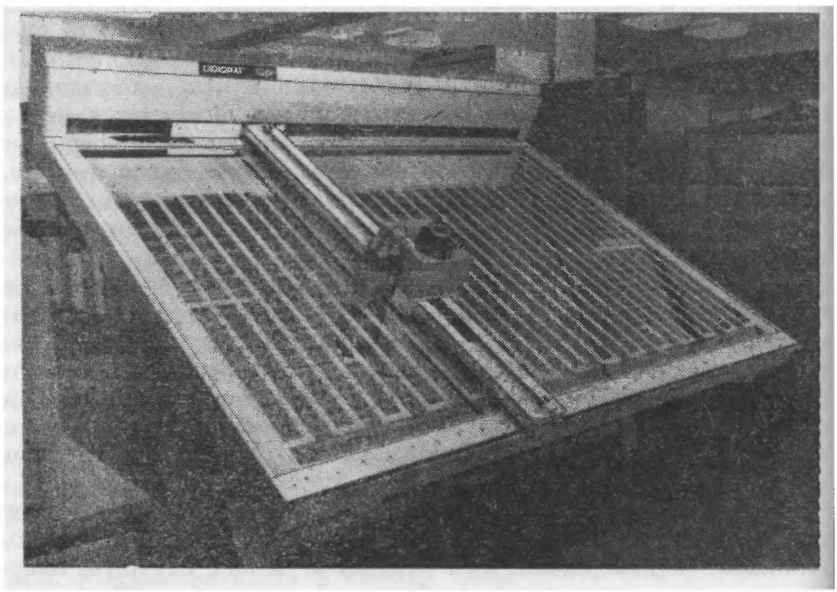

Рис. 4.2

Из дополнительных функций, выполняемых блоком управления графопостроителем, отметим еще возможность изменения масштаба (в автономном режиме), изменение толщины и характера вычерчиваемых линий и др. Кроме того, блок управления осуществляет сопряжение графопостроителя с каналом ЭВМ. Иногда добавляется возможность полностью автономной работы графопостроителя с машинным носителем, на который заранее выведена из ЭВМ требуемая информация.

В качестве примера рассмотрим графическое регистрирующее устройство (графопостроитель) планшетного типа ЕС-7015 (СССР). Оно выполнено в виде стола с добавочным шкафом для электронных блоков. Формат листа бумаги, закрепляемого на планшете, 1200×1150 мм. Размеры рабочего поля 1050×1000 мм. Элемен-

тарный шаг пишущего узла 0,05 мм. Максимальная скорость вычерчивания 50 мм/с. Возможности автономного изменения масштаба — в отношениях 1:2, 1:1, 2:1. Толщина линий записи (трехцветной) 0,3 и 0,5 мм, тип линий записи — сплошной (непрерывный), штриховой и штрихпунктирный. Ориентация вычерчиваемых по коду 253 символов — под 16 углами (через 22,5°). Имеется автономная линейная и круговая интерполяция.

Другой тип графопостроителей — так называемый *рулонный* — предназначен для вычерчивания чертежей и рисунков на длинном рулоне бумаги, перематываемой с помощью двух валиков как в одном, так и в другом направлениях. Это движение бумаги заменяет движение траверсы у планшетных графопостроителей. В данном случае траверса укреплена неподвижно (параллельно осям валиков). Пишущий узел перемещается по траверсе точно так же, как и у планшетного графопостроителя.

В качестве примера рассмотрим рулонный графопостроитель ЕС-7052 (СССР). Он работает с рулоном бумаги длиной 80 м и шириной 420 мм. Размеры рабочего поля 390×600 мм. Элементарный шаг пишущего узла 0,1 мм. Максимальная скорость вычерчивания 200 м/с. Вычерчивание производится непрерывными линиями трех цветов толщины 0,3—0,9 мм. Как и в предыдущем случае, имеется возможность автономного изменения масштаба, а также линейная и круговая интерполяция. Ориентация вычерчиваемых знаков — либо вертикальная, либо горизонтальная. Графопостроитель может работать в автономном режиме с управлением от перфоленты.

Для графопостроителей (графических регистрирующих устройств) иногда употребляется еще одно название — *плоттеры*, причем планшетные графопостроители называются *двухкоординатными*, а рулонные — *однокоординатными* плоттерами. Последнее название не вполне точно (особенно в случае возможности движения рулона бумаги в двух направлениях), ибо этим движением как раз и реализуется вторая координата.

Заметим, что скорость вычерчивания можно увеличивать за счет увеличения элементарного шага пишущего механизма. Но в таком случае уменьшается точность вычерчивания. Поэтому качество графопостроителя принято измерять произведением скорости на точность или частным от деления скорости на величину элементарного шага пишущего механизма (что в принципе то же самое). Для ЕС-7051 этот последний показатель равен 1000, а для ЕС-7052 — 2000 $с^{-1}$.

В составе ЕС ЭВМ имеется графопостроитель рулонного типа ЕС-7053, у которого этот показатель равен 3000 $с^{-1}$ (шаг 0,05 мм, скорость 150 мм/с). Он работает с широкой бумажной лентой (ширина 878 мм, длина 20 м) и обеспечивает все функции устройства ЕС-7051.

Еще одно устройство для ввода и вывода графической информации — так называемый *графический дисплей* — рассматривается ниже в разделе диалоговых средств общения с ЭВМ.

4.7. Устройства подготовки данных на магнитных носителях

Как уже отмечалось выше, подготовка данных для ЭВМ на перфокартах и перфолентах представляет собой дань предшествующей исторической эпохе. В эпоху перехода к безбумажной информатике она постепенно вытесняется подготовкой данных непосредственно на машинных носителях — магнитной ленте и магнитных дисках. Основная трудность, связанная с таким переходом, заключается в том, что эффективное использование поверхности магнитных носителей требует высокой скорости записи, несовместимой с возможностями человека. Поэтому устройства подготовки данных на магнитных лентах и дисках должны содержать достаточно сложные электронные блоки сопряжения с необходимой буферной памятью.

В эпоху ЭВМ 1-го и 2-го поколений, когда электроника была относительно дорога, подготовка данных на магнитных носителях использовалась довольно редко. Успехи микроэлектроники сделали такую подготовку экономически выгодной. В настоящее время выпускаются устройства подготовки данных на магнитной ленте бобинного и кассетного типа, а также устройства подготовки данных на гибких магнитных дисках. Для подготовки данных на магнитных носителях (включая и сменные магнитные диски) иногда используют дешевые ЭВМ, снабженные пультами набора данных в виде электрифицированных пишущих машинок, телетайпов или алфавитно-цифровых дисплеев (см. ниже).

В качестве примера устройства подготовки данных на магнитной ленте бобинного типа рассмотрим устройство ЕС-9002 (НРБ). Работая на клавиатуре, оператор вводит данные в буферную память, содержимое которой высвечивается на экране дисплея, расположенного непосредственно над клавиатурой. Осуществляя непосредственный визуальный контроль вводимой информации, оператор может тут же исправить возможные ошибки. При наличии повторяющихся групп данных они дублируются автоматически, без полного повторного ручного ввода. Эта и ряд других программно-реализуемых функций позволяют повысить на 30% производительность оператора по сравнению с перфорацией карт и лент.

После ввода и проверки блока длиной 80 или 160 символов он автоматически переписывается на стандартную магнитную ленту с плотностью 32 бит/мм. Необходимые промежутки между блоками обеспечиваются устройством автоматически. Записанная ин-

формация может непосредственно вводиться в ЭВМ через стандартные лентопротяжки. Как и для других магнитных носителей, имеются по крайней мере три важных преимущества перед подготовкой данных на перфоносителях. Во-первых — гораздо бо́льшая информационная плотность и гораздо меньшая удельная стоимость магнитных носителей. Во-вторых — увеличение производительности в процессе подготовки данных и особенно при вводе их в ЭВМ. В-третьих — возможность многократного использования носителя для подготовки различных данных.

Устройство ЕС-9006 (ВНР) обеспечивает запись данных с клавиатуры на кассетную магнитную ленту, которая может читаться на ЕС ЭВМ с помощью тех же устройств ЕС-9006 или устройства ЕС-5094 (входящего в состав ЕС-9006). Емкость кассеты 80 Кбайт на каждой стороне. Емкость блока 80 символов.

Устройство снабжено бесконтактной клавиатурой, работающей (по емкостному принципу) от легких прикосновений пальцев. Тем самым обеспечиваются дополнительные удобства для оператора. Как и устройство ЕС-9002, оно обеспечивает визуальный контроль, возможность исправления и многие другие дополнительные функции.

Те же функции и те же принципы построения характерны и для устройств подготовки данных на гибких магнитных дисках.

Весьма перспективными являются устройства подготовки данных на сменных блоках ЗУ на магнитных доменах. Их безынерционность сильно упрощает организацию работы на различных скоростях — весьма малых при подготовке данных человеком и весьма больших при ее чтении ЭВМ. Отпадает необходимость в буферных ЗУ и сложных схемах сопряжения. Привлекают и миниатюрные размеры сменяемых блоков, облегчающие их перемещение. Заметим, что приборы с зарядовыми связями, в отличие от магнитно-доменных, не годятся для автономной подготовки данных (с последующим переносом на ЭВМ) вследствие того, что запасенная в них информация теряется при отключении питания.

4.8. Диалоговые терминалы

Диалоговый терминал — это вводно-выводное устройство, на котором пользователь может вести диалог с ЭВМ. В качестве такого устройства могут быть использованы обычные электрифицированные пишущие машинки или телетайпы. Для *диалогового режима* чрезвычайно важно максимально уменьшить время реакции ЭВМ на запросы пользователя. Поэтому желательно сокращать по возможности и время печати ответов ЭВМ. Такую возможность дают печатающие машинки с точечным (игольчатым) растром. Однако, если ответы коротки, а время «обдумыва-

ния» ответа машиной велико, то увеличение скорости печати мало помогает делу. Это показывает, что выбор типа машинки должен каждый раз определяться конкретными условиями диалога.

Поскольку машинка документирует диалог, то для его последующего разбора удобно, чтобы информация, исходящая от человека и от ЭВМ, печаталась различными цветами. В машинках, в которых это возможно, обычно используется в таком случае красный и черный цвет.

4.8.1. Алфавитно-цифровые дисплеи. Особые удобства для диалога представляют экранные пульты (терминалы) — так называемые *дисплеи*. Обычно дисплей строится с помощью *электронно-лучевой трубки*, аналогичной кинескопу обычного телевизора. Дрожание изображения, характерное для изображения на экране телевизора, в дисплеях устраняется за счет более высокой частоты обновления кадров (обычно 50 кадров/с вместо 24 кадров/с у телевизора).

Подобное повышение в случае телевидения (при принятых ныне принципах его работы) привело бы к необходимости увеличения полосы частот передатчиков и, как следствие, к резкому удорожанию всей системы. В электронно-лучевых дисплеях обновление кадров производится из буферной памяти, конструктивно совмещенной с трубкой в одном футляре, так что никаких дальних передач информации не требуется. Информация, поступающая в дисплей от ЭВМ, служит лишь для обновления информации в буферном ЗУ. Ее передача может вестись на гораздо меньших скоростях, соответствующих возможностям каналов ЭВМ, поскольку, как правило, полное изменение кадра требует предварительного прочтения старого кадра человеком.

С целью уменьшения объема буферного ЗУ при отображении алфавитно-цифровой информации в ЗУ запоминается для каждой позиции экрана только код символа, который должен высвечиваться в этой позиции. Изображение же символа генерируется по его коду специальным электронным устройством, называемым *генератором символов*.

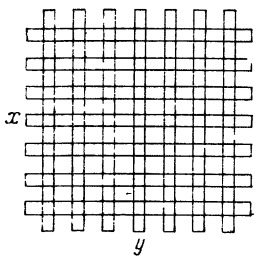

Рис. 4.3

Помимо электронно-лучевых существуют еще так называемые *плазменные дисплеи*. Они строятся на принципе *тлеющего разряда*, возникающего в разреженном газе между двумя проводниками, к которым приложено электрическое напряжение. Подбором газа можно получать различный цвет разряда. В плазменном дисплее микроразряды создаются в точках пересечения двух систем металлических полосок, нанесенных на поверхности двух плоских стекол (рис. 4.3), помещенных на

очень малом расстоянии друг от друга. Промежуток между стеклами заполнен разреженным газом. Полоски настолько тонки, что прозрачны и даже практически невидимы, так что глаз воспринимает не их, а отдельные разряды точечного размера. Между системами полосок x и y приложено постоянное напряжение, достаточное, чтобы поддерживать тлеющий разряд, но недостаточное, чтобы его зажечь. Для зажигания разряда на пересечении двух полосок x и y на эти две полоски подаются импульсы, которые увеличивают напряжение выше порога зажигания только на их пересечении. Тушится разряд подачей импульсов противоположных знаков.

Преимуществом плазменного дисплея является то, что он не требует буферного ЗУ для поддержания изображения. Сам дисплей (точнее, система тлеющих разрядов) представляет собой своеобразное ЗУ с непосредственно наблюдаемой информацией, которая в нем хранится. Второе преимущество плазменного дисплея — плоский прозрачный экран. Это позволяет дополнять изображение на его экране любыми картинками, проецируемыми с помощью видеопроектора на заднюю стенку экрана (рис. 4.4).

В странах СЭВ большое распространение получили венгерские алфавитно-цифровые дисплеи. Приведем характеристики одного из них, имеющего шифр ЕС-7061. Частота регенерации кадров 50 кадров/с. Максимальное число знаков, одновременно воспроизводимых на экране, 1024. Число символов в наборе не менее 64. Размер экрана 150×200 мм. Буферная память — на 1024 байт. Максимальная скорость передачи данных 100 Кбайт/с.

Помимо клавиши для набора символов базисного алфавита (сокращенного) имеются еще *функциональные клавиши* для управления дисплеем. Одной из функций управления является управление специальным значком, называемым *маркером* или *курсором*, который с помощью специальных клавиш может двигаться в двух взаимно перпендикулярных направлениях и устанавливаться в любую позицию на экране. С помощью маркера можно указывать место, с которого надо воспроизводить печатаемую информацию. Можно производить стирание отдельных знаков и целых строк. Специальной клавишей можно стереть и весь текст на экране. Во многих типах дисплеев набор подобных *редакторских* функций очень богат: можно производить автоматическую раздвижку текста при впечатывании в него новой информации, менять форматы текста и т. п.

К дисплею придается блок стандартного сопряжения с каналом ЕС ЭВМ, благодаря чему информация с дисплея может передаваться в ОЗУ ЭВМ и обратно. Быстрота отображения информации, наглядность ее представления и возможности редактирования делают дисплейный терминал гораздо более удобным для пользователя, чем пишущая машинка. Единственный недостаток

дисплея — отсутствие документированного диалога — может быть восполнен специальной программой, осуществляющей документирование с помощью любого подсоединенного к ЭВМ печатающего устройства. Имеются и такие дисплеи, которые с помощью быстрого (сухого) фотографического процесса за несколько секунд дают бумажную копию (обычно уменьшенную) всего, что было изображено на экране в момент нажатия кнопки «копировать».

Рис. 4.4

4.8.2. Графические дисплеи. Будучи устроены в принципе так же, как и алфавитно-цифровые дисплеи, графические дисплеи отличаются от них возможностью высвечивания на экране любых графиков и рисунков (в лучших дисплеях — полутоновых и даже цветных). Для получения рисунков хорошего качества они, как правило, имеют большее разрешение, чем алфавитно-цифровые дисплеи. Помимо генераторов символов, в графический дисплей

встраиваются генераторы векторов различной длины и направлений, а также, возможно, и других линий (дуг окружностей, простых кривых, заданных уравнениями, и др.). Используя программы, закладываемые в ЭВМ, можно производить и другие, более сложные операции: повороты, перемещения, изменение масштабов и др.

Графические дисплеи на электронно-лучевых трубках снабжаются обычно так называемым *световым карандашом* (пером). В конец этого карандаша встроен фотодатчик. Если фиксировать концом карандаша какую-либо точку на экране, то в процессе развертки изображения электронный луч трубки засветит датчик. Полученный в результате этого электрический импульс передается по подсоединенному к карандашу проводу на специальный электронный блок дисплея, вырабатывающий по времени прихода импульса координаты точки касания карандаша. Эти координаты передаются в буферную память и используются в следующем цикле развертки для высвечивания на экране отмеченной точки. Передвигая конец карандаша по экрану, осуществляют ввод (по точкам) и последующее высвечивание траектории его движения. Переключением специального тумблера на карандаше он превращается в «резинку», стирая засвеченные точки, а при движении карандаша и целые линии.

В составе ЕС ЭВМ имеется графический дисплей ЕС-7064 (СССР), изображенный на рис. 4.5. Он имеет буферное ЗУ на 4096 байт с циклом не более 2 мкс. Размеры рабочего поля экрана 250×250 мм. Скорость вычерчивания векторов 3 мм/мкс. Наряду с чертежами дисплей может отображать и различные символы, в том числе буквы и цифры.

4.8.3. Диалог голосом. Вывод информации из ЭВМ обычным человеческим голосом может быть осуществлен двумя путями. В одном из них на специальном устройстве (с прямым доступом) записываются отдельные слова или фразы, произносимые диктором. Задача ЭВМ состоит в том, чтобы последовательно выбирать записанные фрагменты речи и формировать из них нужные ответы. Второй путь — прямой синтез (с помощью специального электронного синтезатора) первичных звуковых элементов человеческого языка — так называемых фонем и формирование из них слитной речи.

На обоих путях достигнуты практические успехи. Разработаны и действуют различного рода справочно-информационные системы, выдающие ответы голосом. Машинной «речи», правда, недостает богатства интонаций и эмоциональности, свойственных человеческой речи, но для деловых разговоров возможностей машинного голоса вполне достаточно. Заметим, что более современными и более быстро развивающимися являются звукосинтезирующие системы.

Ввод человеческого голоса в ЭВМ представляет собой более трудную задачу. Правда, когда речь идет о небольшом числе слов (два-три десятка), произносимых раздельно и достаточно отчетливо, решить задачу автоматизации ввода относительно просто.

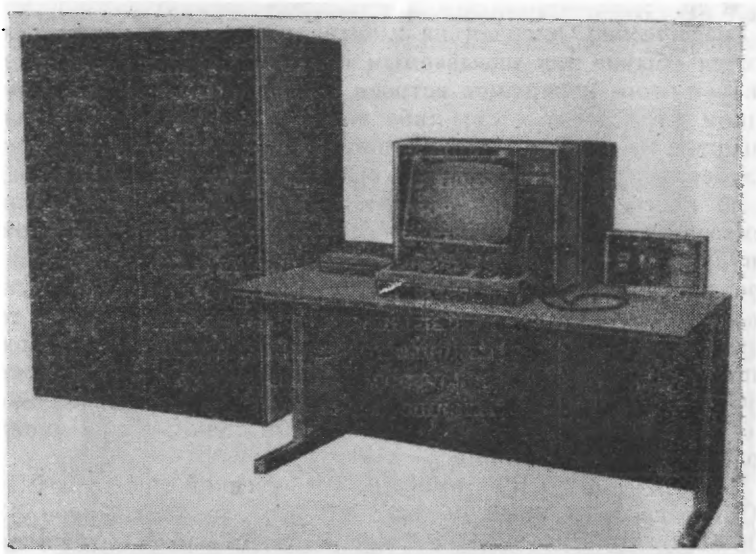

Рис. 4.5

В случае же обычной слитной речи с достаточно большим словарем (порядка тысячи слов и более), да еще в условиях помех (в виде производственных шумов), задача достаточно надежного распознавания была решена сравнительно недавно в СССР, США и Японии (пока в экспериментальных лабораторных образцах), так что голосовые диалоговые терминалы для работы с ЭВМ в сколько-нибудь массовых масштабах — пока дело будущего.

4.9. Удаленные терминалы и абонентские пункты

Дальность передачи обычных сигналов, используемых в каналах ЭВМ, относительно невелика. В зависимости главным образом от быстродействия канала она колеблется от нескольких метров до нескольких сотен метров. При передаче на большие расстояния нужно использовать либо специальные устройства для восстановления слабых сигналов в условиях шумов, либо коренным образом менять форму представления сигналов для передачи их по существующим каналам телеграфной и телефонной связи.

Применительно к первому способу в составе ЕС ЭВМ имеется устройство преобразования сигнала низкого уровня ЕС-8028 (ВНР), работающее по 2-проводной или 4-проводной физической линии в так называемом *дуплексном режиме*, т. е. с одинаковой скоростью одновременно в прямом и обратном направлениях. Расстояние устойчивой работы зависит от числа проводов и скорости передачи информации. При двух проводах и скорости 2400 бит/с дальность передачи достигает 16 км, при четырех проводах и той же скорости 28 км. При четырех проводах и скоростях 4800 и 9600 бит/с максимальная дальность равна соответственно 18 и 10 км.

В состав СМ ЭВМ моделей 1 и 2 включен так называемый *мультиплексорный разветвитель интерфейсов* (РИМ), сопрягающийся на расстоянии нескольких километров с ЭВМ через стандартный машинный интерфейс. Устройство РИМ-1 может быть удалено от ЭВМ на расстояние до 3 км, а устройство РИМ-2 (подсоединяемое через РИМ-1) увеличивает это расстояние еще на 8 км. К каждому из устройств РИМ-1 и РИМ-2 может подключаться до 16 периферийных устройств.

Для выхода на обычные телефонные и телеграфные каналы связи сигналы преобразуются специальными устройствами модулирования и *демодулирования* — так называемыми *модемами*. При модуляции сигнал из машинного вида преобразуется в «связной» вид (т. е. вид, используемый в канале связи). При демодуляции производится обратное преобразование.

Кроме модемов, меняющих *физическое* представление сигналов, при сопряжении с каналами связи необходимо менять их *логическое* представление. В частности, из побайтового представления, принятого в каналах ЭВМ, при выходе в каналы связи необходимо переходить к побитовому представлению. Необходимо также принять дополнительные меры по защите от ошибок. Все эти функции выполняются аппаратурой передачи данных (сокращенно АПД). При работе с коммутируемыми каналами связи в состав АПД включается обычно устройство автоматического вызова, формирующего и посылающего в линию номер требуемого абонента. О принципах построения АПД и модемов несколько более подробно говорится ниже, в главе, посвященной сетям ЭВМ.

По скорости передачи данных различают: 1) *Низкоскоростные* модемы (до 300 бит/с), использующие телеграфные каналы связи. Их примерами могут служить модемы-200 ЕС-8001 (НРБ) и ЕС-8002 (ГДР и ВНР). Они достаточны для работы со всеми типами электрифицированных пишущих машинок, кроме машинок с точечным механизмом печати. 2) *Среднескоростные* модемы перекрывают диапазон скоростей от 600 до 9600 бит/с, используя стандартные телефонные каналы. Наиболее часто используются

скорости 1200 и 2400 бит/с, что достаточно для работы с высокоскоростными печатающими устройствами последовательного действия. Примерами могут служить модем-1200 ЕС-8006 (ЧССР, ПНР), модемы-2400 ЕС-8010 (СССР), ЕС-8011 (ВНР и др.). 3) *Высокоскоростные* модемы (от 19,6 Кбит/с и выше) используют несколько телефонных каналов и позволяют подключать к ЭВМ дистанционно-перфоленточные и перфокарточные устройства вввода, АЦПУ и другие периферийные устройства среднего и высокого быстродействия.

4.9.1. Мультиплексоры передачи данных. Для более рационального использования аппаратуры передачи данных при работе в режиме разделения времени со многими абонентами эта аппаратура объединяется в так называемые *мультиплексоры передачи данных*. С одной стороны, такой мультиплексор подсоединяется к стандартному каналу ЭВМ, а с другой — через модемы — к линиям связи.

В качестве примера рассмотрим мультиплексор передачи данных ЕС-8400 (СССР). Он обеспечивает подключение до 15 линий связи. В их составе могут быть физические линии (провода), а также коммутируемые и некоммутируемые телеграфные и телефонные каналы. Скорость передачи по физическим линиям и телеграфным каналам — до 75 бит/с, по телефонным каналам — до 2400 бит/с. Входящие и исходящие соединения выполняются автоматически, за исключением исходящих соединений по коммутируемым телеграфным линиям связи. Наличие 2-канального переключателя позволяет мультиплексору работать поочередно с двумя различными ЭВМ.

Мультиплексор передачи данных ЕС-8401 (НРБ) при скорости 600 бит/с обеспечивает работу с 63 линиями связи, а при скорости 1200 бит/с — с 32 линиями. Диапазон скоростей, перекрываемый мультиплексором, 50, 75, 100, 200, 600, 1200, 2400 бит/с.

При наличии группы терминалов (абонентских пунктов), удаленных от ЭВМ, но близких друг к другу, для экономии линий связи употребляется *удаленный мультиплексор*, называемый иногда также *концентратором*. Он принимает информацию от терминалов и передает ее (по одному каналу связи) в режиме разделения времени в удаленную ЭВМ, а также распределяет по терминалам приходящую от ЭВМ информацию. Примером такого устройства может служить мультиплексор ЕС-8421 (ВНР).

4.9.2. Абонентские пункты. Абонентские пункты представляют собой определенные наборы вводно-выводных устройств (включая накопители на магнитных лентах и гибких дисках), предназначенные для так называемой *телеобработки*, т. е. обработки данных с помощью удаленной от них ЭВМ. Их можно использовать также для простой передачи данных. В состав абонентского пункта входят модем и аппаратура передачи данных (с блоком

защиты от ошибок), переговорно-вызывное устройство (обычный телефон) и собственно вводно-выводные устройства.

В абонентском пункте ЕС-8501 М (НРБ) в качестве таких устройств используются электрифицированная пишущая машинка ЕС-7177, а также устройства ввода и вывода с перфолент и карт с краевой перфорацией ЕС-6191 и ЕС-7191.

Устройство использует встроенный модем ЕС-8002. При автономном использовании он может готовить перфоленты и карты с краевой перфорацией или прочитывать их с документированием на пишущей машинке, а также реперфорировать информацию с лент или карт. При дистанционном использовании он может выполнять просто функции передачи и приема данных при работе с другими абонентскими пунктами, осуществлять обмен информацией с ЭВМ (через мультиплексор передачи данных), обеспечивать множественный режим передачи информации (один пункт передает, а остальные принимают).

Абонентский пульт ЕС-8550 (ВНР) построен на базе ЭВМ ЕС-1010, благодаря чему имеет широкий набор различных устройств. Помимо пишущей машинки и дисплеев, в него включаются АЦПУ, механизмы ввода и вывода на перфоленту, накопитель на постоянном магнитном диске, матричное печатающее устройство и устройство ввода с перфокарт. Скорость передачи данных 600 — 2400 бит/с в дуплексном и *полудуплексном* (с большой скоростью в одном направлении и с малой в обратном) режимах.

4.10. Автоматизированные рабочие места

АРМ представляют собой набор аппаратуры, соединяющей рабочие места различных типов с ЭВМ или с системой ЭВМ и помогающей работающим на этих местах людям лучше (быстрее и качественнее) выполнять свои функции. Состав АРМ определяется выполняемыми им функциями. Так, АРМ конструктора оборудуется графическим дисплеем со световым пером или даже тремя такими дисплеями (для работы с тремя проекциями объекта одновременно). В состав такого АРМ могут включаться графопостроители и графосчитыватели, электрифицированные пишущие машинки и другие вводно-выводные устройства. АРМ кладовщика оборудуется электрифицированной пишущей машинкой либо непосредственно, либо через линию связи, подключенной к ЭВМ.

При заполнении на этой машинке приемо-сдаточных ведомостей их содержимое передается на ЭВМ либо непосредственно, либо через промежуточный носитель (сейчас чаще всего перфоленту), изготовляемый автоматически в процессе печатания документов.

Рабочие места руководителей разных уровней оборудуются обычно алфавитно-цифровыми дисплеями с документирующими устройствами. Для высшего руководства и различного рода коллегий создаются специальные *ситуационные комнаты*, оборудованные большими экранами (управляемыми ЭВМ) для наглядного отображения запрашиваемой информации. Выпускаются специальные регистраторы производства для ввода в ЭВМ сменных заданий бригадам и отдельным рабочим и учетных данных об их выполнении. Специальной вводно-выводной аппаратурой оборудуются места оформления билетов на различные виды транспорта, кассы магазинов и т. д.

На рабочих местах у станков и другого оборудования устанавливаются простые *сигнализирующие* устройства, с помощью которых рабочие передают в ЭВМ данные о работе и неисправностях оборудования (и других помехах), не охваченные системой автоматического контроля.

Данные, собираемые с автоматизированных рабочих мест, через систему концентраторов-мультиплексоров поступают в центральную ЭВМ или непосредственно на ее внешние ЗУ. Через эту же систему передаются на рабочие места (оборудованные устройствами вывода) различного рода указания и распоряжения. Осуществление такого информационного обмена как раз и составляет сущность методов безбумажной информатики применительно к задачам организационного управления.

Большое разнообразие аппаратуры, устанавливаемой на автоматизированных рабочих местах, предъявляет большие требования к мультиплексорам, через которые эта аппаратура подсоединяется к ЭВМ. Таким требованиям в наибольшей степени удовлетворяют так называемые *программируемые мультиплексоры*. В них такие показатели интерфейса с АРМ, как форма передачи кодов (побитово, побайтово или пословно), а иногда и некоторые физические характеристики сигналов могут меняться путем изменения программ в памяти мультиплексора без какой-либо его переделки. Аналогично программным средствам могут меняться также характеристики интерфейса мультиплексора с ЭВМ. Тем самым без переделки мультиплексора его можно использовать совместно с ЭВМ различных типов.

К терминальному процессору (ТП) Барс могут подключаться разнообразные периферийные устройства: телетайпы, пишущие машинки, алфавитно-цифровые дисплеи, регистраторы производства, устройства контроля оборудования и пульты рабочих мест определенного вида. При последовательном подключении скорость обмена данных ТП с подключаемым устройством не должна превосходить 9600 бит/с, а при параллельном подключении 25 000 байт/с. Скорость обмена ТП Барс по подключаемым к нему коммутируемым каналам связи может колебаться от 50 до

4800 бит/с, а по некоммутируемым — от 50 до 9600 бит/с. При этом могут использоваться модемы любых типов. Процессор настраивается (путем смены блока постоянной памяти) на обмен с каналами различных ЭВМ и прежде всего, разумеется, с каналами ЕС ЭВМ. Всего он имеет 128 входных и 128 выходных шин. Встроенные усилители позволяют осуществить надежную передачу данных без модемов на расстоянии до 20 км.

Заметим, что функции программируемого мультиплексора может выполнять любая универсальная ЭВМ. Однако при использовании для этой цели, например, ЭВМ СМ-3 производительность (по сравнению с ТП Барс) упадет в несколько десятков раз, а стоимость возрастет в 3—4 раза.

4.11. Сбор данных с автоматических датчиков

В условиях высокоавтоматизированной технологии основная масса информации о производственных процессах поступает с автоматических датчиков. Такие датчики могут выдавать информацию как в аналоговой, так и в цифровой форме. Имеются стандарты для конкретных характеристик выдаваемых ими сигналов. Применительно к этим стандартам создаются так называемые *устройства связи с объектами* УСО. Их задача — преобразование стандартных сигналов, поступающих с датчиков, в стандартные сигналы канальных интерфейсов тех или иных ЭВМ. Такие УСО созданы, в частности, для машин СМ ЭВМ, одной из главных задач которых является управление сложными технологическими процессами. Для ввода в ЭВМ данных с измерительных приборов в сложных экспериментах используются УСО, построенные на базе крейтов системы Камак, о которых уже шла речь в гл. II.

4.12. Исполнительные механизмы

В автоматизированной технологии сигналы, поступающие из управляющих ЭВМ, должны выполнять те или иные действия (механические перемещения, включение и выключение тока и т. д.). Все эти действия выполняются специальными сервомоторами (непрерывными, шаговыми и др.), управляемыми ЭВМ. Эти приводы, называемые *исполнительными механизмами*, также поддаются стандартизации, а согласование их интерфейса с канальными интерфейсами ЭВМ осуществляется также с помощью УСО. Таким образом, УСО выступают как согласователи интерфейсов в двух направлениях: от объектов управления к управляющим ЭВМ и наоборот.

Глава V

ПРОГРАММИРОВАНИЕ И УПРАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

5.1. Методы адресации

Как уже указывалось выше (п. 1.11.1), код машинной команды (инструкции) разбивается на *код операции* и *адресную часть*. При первом взгляде на проблему адресации данных (операндов) наиболее естественной представляется *трехадресная система* адресации. Она предусматривает указание в каждой команде адресов двух операндов (слагаемых, сомножителей и др.), а также адреса, по которому в ЗУ засылается результат операции. На первом этапе развития вычислительной техники многие ЭВМ строились именно по такой системе.

Нетрудно видеть, однако, что трехадресная система малоэкономична. Не говоря уже о громоздкости кодов команд, она приводит, как правило, к значительному увеличению числа обращений к памяти. Действительно, арифметические операции при вычислениях по формулам выстраиваются в последовательные цепи. Например, пусть $q := a \times b + c$. При вычислении даже по такой простой формуле в случае трехадресной системы надо заслать результат d выполнения умножения $a \times b$ в ОЗУ и тут же извлечь его снова для выполнения сложения $d + c$. Таких ненужных обращений в ОЗУ не будет, если результат выполнения первой операции оставить в арифметико-логическом устройстве (АЛУ) центрального процессора.

Это достигается при уменьшении числа адресов в команде до двух и даже до одного. В случае наиболее распространенной в современных ЭВМ *одноадресной системы* в команде явно указывается адрес лишь одного операнда. Вторым операндом при этом обычно служит содержимое накапливающего сумматора (аккумулятора), а результат операции также остается в аккумуляторе. Поскольку адрес аккумулятора в команде явно не указывается, говорят, что имеет место так называемая *подразумеваемая* (скрытая в коде операции) или *неявная адресация*.

При подобной адресации в такой, например, программе, как программа суммирования нескольких чисел: $S = a_1 + a_2 + \ldots + a_n$, полностью устраняются лишние обращения в ОЗУ. Первой командой такой программы будет *очистка* сумматора (т. е. засылка в него нулевого кода), затем последует n команд сложения (прибавить a_i для $i = 1, \ldots, n$) и, наконец, в случае необходимости, последней командой (также одноадресной) можно заслать содержимое сумматора (сумму S) в любую требуемую ячейку ОЗУ.

Особенно большие удобства для неявной адресации и экономии числа обращений в ОЗУ представляет использование в АЛУ *стековой памяти* (п. 1.8.1). Например, вычисление по формуле $S := (a \times b) + (c \times d)$ можно при использовании стека выполнить таким набором команд:

заслать a	стек	a
умножить на b	»	$p = a \times b$
заслать c	»	c, p
умножить на d	»	$q = c \times d, p$
сложить	»	$p + q$

В результате в верхней ячейке стека останется число $S = p + q$, которое, как и в предыдущем примере, еще одной командой можно заслать в любую ячейку ОЗУ.

В случае наличия в АЛУ регистров или нескольких стеков при операциях над их содержимым указываются их номера, выполняющие роль адресов с очень *короткими кодами* (ввиду малости числа регистров и стеков). Чтобы не путать эти коды с адресами в ОЗУ, подобные (регистровые и стековые) операции выделяются в специальные команды. Ввиду малой длины кода в таких командах используется наряду с одноадресной также и двухадресная система. Возможны также и *смешанные двухадресные* операции, в которых один адрес относится к ОЗУ, а второй — к регистру или стеку.

5.1.1. Составная (условная) адресация. При адресации памяти большого объема код адреса может не уместиться в формат даже одноадресной команды. В таком случае, как уже указывалось выше (п. 1.11.1), может быть использована *составная* (условная) *адресация с базовым регистром*. Полный (составной) адрес получается при присоединении к коду *базы* (содержимого базового регистра) относительно короткого кода дополнительного адреса — так называемого *смещения*, указываемого в коде команды. База при этом образует старшие разряды полного адреса, а смещение — младшие.

Помимо укорочения кода адреса в команде, составная адресация обеспечивает так называемую *позиционную независимость*

программ. Дело в том, что в процессе пользования программой как она сама, так и относящиеся к ней данные могут загружаться в разные участки памяти. При *безусловной* адресации, т. е. при указании в командах *истинных* адресов данных или команд, подобные перемещения потребовали бы изменений адресов в командах программы. При условной адресации для этого достаточно лишь одного изменения базы. Тем самым программа делается независимой от позиции, занимаемой ею в памяти.

При обработке массивов обычно одни и те же операции применяются к различным элементам массива. Чтобы не повторять в программе одни и те же наборы команд, различающиеся лишь адресами операндов, эти наборы пишутся лишь один раз с неизменным (безусловным или условным) адресом. Необходимое же *сканирование* (перебор) элементов массива производится изменением особой части составного адреса, называемой *индексом*. Индекс запоминается в отдельном регистре (называемом в этом случае *индекс-регистром*) и, в соответствии с заданным шагом, меняется (с помощью специальных команд) в процессе выполнения цикла обработки массива. В этом случае полный (безусловный) адрес получается сложением трех адресов: базы, индекса и смещения.

5.1.2. Косвенная адресация. В ряде случаев оказывается удобным хранить в ячейке ЗУ адрес a_1, который указан в команде, и не значение операнда, а лишь адрес a_2 ячейки, из которой требуется выбрать (или в которую требуется записать) значение операнда. При этом адрес a_1 называется *адресом второго ранга*, а способ адресации — *косвенной адресацией*. Использовавшийся до сих пор способ прямого указания адреса операнда в команде носит наименование *прямой адресации*, а сам адрес — *адреса первого ранга*.

По аналогии с адресами второго ранга можно ввести адреса *третьего*, *четвертого* и любых последующих рангов. При использовании адреса n-го ранга извлечение значения операнда будет выполнено на n-м обращении в память. Иногда удобно использовать также адреса *нулевого ранга*. В этом случае в команде вместо адреса указывается непосредственно само значение операнда, так что его извлечение не требует никакого обращения к памяти. Подобный способ адресации носит наименование *непосредственной адресации*.

Непрямая адресация (косвенная или непосредственная) производится либо с помощью специальных команд, отличающихся особыми кодами операций, либо с помощью *особых признаков* в адресных частях команды.

Косвенная адресация оказывается очень удобной, когда требуется изменять адреса, в которых хранятся операнды, в процессе выполнения программы без изменения самой программы.

В частности, использование адресов второго ранга заменяет индекс-регистры ячейками ОЗУ, в которых хранятся прямые (1-го ранга) адреса операндов. Таким образом, в нашем распоряжении оказывается практически неограниченное число подобных псевдоиндекс-регистров, тогда как число истинных индекс-регистров всегда относительно невелико. Впрочем, косвенную адресацию можно применять (с помощью специальных команд) не только к ячейкам ОЗУ, но и к любым регистрам процессора.

Использование многих способов адресации увеличивает возможности программистов по *оптимизации* составляемых ими программ. Под оптимизацией здесь понимается улучшение таких характеристик программ, как *быстродействие* (за счет сокращения числа обращений в ЗУ), *объем* (за счет использования укороченных адресов), *релокабельность* (простота перемещаемости в памяти), а также простота и ясность понимания и работы с программой.

5.1.3. Адресация операндов переменной длины. В случае использования десятичной арифметики, а также при обработке алфавитной информации приходится иметь дело с операндами переменной длины, занимающими несколько ячеек ОЗУ. При адресации переменного операнда обычно указывается адрес его начала и длина операнда. Поскольку формат кода длины (как и всего адреса в целом) постоянен, то длина переменного операнда не может превышать некоторого (обычно достаточно большого) значения. В ЕС ЭВМ длина кода десятичных чисел ограничена 16 (десятичными) разрядами, а длина слов алфавитной информации — 256 символами (байтами).

5.2. Управление однопрограммным процессом

В простейшем случае процесс обработки данных в ЭВМ задается одной-единственной *прикладной программой*. Если при этом вся программа и данные, которыми она оперирует, помещаются в оперативную память, то управление процессом выполнения программы осуществляется в основном ею самою. Роль *системного программного обеспечения* (операционной системы) сводится в этом случае лишь к загрузке программы и данных в ОЗУ. Последовательность операций при этом такова: пользователь данной прикладной программы самостоятельно или с помощью оператора, обслуживающего ЭВМ, помещает программу и необходимые для нее данные (на тех или иных машинных носителях, например перфокартах) в одно из вводных устройств данной ЭВМ. Специальная программа ОС, называемая *загрузчиком* или *монитором*, получив от оператора с пульта ЭВМ имя вводного устройства и адрес A начальной ячейки ОЗУ, вводит в последовательные ячейки ОЗУ (начиная с адреса A) программу и данные. После

этого (обычно по команде с пульта) адрес первой команды программы передается в так называемый *программный счетчик* (счетчик команд), представляющий собой специальный регистр устройства управления (УУ) центрального процессора (ЦП). Теперь ЭВМ готова к исполнению программы.

По специальному сигналу с пульта управления ЭВМ начинает цикл исполнения первой команды. Прежде всего по адресу, указанному в программном счетчике, осуществляется выбор из ОЗУ кода первой команды. Этот код помещается в так называемый *регистр команд* УУ. По адресной части кода команды находятся необходимые данные, и происходит исполнение команды. В конце цикла исполнения меняется содержимое программного счетчика: в нем формируется (автоматически) адрес команды, которая подлежит исполнению в следующем цикле. Процесс формирования адреса происходит различными способами в зависимости от типа исполненной команды. Если это *команда перехода*, то в ее коде содержится код адреса следующей команды, который в этом случае просто засылается в счетчик. Различают команды *безусловного* и *условного перехода*. При безусловном переходе адрес перехода единствен.

Условный переход осуществляется на один из двух или нескольких адресов в зависимости от выполнения или невыполнения тех или иных *условий*, которые могут относиться к результатам операций, выполняемых как самой командой условного перехода, так и предшествующей ей командой. Первый случай иллюстрируется командой: «Если число в ячейке A больше числа в ячейке B, то перейти к команде по адресу C, в противном случае — к следующей по порядку команде». Во втором случае используется специальный малоразрядный (обычно не более двух разрядов) регистр ЦП, называемый *регистром признака*. Каждая выполняемая в ЦП команда устанавливает то или иное содержимое этого регистра. В команде же перехода указываются два или более адреса, засылаемые ею в программный счетчик в зависимости от содержимого регистра признака.

Специальным видом перехода является *переход по регистру* или *стеку возврата*. В регистр или стек специальной командой засылается адрес той или иной команды. По команде же перехода по регистру (стеку) возврата этот адрес направляется в программный счетчик. В случае использования стека в него можно засылать не один, а несколько адресов. Тем самым возникает возможность многократных переходов по адресам, последовательно выдвигаемым из стека.

В случае повторения одной команды в цикле ее код и адрес могут сохраняться в УУ до окончания цикла (определяемого тем или иным условием). Возможно также использование специальной команды «выполнить команду с адресом A», после выполне-

ния которой осуществляется переход на команду, следующую за ней (а не за командой по адресу A). Понятие команды, *следующей за данной*, означает команду, хранящуюся в ОЗУ непосредственно вслед за ней (в порядке возрастания адреса). Если команды занимают по одной ячейке ОЗУ, то при таком *естественном* следовании команд содержимое программного счетчика должно увеличиваться в каждом цикле на единицу. В более общем случае в ЦП вместе с кодом каждой команды запоминается код ее длины, который и прибавляется при естественном переходе к содержимому программного счетчика.

После заполнения программного счетчика, определяющего следующую подлежащую исполнению команду, осуществляется цикл ее выборки и выполнения, формирование нового содержимого программного счетчика, выполнение нового цикла и т. д. Окончание процесса происходит по команде *останов* (без выключения питания).

Вывод заключительной (выходной) информации в описанном простейшем случае предполагается запрограммированным в самой прикладной программе, так что к моменту останова машины все необходимые данные будут уже выведены с помощью устройств, указанных в прикладной программе.

5.2.1. Подпрограммы. Следует особо подчеркнуть, что благодаря наличию команд перехода и образованию циклов одни и те же команды написанной пользователем программы будут выполняться (вообще говоря, над разными данными) много раз. Благодаря этому длина программы, т. е. число команд в тексте программы, как правило, значительно меньше, чем число команд, фактически исполняемых ЭВМ при работе по этой программе. Особые удобства для уменьшения длины программ представляет использование так называемых *подпрограмм*.

В программах часто оказывается необходимым выполнять по многу раз последовательности одних и тех же операций. В вычислительных программах это могут быть, например, операции вычисления синуса, логарифма и других элементарных функций. Можно, разумеется, каждый раз, когда необходимо выполнить ту или иную повторяющуюся последовательность команд, фактически включать ее в текст программы. Однако при этом неоправданно увеличивается длина программы, что в свою очередь ведет к неоправданному увеличению количества используемой памяти.

Естественно поэтому выделить такую последовательность команд в отдельную программу и включить ее в рассматриваемую программу в качестве *подпрограммы*. Такое включение требует изменения естественного порядка следования команд. Для ухода на подпрограмму программист должен записать в программный счетчик адрес первой команды подпрограммы. Кроме того, в регистр (или стек) возврата засылается адрес той команды основ-

ной программы, к которой следует перейти после выполнения подпрограммы.

Подпрограмма должна при этом заканчиваться не командой останова, а командой перехода по регистру (стеку) возврата. Такой способ связи программы с подпрограммой необходим для обеспечения *независимости подпрограммы* от используемых ее программ. Подпрограмма при этом «не знает», где расположены вызывающие ее программы, и каждая программа сама, и только сама, несет ответственность за правильность обращения к используемым в ней подпрограммам. Текст каждой подпрограммы имеется лишь в одном экземпляре, независимо от того, какое число раз она используется. Более того, при использовании стека возврата имеется возможность *рекурсивного вызова* подпрограмм, т. е. использования в их тексте обращения к другим подпрограммам, в том числе и к ним самим.

Описанный прием обеспечивает независимость подпрограмм лишь в смысле порядка следования команд. Для полной их независимости необходимо обеспечить также их *независимость по данным* так, чтобы подпрограмма не требовала знания адресов используемых в ней данных основной программы. При простом, нерекурсивном обращении к подпрограмме такую независимость можно обеспечить, засылая с помощью команд основной программы необходимые данные в фиксированные (для данной подпрограммы) ячейки ОЗУ и возвращая подобным же образом результаты из таких ячеек в текст основной программы. При рекурсивном вызове наиболее удобно для подобных обменов данными использовать специальные стеки.

Заметим, что при отсутствии в ЭВМ реальной (физической) стековой памяти можно с помощью специальных программ операционной системы (или подпрограмм прикладных программ) образовать *виртуальные стеки* из фиксированных и даже переменных ячеек ОЗУ. Информация в таком стеке обычно не проталкивается из ячейки в ячейку. Вместо этого используется переменный указатель адреса начальной (верхней) ячейки стека в какой-либо фиксированной ячейке ОЗУ. Аналогичный прием зачастую используется и при образовании обычных (регистровых) стеков. При образовании стеков с переменными адресами (своими для каждой основной программы) доступ к ним из подпрограмм осуществляется с помощью косвенной адресации — адресов второго и даже третьего ранга, использующих фиксированные ячейки для помещения в них переменных адресов стеков. По аналогии с виртуальными стеками в ячейках ОЗУ могут формироваться также виртуальные регистры возврата, чем обеспечивается практически неограниченный их запас.

5.2.2. Библиотеки стандартных подпрограмм. Как следует из изложенного, каждый пользователь ЭВМ (и даже каждая от-

дельная программа пользователя) может создавать и использовать свой собственный набор подпрограмм. Наряду с этим ЭВМ снабжаются также набором *стандартных подпрограмм*, доступным для всех пользователей. Каждая из таких подпрограмм снабжается некоторым *фиксированным именем*, по которому пользователи могут вызывать требуемые подпрограммы, не зная их расположения в памяти.

Преобразование имени в адрес производится с помощью специальных программ операционной системы, использующих *машинный каталог*, аналогичный каталогам обычных библиотек. Операционная система автоматически организует также необходимые поля в памяти (или виртуальные стеки) для передачи данных между программами пользователей и вызываемыми ими подпрограммами. Набор стандартных подпрограмм с соответствующими частями ОС носит наименование *библиотеки стандартных подпрограмм* (БСП). Программное обеспечение таких библиотек осуществляет также автоматическое перемещение вызываемых подпрограмм из внешней памяти в ОЗУ. Благодаря этому БСП могут наращиваться практически неограниченно.

При составлении стандартных подпрограмм (равно как и любых программ длительного пользования) по возможности стремятся достичь большей их общности. Это означает, что, например, подпрограмма для решения алгебраических уравнений составляется применительно к уравнению произвольной степени n. Вместе с тем, если тот или иной частный случай (например, решение квадратного уравнения) достаточно часто встречается на практике и может решаться более быстро и эффективно программой частного вида, то такая программа также включается в библиотеку стандартных подпрограмм.

5.3. Модули загрузки и мультипрограммирование

Описанный в § 5.2 случай, когда рабочая программа вместе с данными полностью помещается в ОЗУ, представляет собой скорее исключение, чем правило. Поэтому и управление вычислительным процессом на практике выглядит существенно сложнее. Большая программа, не помещающаяся целиком в ОЗУ, делится на отдельные блоки, загружаемые в ОЗУ и исполняемые по очереди. Эти блоки, называемые также *модулями загрузки*, будучи записаны на те или иные машинные носители (с помощью устройств подготовки данных), помещаются во вводные устройства ЭВМ. Загрузчик (монитор) переписывает первый модуль в ОЗУ, после чего он исполняется так, как это было описано в предыдущем параграфе. По окончании этого процесса управление вновь передается загрузчику, который загружает в ОЗУ

следующий модуль, и так далее — до полного исчерпания программы.

Заметим, что благодаря *нелинейной* (циклической) структуре большинства программ, вызванной наличием команд перехода, уже выполненные модули могут вновь и вновь вызываться в ОЗУ для повторного исполнения. В случае, когда рабочая программа записана на перфокартах, это создает определенные неудобства и даже требует вмешательства оператора для перемещения уже считанных перфокарт из выходного во входной карман считывающего устройства. Поэтому, если даже первоначальный ввод делается с перфокарт, удобно создать копию рабочей программы на машинных носителях (лентах или дисках).

При *произвольном чередовании модулей* для всякого изменения естественного порядка следования модулей загрузчик должен получать физический адрес очередного модуля. В простейшем случае такая информация может быть передана ему самой рабочей программой. Однако при этом от программиста — автора рабочей программы — требуется знание истинного размещения модулей во внешней памяти. В современных ЭВМ планированием такого размещения занимается специальная *программа-диспетчер* (планировщик). Она же составляет таблицы соответствия *имен модулей* (условных адресов) и их реальных (физических) адресов. Программе-диспетчеру поручается также планирование размещения программных модулей (и соответствующих данных) в ОЗУ, закрепление за программой необходимых вводно-выводных устройств и вообще планирование и управление использованием различных *ресурсов*, которыми обладает ЭВМ.

Наряду с загрузчиком и диспетчером-планировщиком в операционные системы современных ЭВМ входят и другие программы. Порядком их выполнения и взаимодействия с прикладными программами (программами пользователей) руководит включаемая в ОС в качестве «верховного распорядителя» специальная программа, называемая обычно *супервизором*.

В состав системного программного обеспечения современных ЭВМ, как правило, включается специальная *макрокоманда* (подпрограмма) *обращение к супервизору*, которая может использоваться как в прикладных программах, так и в программах ОС. Исполнение этой макрокоманды вызывает так называемое *прерывание* в работе ЭВМ, которое может возникать и по другим (перечисляемым ниже) причинам. Во всех случаях в первой стадии отработки прерывания происходит «упрятывание» (целиком или частично) в определенные заранее ячейки ОЗУ состояния центрального процессора. Такое упрятывание должно обеспечить после отработки прерывания возможность возвращения к тому месту прерванной программы, в котором произошло прерывание. В наиболее общем случае бывает нужно упрятать в ОЗУ не только

содержимое счетчика команд, но и содержимое всех регистров и стеков центрального процессора.

Помимо обращения к супервизору, причиной прерывания может стать сигнал от средств схемного и программного контроля за правильностью вычислительного процесса, а также сигналы от периферийного оборудования и других внешних (по отношению к центральному процессору) сигналов. К их числу относятся специальные сигналы с пульта управления ЭВМ (подаваемые оператором), сигналы, поступающие от встроенных в ЭВМ электронных часов — так называемого *таймера*, сигналы со специальных *шин прямого управления* (с помощью которых в процессор передается информация из особо быстродействующих внешних источников, например от других процессоров).

При общности первой стадии (упрятывания в ОЗУ состояния ЦП) обработка второй стадии прерывания производится различными программами (зависящими от вида прерывания). Программы обработки прерываний также входят в состав ОС.

Наличие прерываний и программ их обработки позволяет реализовать в ЭВМ так называемый *мультипрограммный режим* работы. Суть его состоит в том, что во входные устройства ЭВМ (или в устройства внешней памяти) помещается сразу несколько прикладных программ с соответствующими наборами входных данных. Этим программам приписываются те или иные *приоритеты*. Супервизор ОС выбирает из готовых для исполнения программных модулей один, принадлежащий программе наивысшего приоритета. Готовность к исполнению означает, во-первых, что соответствующему модулю передано управление содержащей его программой, а во-вторых, что как сам модуль, так и необходимая для его выполнения (хотя бы частичного) информация находятся в ОЗУ. Выполнение модуля происходит либо до полного его окончания (после чего управление передается супервизору для организации исполнения следующего модуля), либо до прихода сигнала прерывания, которое может быть вызвано прежде всего тем, что для дальнейшего выполнения модуля требуется обмен данными с периферийными устройствами (ВЗУ или вводно-выводным). Но столь же возможны и любые другие причины прерывания.

При наличии независимых от ЦП канальных процессоров обмен данными между ОЗУ и периферией может происходить одновременно с работой ЦП по исполнению очередного программного модуля. Окончив передачу данных в ОЗУ, канал формирует сигнал прерывания, по которому может возобновиться выполнение ранее прерванного (ввиду отсутствия данных) программного модуля. Таким образом, хотя ЦП в каждый данный момент работает лишь с одной программой, ЭВМ в целом может осуществлять одновременно подготовительную работу и с другими про-

граммами. Диспетчер-планировщик при этом должен осуществлять распределение ресурсов (прежде всего свободных областей ОЗУ) для всех исполняемых ЭВМ программ.

5.3.1. Маскирование прерываний. В ряде случаев прикладной программист может быть заинтересован в том, чтобы некоторые виды прерываний на том или ином участке его программы не выполнялись. Например, при выполнении арифметической операции ее результат может выйти из предписанного формата, так что какие-то разряды этого результата могут быть утеряны. В обычном случае при таком *переполнении формата* возникает прерывание. Однако не исключено, что программист предвидел такую возможность, а построенная им программа не нуждается в утерянных разрядах (например, в сотых долях копейки при финансовых расчетах). В таком случае программисту предоставляется возможность с помощью специальной команды *маскировать* сигналы прерывания заданного вида. В результате выполнения этой команды в один из разрядов специального *маскирующего регистра* ЦП записывается единица. По истечении необходимости маскирования оно может быть отменено (прикладным программистом) с помощью специальной *команды демаскирования.*

Для обеспечения возможности эффективного управления вычислительным процессом со стороны ОС некоторые сигналы прерывания (прежде всего требующие вызова супервизора) маскированию не подлежат и соответствующие средства в руках прикладных программистов отсутствуют.

5.3.2. Слово состояния программы. В первых моделях ЭВМ текущее состояние устройства управления (УУ) центрального процессора полностью определялось содержимым регистра команд и счетчика команд. Позже к ним добавилось содержимое регистра признака результата выполнения предыдущей команды. Развитие ОС добавило сюда еще содержимое регистра маскирования и регистра для хранения кода прерывания. Кроме того, для определения очередного приращения счетчика команд (при их естественном следовании) может не только вычисляться, но и запоминаться в специальном регистре код длины исполняемой команды. Содержимое всех регистров УУ центрального процессора (обычно без содержимого регистра команд) принято называть *словом состояния программы.* При прерываниях это слово запоминается в специально отведенных для этой цели ячейках ОЗУ.

5.3.3. Управление заданиями и управление задачами. В практике применения ЭВМ часто случается, что одни и те же программы могут многократно (быть может, в различных последовательностях) использоваться для обработки различных данных. В этом случае оказывается целесообразным, присвоив программам (или программным модулям), а также массивам данных те или иные *имена*, описывать сокращенно различные процедуры обра-

ботки данных, пользуясь только этими именами. Такие сокращенные описания принято называть *заданиями*, в отличие от *задач*, представляющих собой соединение *полных текстов* (кодов) соответствующих программ с *полными наборами значений* соответствующих исходных данных.

Заметим, что отдельно друг от друга программы и данные еще не составляют задачи. Она возникает только при их соединении. Вводя в ЭВМ раздельно различные программы и различные наборы исходных данных, мы получаем возможность при помощи *языка управления заданиями* (ЯУЗ) формировать различные задачи. Ввод и анализ заданий, а также фактическое формирование описанных с их помощью задач (приведение в соответствие программ и данных) производятся специальными программами ОС, выполняющими функцию управления заданиями.

После «стыковки» программ и данных производится уже описанный выше процесс выполнения программ. Управление этим процессом носит название *управления задачами*. Его сущность (как в однопрограммном, так и в мультипрограммном режимах) была уже описана выше.

5.3.4. Пакеты программ. Заготавливая заранее именованные программные модули, направленные на решение того или иного класса задач (например, на решение систем алгебраических уравнений), и вводя имена и формы представления для массивов данных (например, множеств коэффициентов уравнений), мы получаем возможность программировать любые задачи данного класса на языке управления заданиями, т. е. в весьма сокращенном и удобном виде. Возникающая таким образом заготовка (вместе с соответствующей системой управления заданиями) носит наименование *пакета программ*. Если ОС ЭВМ, для которой составляется пакет, не содержит средств управления заданиями (или если они полностью не устраивают программиста), то такие средства включаются в состав пакета.

С помощью пакетов прикладных программ можно перекрывать целые области человеческой деятельности (бухгалтерский учет, планирование работы предприятий, проектно-конструкторские расчеты в той или иной области и др.). В таком случае вместо программирования любой конкретной задачи из соответствующей области достаточно написать лишь *задание* на ее решение, т. е. имена программ и данных, которые нужно привести в действие для получения решения поставленной задачи.

Иногда *управляющая часть* (специализированная ОС) пакета программ представляет дополнительные возможности для упрощения пользования этим пакетом. Можно, например, включить в управляющую часть пакета специальные программы для анализа исходных данных и *автоматического выбора* программных мо-

дулей, наиболее подходящих для решения требуемой задачи именно при этих исходных данных. В случае подобных пакетов (называемых *интеллектуальными*) программист в задании указывает не имена конкретных программ, а имена множеств таких программ, решающих одну и ту же задачу, но различными методами.

5.4. Разделение времени

При обычном мультипрограммном режиме работы скорость решения задач в значительной степени определяется тем, насколько удачно расположены их программные модули и блоки данных во внешней памяти. Поскольку в современных ЭВМ такое расположение выполняется, как правило, автоматически, часто бывает затруднительно предсказать заранее срок решения задач даже высоких приоритетов. Можно сказать, что мультипрограммирование направлено на увеличение пропускной способности ЭВМ по потоку задач в целом, а не на ускорение решения отдельной наперед заданной задачи. Этот недостаток становится особенно заметным, когда на ЭВМ решается одновременно большое число относительно мелких задач. Для такого случая лучший результат дает *режим разделения времени*, уже упоминавшийся выше.

Для эффективности подобного режима с помощью специальных программ ОС организуется *виртуальная* (воображаемая) оперативная память, объем которой складывается из объема реального ОЗУ и объема ВЗУ (во всяком случае — полного объема памяти на магнитных дисках). Каждая программа получает свою область подобного воображаемого ОЗУ, на которой полностью помещаются как сама программа, так и все данные, которыми она оперирует. Кроме того, каждая программа получает свою элементарную порцию времени, по истечении которой по сигналу от электронных часов производится прерывание данной программы и передача управления следующей по порядку программе.

Подобный процесс повторяется циклически, так что в течение полного цикла каждая программа продвигается в своем исполнении в соответствии с выделенной для нее порцией времени. Правда, при этом требуется заблаговременно перебросить нужные страницы виртуальной памяти в реальное физическое ОЗУ. Благодаря цикличности подобного процесса в принципе (при наличии независимых канальных процессоров и знании характера решаемых задач) можно построить такую *дисциплину обслуживания*, при которой будет осуществляться упреждающая подготовка нужных страниц.

На практике при случайных потоках задач применяют простейшие дисциплины обслуживания, что, как правило, приводит

к дополнительным задержкам на информационные обмены. Тем не менее режим разделения времени во многих случаях гарантирует решение задач в заданные сроки, и поэтому он особенно охотно применяется в системах диалога с пользователями, где требуется гарантированная верхняя оценка *времени реакции* системы (т. е. времени, требуемого для получения ответов на задаваемые ей вопросы). С таким требованием приходится встречаться, например, в справочно-информационных системах с большим числом пользователей (см. гл. VI).

5.5. Мультипроцессирование

В многопроцессорных вычислительных системах управление вычислительным процессом усложняется за счет необходимости организовать рациональную загрузку всех центральных процессоров системы. Диспетчер-планировщик при этом рассматривает ЦП как новый вид ресурса, который закрепляется и своевременно перераспределяется между различными программными модулями и различными потоками данных. Наиболее просто этот вопрос решается в том случае, когда каждый ЦП решает свою собственную независимую задачу. Если несколько ЦП используются для решения одной и той же задачи, то должно производиться *распараллеливание* вычислительного процесса.

Необходимо прежде всего отметить, что распараллеливание допускают не все процессы. В тех случаях, когда оно оказывается возможным (а таких случаев — большинство), различают два основных типа распараллеливания. Первый тип использует для всех процессоров один и тот же поток команд, но различные потоки данных. Например, при сложении двух матриц можно разделить между процессорами строки этих матриц и выполнить параллельно сложение различных пар строк с помощью одного и того же программного цикла. Второй тип распараллеливания использует для каждого процессора свой программный модуль (много потоков команд), выполняемый над одними и теми же или различными данными. В этом случае обычно программист указывает, какие программные модули могут исполняться независимо друг от друга. Используя эти указания, ОС мультипроцессорной системы с помощью таблицы занятости процессоров (ведущейся программой-диспетчером) распределяет максимально возможное число параллельно исполнимых модулей между свободными процессорами.

Поскольку разные модули исполняются, как правило, за разное время, ОС принимает на себя задачу их *синхронизации*. Смысл ее состоит в том, что очередной программный модуль запускается в работу лишь после того, как для него подготовлены (в результате работы предшествующих модулей) все необходи-

мые исходные данные. Подобная взаимозависимость модулей, задаваемая в простейшем случае программистом, определяет одновременно и возможность параллельного исполнения модулей. Более интеллектуальные мультипроцессорные ОС могут полностью или частично освобождать программиста от этой работы, выполняя анализ программы и выделение модулей для параллельного исполнения в автоматическом режиме.

В большинстве современных универсальных мультипроцессорных систем программы ОС могут исполняться любым ЦП, который в этом случае берет на себя роль своеобразного дирижера работой всех остальных ЦП. Возможно, однако, и такое решение, в котором такая задача возлагается на специальный *управляющий процессор* или даже *иерархию* таких процессоров. В первом случае обеспечивается бóльшая надежность и гибкость системы, во втором — бóльшая простота и ясность организации управления.

5.6. Управление данными

Когда необходимые для программы данные находятся в ОЗУ, каждая команда программы (с помощью ЦП) сама обеспечивает себя необходимыми данными. Если же такие данные находятся во внешней памяти (ВЗУ), то обращение к ним обычно производится с помощью специальных программ ОС. Главная задача этих программ заключается в том, чтобы по имени или месту в файле той или иной порции данных (обычно отдельной записи) найти фактическое местоположение требуемой порции по ВЗУ и передать ее на специально выделенную (программистом) для этой порции *рабочую область* исполняемого программного модуля в ОЗУ.

Работа таких программ инициируется макрокомандой «открыть файл» (англ. open file). Используя данные, сообщаемые программистом с помощью специальной макрокоманды *описания файла*, макрокоманда «открыть файл F» готовит файл (или периферийное устройство) с именем F к обмену с ОЗУ. С этой целью в ОЗУ выделяется *буферная зона*, которой предстоит обмениваться данными с названным файлом. Обмен осуществляется обычно не отдельными записями, а блоками, содержащими по нескольку записей. В случае файлов на ленте тем самым экономятся участки ленты для ее разгона и остановки, которые необходимо было бы оставить между записями при их раздельной передаче. Впрочем, обмен блоками оказывается целесообразным для всех последовательно просматриваемых файлов, независимо от носителя, на котором этот файл расположен. Режим обмена (включая сведения о размерах требуемых буферов) устанавливается макрокомандой описания файла.

По макрокоманде «открыть файл» требуемый файл ищется на носителе, причем в случае последовательно просматриваемого файла соответствующее периферийное устройство устанавливается в его начало и начинает обмен первым блоком. Возможна также (если носитель с файлом не установлен ни на одно из устройств) выдача информации на системный пульт человеку-оператору для поиска в библиотеке и установки на указанное устройство носителя с требуемым файлом.

По другой макрокоманде «взять F» (англ. get F) очередная порция информации (как правило, отдельная запись) из буфера, соответствующего файлу F, перемещается в *рабочую область* программы. Эта область может быть либо заранее определена в рабочей программе и сообщена операционной системе, либо адресована к макрокоманде «взять F, A» с помощью явно указываемого адреса A. В случае обращения к записи по ее имени или *ключу* (см. гл. VI), а не в порядке ее расположения в буфере, в макрокоманде «взять» должны быть указаны необходимые сведения, причем в случае отсутствия требуемой записи в буфере (в частности, при исчерпании буфера с последовательным просмотром) инициируется обмен буфера с ВЗУ новым блоком.

Выбор конкретных областей ВЗУ, в которые записываются данные, осуществляется диспетчером-планировщиком в результате анализа соответствующего задания. В момент же загрузки данных осуществляется то или иное их конкретное представление — так называемая *физическая структура данных*, в которую могут входить дополнительные средства, облегчающие поиск нужных порций информации (каталоги, указатели и др.). Более подробно о физических структурах данных и способах поиска рассказывается в следующей главе.

Обратная операция отсылки в буфер обработанных прикладной программой записей осуществляется макрокомандой «поместить F» (англ. put F). Здесь через F обозначено имя файла, которому принадлежит данный буфер. Как и в макрокоманде «взять», перемещаемая в буфер запись берется либо из рабочей области (неявно адресуемой) исполняемой прикладной программы, либо из какой-либо прямо адресуемой (макрокомандой «поместить F, A») области ОЗУ. В последовательно просматриваемых файлах запись в буфер происходит в порядке очередности поступления перемещаемых записей. В более сложных случаях порядок размещения определяется режимом обмена, устанавливаемого макрокомандой описания файла. При отсутствии места в буфере для перемещаемой записи автоматически инициируется необходимый групповой обмен с ВЗУ.

Макрокоманда «закрыть файл F» освобождает устройство, на котором он был помещен, равно как и области ОЗУ, занятые под соответствующие буферы, для другого использования. Заметим,

что во многих современных ОС допускается работа (в режиме автоматического переключения) с файлами, размещенными на нескольких периферийных устройствах. Необходимо отметить, что при организации буферов обычно для одного файла открывается не один, а по крайней мере два буфера. Пока с одним из них работает основная программа, другой осуществляет обмен с ВЗУ (через канальный процессор). В ряде случаев удобно организовать *пул* (специальный массив) из многих буферов с различными режимами заполнения и использования. Помимо собственно записей файла, такие буферы могут служить для помещения в них дополнительной справочной информации, облегчающей поиск нужных записей (по имени или ключу). Для образования буферов и буферных пулов могут использоваться специальные макрокоманды.

Соответствие между именами и физическими адресами (во ВЗУ) перемещаемых порций информации называется *интерфейсом* между логической и физической структурами данных. В описанном простейшем случае такой интерфейс индивидуален для каждой задачи и устанавливается с помощью средств ОС во время загрузки данных в ЭВМ. В общем случае различные программы могут пользоваться общими *базами данных*. Возникающие здесь проблемы управления данными разбираются в следующей главе.

5.6.1. Программы канала. При наличии канальных процессоров команды обмена с периферией (включая обмен с ВЗУ) сводятся по существу к трем видам команд: «начать обмен», «окончить обмен», «опросить состояние» (канала или устройства). При этом в команде, кроме кода операции, указывается имя (номер) устройства и канала, с которыми будет производиться операция (одно и то же устройство может быть подключено к нескольким каналам). После выполнения подобной *обменной команды* центральный процессор передает управление обменом поименованному в ней каналу, а сам освобождается для работы с другими программными модулями (вызываемыми супервизором, которому передается управление в ЦП).

В простейшем случае *программа канала* может состоять из одной-единственной команды. *Код операции* в команде обмена определяет прежде всего направление обмена (чтение или запись). Кроме того, попутно с обменом могут выполняться и какие-либо дополнительные операции (например, сортировка читаемых перфокарт по двум выходным карманам). Операции обмена могут выполняться при прямом и обратном движении магнитной ленты. Эти и другие дополнительные признаки операции также указываются в ее коде.

Наряду с командами обмена имеются также различные *команды управления*. Некоторые из них (как, например, коман-

да обратной перемотки ленты и др.) передаются через канал на исполнение контроллерам соответствующих устройств, другие могут исполняться самим каналом.

Кроме кода операции в команде обмена указывается также адрес первой ячейки ОЗУ, с которой начинается обмен, количество передаваемых байтов, а также некоторые признаки. При исполнении команды ведется счет числа переданных байтов, и при совпадении его с заданным выполнение команды заканчивается. Специальный признак в коде команды указывает, нужно ли закончить ею выполнение программы канала или передать управление следующей по порядку команде. При окончании программы канала формируется сигнал прерывания.

Кроме кода команды в УУ канала обычно хранятся в специальных регистрах коды состояния канала и состояния устройства, с которым работает канал. Эти состояния передаются в ЦП по специальным командам (опроса канала или опроса устройства) и анализируются исполняемой ЦП программой на предмет выбора канала или устройства для дальнейшей работы.

Программы канала включаются в качестве подпрограмм в основную программу, разбиваясь по соответствующим модулям загрузки. Команды таких подпрограмм обычно хранятся в последовательных ячейках ОЗУ (выбираясь одна за другой каналом). В некоторых машинах (в том числе в ЕС ЭВМ) возможно также хранение команд программ канала в произвольных ячейках ОЗУ. В этом случае, разумеется, должны быть указаны адреса переходов на соответствующих ячейках.

В ЭВМ, у которых специальные канальные процессоры отсутствуют, программы обмена осуществляются центральным процессором. При этом используются сигналы готовности устройств к обмену очередной порцией информации. Эти сигналы анализируются либо непосредственно рабочей (прикладной) программой, либо (чаще всего) специальными программами ОС, осуществляющими *синхронизацию* обменных процессов с процессом исполнения рабочих программ.

5.7. Автоматизация программирования

Программирование непосредственно в машинных языках представляет собой весьма трудную задачу. Прежде всего, сам по себе язык машинных команд является непривычным для человека и требует достаточно длительной подготовки для свободного владения им. Дополнительные трудности вносит использование двоичного представления кодов операций и адресов в командах. Практика показывает, что при программировании непосредственно в машинных кодах даже весьма опытные квалифицированные

программисты делают много ошибок, процесс программирования длится достаточно долго.

Поэтому с самого начала широкого использования ЭВМ прилагалось (и продолжает прилагаться) много усилий для *автоматизации процесса программирования*. Основой для такой автоматизации является использование *языков программирования* (или, как их еще называют, *входных языков* ЭВМ), более приспособленных к человеческому восприятию и потому более легких для освоения и использования. Поскольку написанная на любом из таких языков программа не может быть непосредственно выполнена ЭВМ, ее необходимо транслировать в машинное представление. Такая трансляция выполняется специальными *программами-трансляторами*, которые в зависимости от вида используемого входного языка называются либо *ассемблерами*, либо *компиляторами*.

Другой способ исполнения программы, написанной в любом языке, отличном от машинного языка используемой ЭВМ, представляет собой так называемая *интерпретация*. При интерпретации специальная *программа-интерпретатор* просматривает операторы программы во входном языке, формирует (из машинных команд данной ЭВМ) программные блоки (по возможности более короткие), выполняющие работу этих операторов, и организует их немедленное исполнение (до окончания полного просмотра введенной программы).

Время t_1 исполнения интерпретируемой программы, как правило, больше, чем время исполнения t_2 транслированной программы. Однако во втором случае следует иметь в виду достаточно большие разовые затраты времени T_0 на трансляцию программы. Если программа исполняется n раз, то суммарное время, затрачиваемое на ее исполнение в случае интерпретации, равно nt_1, а в случае трансляции $nt_2 + T_0$. Поскольку $t_1 > t_2$, то при $n < T_0/(t_1 - t_2)$ более выгодной будет интерпретация, а при $n > T_0/(t_1 - t_2)$ — трансляция. Поэтому при многоразовом употреблении рабочих программ в неизменном виде более экономичным является процесс трансляции. Если же программа используется небольшое число раз или же подвержена частым изменениям, то преимущество оказывается на стороне интерпретации.

Еще одно преимущество интерпретации проявляется в задаче *оптимизации* программ с целью уменьшения времени их выполнения. Дело в том, что при трансляции далеко не всякая оптимизация программы на входном языке приведет к реальной оптимизации рабочей программы. Поэтому в случае трансляции программист в значительной мере лишен возможности помочь процессу оптимизации рабочих программ и эта задача практически полностью должна быть возложена на транслятор. Однако по-

добные *оптимизирующие трансляторы* при достаточно высоких уровнях оптимизации сложны и к тому же требуют для выполнения процесса трансляции гораздо больше времени, чем простые (неоптимизирующие) трансляторы.

При замене одной ЭВМ другой (отличающейся от нее системой команд) нередко возникает задача интерпретации программ старой машины в новой. Процесс такой интерпретации получил специальное наименование *эмуляции*, а осуществляющая его программа — наименование *эмулятора*. Достаточно часто возникает также задача трансляции на одних машинах (обычно достаточно мощных) программ для других машин (чаще всего для мини и особенно для микро-ЭВМ). Такой процесс получил наименование *кросс-трансляции*.

5.8. Машинно-ориентированные языки программирования

Входные языки, включающие в наборы своих операторов операции, выполняемые командами той или иной ЭВМ, принято называть *машинно-ориентированными языками*, языками ассемблерного типа, или иногда *автокодами*. Трансляторы с таких языков называются *ассемблерами*.

5.8.1. Мнемокоды. Простейший язык ассемблерного типа получается в результате простой замены в кодах машинных операций той или иной ЭВМ двоичных кодов операций более привычными для человека буквенными или буквенно-цифровыми кодами с десятичными цифрами. Кроме того, двоичные коды адресов в командах заменяются десятичными (или иногда восьмеричными) кодами. Трансляция с такого языка, называемого *мнемокодом*, осуществляется весьма просто, сводясь по существу к простой перекодировке. Тем не менее использование даже столь простого входного языка намного упрощает программирование и уменьшает количество ошибок.

5.8.2. Символические автокоды. При использовании простых мнемокодов программист использует истинные адреса, предопределяя тем самым фактическое расположение программы и данных в памяти ЭВМ. Для сложных программ и систем данных держать в памяти такое расположение достаточно сложно. Поэтому количество ошибок при использовании мнемокодов в сколько-нибудь сложных случаях продолжает оставаться недопустимо большим. Особенно велика вероятность таких ошибок при внесении изменений в уже составленные программы. Типичная ошибка — забывание изменения адресов в командах перехода при вставках или удалении каких-либо групп команд из программы.

Выход из положения дает использование так называемых *символических адресов*, в качестве которых могут обычно упо-

требляться любые буквенно-цифровые коды, начинающиеся буквой (например, *MX*5, *abd* и т. д.). Символические адреса выполняют роль переменных в обычной алгебре: они представляют собой как бы абстрактные (не привязанные ни к какому конкретному ЗУ) ячейки, в которые могут помещаться те или иные конкретные значения отождествляемой с ними переменной величины. Число таких ячеек не ограничено, так что программист не обязан их экономить в отличие от ячеек реальных ЗУ. Важно лишь, чтобы в одну ячейку не помещалось несколько различных значений одновременно. Достичь этого, не преследуя целей экономии ячеек, относительно просто: достаточно каждому новому значению присваивать новый символический адрес. Таким образом, число ошибок при символической адресации резко уменьшается.

Ассемблер, транслирующий программу с языка символического автокода, содержит специальный блок, присваивающий символическим адресам значения истинных адресов ячеек реального ЗУ данной ЭВМ. При этом автоматически решается *задача экономии ячеек* за счет использования уже отработавших ячеек для хранения новых значений. Автоматизм подобной процедуры делает ее (в отличие от ручного исполнения) практически безошибочной: ошибки могут возникнуть лишь в случае сбоев в работе ЭВМ. В условиях развитых систем контроля в современных ЭВМ подобные сбои, как правило, регистрируются, а их последствия автоматически устраняются (в результате повторного счета или другими путями, например использованием так называемых кодов с исправлением ошибок).

5.8.3. **Макроассемблерные языки.** Следующий шаг в совершенствовании машинно-ориентированных языков делается путем включения в их состав наряду с обычными машинными операциями так называемых *макрооператоров* (или, сокращенно, просто *макро*). Ассемблер переводит обычные операторы автокода в отдельные машинные команды, а макрооператоры — в последовательности команд. Такая последовательность обычно не конструируется ассемблером заново, а берется из библиотеки стандартных подпрограмм, именованных названиями соответствующих макрооператоров. Задача ассемблера состоит в том, чтобы вмонтировать нужные подпрограммы в тело вырабатываемой им машинной программы *).

Обычно в качестве макрооператоров выступают операторы вычисления элементарных функций (синус, логарифм и др.). К числу макрооператоров могут быть отнесены также рассмот-

*) Термин «ассемблер» можно перевести на русский язык как «собиратель» или «монтировщик».

ренные выше *операторы манипулирования с данными* во ВЗУ (get, put и др.).

При достаточно больших наборах макрооператоров программирование в соответствующих макроассемблерных языках может сильно упрощаться, поскольку даже весьма длинные программы могут быть записаны в виде относительно коротких последовательностей во входном языке. В таком случае основная часть работы по составлению машинных программ ложится уже не на программиста, а на систему автоматизации программирования. К сожалению, однако, невозможно обеспечить себя заблаговременно набором макрооператоров на все случаи жизни. Поэтому для облегчения программирования наряду с макроассемблерными языками используются и другие средства.

5.9. Машинно-независимые языки

Машинно-независимые языки, как следует из самого их названия, не ориентированы на конкретные ЭВМ. Операторы и логические условия, которыми они оперируют, определяются над свободными от особенностей машинной реализации *логическими* структурами данных. Сами операторы и условия не укладываются в «прокрустово ложе» строго определенных машинных форматов. В таких языках, называемых также *входными языками высокого уровня*, широко используется привычная для человека формульная символика и другие средства, ориентированные в первую очередь на человека, а не на машину.

Огромным преимуществом языков высокого уровня является то, что написанные на них программы могут исполняться на различных ЭВМ, снабженных трансляторами (называемыми в этом случае обычно *компиляторами*) или интерпретаторами с этих языков. За это преимущество, правда, приходится расплачиваться большей сложностью трансляторов (по сравнению с трансляторами с автокодов). Самое же главное состоит в том, что, изучив один из наиболее распространенных языков высокого уровня, человек может пользоваться потом любыми ЭВМ, не вдаваясь в специфику их систем команд. Процесс программирования также при этом обычно существенно облегчается. Правда, такое облегчение имеет место главным образом для *прикладных программ*, особенно тех, в которых основное место отводится *вычислительным операциям*.

Для так называемых *системных программ*, главная задача которых — не выполнение расчетов или других сложных преобразований внешней (по отношению к ЭВМ) информации, а *организация вычислительного процесса*, большинство языков высокого уровня теряет свою эффективность. В этом случае (включающем в себя программы ОС, системы автоматизации программи-

рования и различного рода служебные программы для переупорядочивания и других изменений форм представления данных) предпочитают, как правило, машинно-ориентированные языки. Они дают больше возможностей для (человеко-машинной) оптимизации системных программ, что особенно важно именно для таких программ в силу большой многократности их использования.

Среди языков высокого уровня различают процедурно- и проблемно-ориентированные языки. К *процедурно-ориентированным* принято относить алгоритмически универсальные языки, в принципе пригодные для программирования любых задач, хотя обычно и они имеют определенную (но обязательно достаточно широкую) проблемную специализацию, например специализацию на процедуры вычислительного характера. *Проблемно-ориентированные языки*, как правило, специализируются на описании процедур достаточно узкой предметной области (например, процедур управления фрезерными станками) и не обязательно обладают алгоритмической универсальностью. Заметим, что граница между проблемно- и процедурно-ориентированными языками очерчена не очень четко.

Дадим теперь беглое описание наиболее распространенных языков высокого уровня. При этом мы не преследуем цели описать все средства таких языков и тем более научить читателя пользоваться этими средствами (для этого требуются отдельные монографии и учебники). В соответствии с общими принципами настоящей книги речь идет лишь о том, чтобы ввести читателя в круг основных идей, используемых в языках высокого уровня.

5.9.1. Алгол-60. *Алгол*-60 — международный язык программирования, использующий стандартную математическую символику и некоторые английские термины. В национальных версиях языка эти термины могут быть, разумеется, изменены, хотя в целях упрощения международного обмена программами этого обычно не делается. Основные понятия языка алгол-60 (операторы, идентификаторы, арифметические и булевы выражения и др.) были уже фактически описаны в гл. I, поскольку именно этот язык принято считать исходной теоретической базой для всех остальных входных языков (во всяком случае, языков для расчетных задач).

Покажем теперь некоторые средства алгола-60 на примере перемножения двух квадратных матриц A и B с элементами A_{ij} и B_{jk}. Как известно из линейной алгебры, результатом будет квадратная матрица C с элементами C_{ik}, вычисляемыми по формуле $C_{ik} = \sum_{j=1}^{n} A_{ij}B_{jk}$, где через n обозначена размерность матриц A, B, C. Пусть $n = 10$. Тогда соответствующая программа на

алголе-60 запишется в виде:

```
begin real array A, B, C[1:10, 1:10]; real x;integer i, j, k;
for i := 1 step 1 until 10 do
    for k := 1 step 1 until 10 do
        begin x := 0
            for j = 1 step 1 until 10 do x := x + A[i, j] × B[j, k];
        end C[i, k] := x
end
```

Здесь употребляются две пары *операторных скобок* (**begin** — **end**). Наружные скобки выделяют рассматриваемый *программный блок*, внутренние служат для выделения *составного оператора*. Три составляющих его оператора записаны один под одним в три строки. Первый и третий из них являются *операторами присваивания*, а второй — *оператором цикла* по индексу j, меняющемуся с шагом (**step**) 1 от $j = 1$ до $j = 10$. Внутри этого цикла выполняется оператор присваивания $x := x + A[i, j] \times B[j, k]$, составляющий основную сущность процесса последовательного вычисления суммы $\sum_{j=1}^{10} A_{ij}B_{jk}$. Внешняя пара циклов выполняется по двум индексам i и k (меняющимся так же, как и индекс j), указывающим, что операция внутреннего цикла применяется для вычисления не одного, а всех элементов матрицы C.

В верхней строчке программы (сразу после слова **begin**) помещается так называемое *описание*. Описаны три *вещественных массива* (**real array**). В квадратных скобках указаны *границы изменения индексов* этих массивов (от 1 до 10). Кроме того, описана вспомогательная вещественная (**real**) переменная x и целые (**integer**) переменные индексы i, j, k. Описание дает информацию для компилятора о том, с какими данными будет иметь дело программа. Оно используется при трансляции для правильного выделения ресурсов памяти (индекс-регистров, ОЗУ и, быть может, ВЗУ). Предполагается, что входные данные (массивы A, B), выходные данные (массив C) и значения вспомогательной переменной x размещаются в ОЗУ.

Заметим, что специальных средств для описания процедур ввода и вывода данных (включая обмен с ВЗУ) алгол-60 не содержит. Предполагается, что такие процедуры оформляются другими средствами, доводящими их до машинных программ, которые должны использоваться основными программами в качестве подпрограмм.

В процессе написания основной программы программист может задать средствами алгола-60 свои собственные подпрограммы (без процедур ввода-вывода). Такие подпрограммы оформляются в тексте основной программы в виде программного блока, аналогичного блокам основной программы. Отличие заключается лишь

в заголовке, который для подпрограммы начинается служебным словом **procedure** (процедура), за которым следует *идентификатор процедуры* (выбираемый программистом), а за ним в круглых скобках — список *формальных параметров*. В общем случае в этот список включаются подряд как входные, так и выходные параметры. Например, процедура, выполняющая вычисление трех высот $h1$, $h2$ и $h3$ треугольника по его сторонам x, y, z, может быть описана заголовком: **procedure** $H(x, y, z, h1, h2, h3)$.

При обращении к процедуре в тексте основной программы нужно указать ее идентификатор и список *фактических параметров* (определенных в основной программе). Например, оператор $H(5, 7, 8, p, q, r)$ будет присваивать значения высот треугольника со сторонами 5, 7, 8 переменным p, q, r.

Если процедура представляет собой вычисление значения функции, то (единственный) ее выходной параметр может отождествляться с идентификатором самой процедуры и не указываться в круглых скобках. В этом случае тип этого параметра должен указываться перед словом **procedure**. Например, **real procedure** $\log(x)$. В тексте такой процедуры вычисленное значение функции присваивается ее идентификатору. В основной же программе введенную таким образом функцию можно использовать при образовании арифметических выражений, как это имеет место в обычной алгебре, например, $\log(x + 2)$, $\sin x \uparrow 2$ и т. д.

Заметим, что вертикальная стрелка в алголе-60 употребляется для обозначения операции возведения в степень, так что $x \uparrow 2$ означает x^2. Подобный отход от традиционных обозначений связан с тем, что при вводе программы в ЭВМ ее нужно представлять в виде простой последовательности символов, а не в виде «многоэтажных» конструкций. Поэтому, например, вместо $\sqrt{x^2 + y^2}$ пишется обычно $\mathrm{sqrt}(x \uparrow 2 + y \uparrow 2)$, где sqrt — идентификатор функции извлечения квадратного корня, а вместо A_{ij} — $A[i, j]$ и т. д.

Блочное построение программы (с помощью операторных скобок (**begin** — **end**)) позволяет локализовать описания переменных, а если надо, то и процедур, внутри отдельных блоков. Это позволяет программистам, работающим над разными блоками, употреблять одни и те же идентификаторы для обозначения неодинаковых переменных и процедур без опасности их перепутать. В случае необходимости использования идентификаторов в одном и том же смысле во всей программе их описания помещаются в начале самого внешнего блока (охватывающего всю программу). Если те или иные процедуры включены в библиотеку стандартных подпрограмм данной ЭВМ, то транслятор должен «знать» их идентификаторы, так что они могут включаться во все программы без специальных описаний. Их идентификаторы должны, разумеется, быть при этом исключены для обозначений

подпрограмм, формируемых самими прикладными программистами.

5.9.2. Фортран. *Фортран*, название которого происходит от сокращения слов «формульный транслятор», употребляется ныне для обозначения языка программирования, а не транслятора с него. Язык этот был разработан американской фирмой IBM, машины которой наиболее распространены сегодня в мире. Он, как и алгол-60, предназначен в первую очередь для научных расчетов и в основном использует те же самые средства. Разумеется, имеются и отличия. Например, для обозначения операции умножения в фортране используется звездочка, а для операции возведения в степень — две звездочки, так что вместо $a \times b$ в фортране пишется a * b, а вместо a^2 — a ** 2 и т. д. Имеется множество других малосущественных отличий, например употребление в операторе присваивания обыкновенного знака равенства (а не значка :=), использование при индексировании круглых (а не квадратных) скобок, отсутствие операторных скобок вида (**begin — end**) и др.

Наиболее существенным отличием фортрана от алгола-60 следует, однако, считать наличие в фортране мощных средств для управления вводом-выводом, в том числе специальных средств *форматирования* данных. Следует подчеркнуть, что под именем фортран скрывается ныне целое семейство языков, из которых наибольшее распространение получили так называемый *базисный* (basic) фортран, средства которого входят во все языки семейства, и мощный язык фортран IV, в состав которого включается арифметика комплексных чисел и многие другие дополнительные средства. Наши последующие рассмотрения будут касаться в основном базисного фортрана.

Предложения фортрана помещаются обычно в начале программы. Они служат прежде всего для спецификации результатов выходных данных, печатающихся (или формирующихся) на том или ином выводном устройстве. Например, если нужно вывести целые числа, укладывающиеся (вместе со знаком) в 6 (десятичных) разрядов, то употребляют предложение вида N FORMAT (J6). В операторе вывода WRITE (n, N) A, B, C через n обозначен номер устройства, на котором производится вывод, через N — номер предложения, задающего формат вывода, а через A, B, C — идентификаторы переменных, значения которых должны быть выведены: если A = +325, B = —7, C = 67 439, то вывод будет иметь вид ...325, ... —7, 67 438, где точками обозначены пробелы (пустые места) на соответствующем выходном носителе.

Через F$m.n$ специфицируются десятичные числа с фиксированной запятой, причем m означает длину полного формата, а n — количество дробных разрядов. При большем числе разрядов

вывод производится с соответствующим округлением. Для спецификации чисел с плавающей запятой используется символ F*m.n*. Имеются и другие форматы (особенно многочисленные — в фортране IV).

При вводе, выполняемом оператором READ (*n*, N) A, B, ..., значения величин A, B, ..., вводимых устройством с номером *n*, заносятся на соответствующие носители в формате, заданном предложением с номером N. Заметим, что в одном функционирующем предложении могут задаваться разные форматы. Операторы же READ (читать) и WRITE (писать), ссылающиеся на это предложение, вводят или выводят первое значение в первом формате, второе — во втором и т. д.

Фортрановская программа пишется на специальных бланках, в каждой строке которых могут быть записаны 80 символов (по числу колонок в перфокарте). Первые 6 позиций используются для метки оператора (любого целого числа от 000001 до 999999), следующие 66 — уже записи оператора, а последние 8 — под служебную информацию (название программы и др.), которая не обрабатывается компилятором. Если оператор длинный, он может быть перенесен на следующие строки. Некоторые строки, помеченные буквой C в первой позиции (колонке), могут использоваться для любых пояснений (комментариев), которые программист считает нужным сделать для облегчения понимания программы. Эти строки также не обрабатываются компилятором.

Заметим, что фортран в практике программирования употребляется чаще алгола-60. Однако, в отличие от алгола-60, его грамматика менее формализована и к тому же сохраняет в известной мере (прежде всего во вводно-выводной части) машинную ориентацию. Поэтому в теоретических рассмотрениях, а также в заявочных публикациях новых алгоритмов принято использовать алгол-60.

5.9.3. Кобол. Язык *кобол* был разработан американской фирмой IBM прежде всего для автоматизации программирования процессов обработки деловой информации (бухгалтерского учета, банковских операций и др.). Он получил широкое распространение в мире. Поскольку в коболе используется большое количество английских слов, разработаны и применяются на практике его национальные версии (в том числе русская). В русской версии могут использоваться русские идентификаторы, благодаря чему операторы и программы кобола могут быть сделаны удобочитаемыми даже для малоквалифицированных в программировании людей.

Пример такого оператора: ВЫЧИСЛИТЬ СУММА-К-ВЫДАЧЕ = ЗАРПЛАТА ВЫЧЕСТЬ НАЛОГ. Здесь фигурируют три идентификатора переменных: СУММА-К-ВЫДАЧЕ, ЗАРПЛАТА и НАЛОГ. Для их обозначения принято употреблять прописные

(заглавные) буквы. Чтобы идентификатор, выраженный несколькими словами, воспринимался как один, а не несколько разных идентификаторов, они объединяются в единый комплекс с помощью дефисов (черточек), как это имеет место в первом из трех рассмотренных выше идентификаторов.

Заметим, что при получении арифметических выражений в коболе наряду с обычными фортрановскими обозначениями арифметических операций (+, —, *, /), (но не **) можно употреблять их словесное выражение: сложить, вычесть, умножить, разделить. То же самое имеет место для операций сравнения (>, <, =, ≠) и для булевых операций ($\wedge$, $\vee$, $\neg$). Поэтому арифметические и логические выражения могут принимать (полностью или частично) *словесную форму*.

В остальном операторы и составленные из них участки программы в коболе имеют примерно тот же вид, что и в фортране. В полных текстах кобол-программы их операторная часть помещается в последнем по порядку (четвертом) разделе, называемом *разделом процедур*.

В первом разделе, называемом *разделом идентификации*, помещается название программы, имя ее автора (или авторов), дата окончания работы над программой и другие сведения, необходимые для ведения программной документации. Это наименее стандартизованный раздел текста кобол-программы, правила формирования которого могут варьироваться для различных ЭВМ и вычислительных центров.

Во втором разделе, называемом *разделом оборудования*, описываются *конфигурации* (прежде всего ОЗУ и число периферийных устройств разных типов) в общем случае для двух различных ЭВМ (которые в частном случае могут совпадать). Первая ЭВМ будет производить трансляцию, на второй будет работать полученная в результате трансляции машинная программа.

В третьем разделе, называемом *разделом данных*, описываются *логические структуры* файлов и записей, используемых в программе. Термин «логические» призван подчеркнуть, что речь идет о структурных данных, какими они представляются программисту и фиксируются в документах, в отличие от *физических структур*, определяющих реальное размещение данных на машинных носителях. Принципиально важно, что в кобол-программах программист имеет дело с более простыми и наглядными логическими структурами. Реальное же размещение данных во ВЗУ, а также установление соответствия, или, как принято говорить, интерфейса, между физическими и логическими структурами данных возлагаются на транслятор.

Для описания форматов элементарных данных используются символы 9(n), A(n), X(n), называемые в коболе *шаблонами*. Число n в скобках указывает число символов в шаблоне, а 9, A,

X — их тип. Через X обозначается любой символ, через A — буква алфавита, дефис или пробел, а через 9 — произвольная десятичная цифра (не обязательно девятка). Вместо задания числа n можно просто повторить n раз символ типа шаблона. Например, 999 есть шаблон трехзначного десятичного числа. Через T в шаблонах десятичных чисел указывается позиция десятичной запятой (или десятичной точки). Например, шаблоном для числа 027,30 служит выражение 999T99 или 9 (3) T 9 (2). Через M(n) обозначается шаблон наличия множителя 10^n. Например, число 25 в шаблоне 99M(3) интерпретируется как $25 \cdot 10^3 = 25000$. Наличие знака в шаблоне числа указывается символом З (первая буква слова знак).

Для *описания записей* в коболе используются *иерархические* (древовидные) структуры, уровни которых можно нумеровать числами от 01 до 49. Уровень 01 употребляется только для *названия записи* (корня дерева). По мере увеличения уровня иерархии номера уровней при переходе на каждый следующий должны возрастать (не обязательно на единицу). Порядок размещения уровней и названий данных должны соответствовать их размещению в документе. *Атрибуты записи* (листья дерева) сопровождаются описанием соответствующих шаблонов.

Пример описания записи с полными сведениями о сотруднике:

```
01 СВЕДЕНИЯ-О-СОТРУДНИКЕ.
   02 АНКЕТНЫЕ-ДАННЫЕ.
      03 ФАМИЛИЯ; ШАБЛОН A(20).
      03 ИМЯ; ШАБЛОН A(12).
      03 ОТЧЕСТВО; ШАБЛОН A(15).
      03 ДАТА РОЖДЕНИЯ.
         04 ГОД; ШАБЛОН 9(4).
         04 МЕСЯЦ; ШАБЛОН 9(2).
         04 ЧИСЛО; ШАБЛОН 9(3).
      03 ОБРАЗОВАНИЕ; ШАБЛОН A(21).
   02 НОМЕР ОТДЕЛА; ШАБЛОН 9(2).
   02 ЗАРПЛАТА; ШАБЛОН 999T99.
```

В описании файлов указываются имя файла, тип метки, имена записей различных типов, входящих в файл, а также возможное их число, минимальная и максимальная длина записей каждого типа, количество записей в физических блоках и др. Возможные сведения становятся обязательными, если транслятор не способен определить их самостоятельно. Более «интеллектуальные» трансляторы в таких сведениях обычно не нуждаются.

Перечисленные средства составляют так называемый *язык описания данных* кобола. *Язык манипулирования данными* включает в себя операторы ОТКРЫТЬ, ЗАКРЫТЬ, ЧИТАТЬ, ПИСАТЬ, в принципе аналогичные по своим функциям фортранов-

ским операторам OPEN, CLOSE, READ, WRITE. Будучи основными для организации обменных операций, эти операторы дополняются в коболе целым рядом дополнительных операций (отличных от фортрановских), которые, мы, однако, описывать не будем.

5.9.4. PL-1. Язык программирования PL-1 соединяет в себе черты, характерные для языков, предназначенных для научных расчетов, и языков для обработки данных. Это очень мощный язык, получающий с каждым годом все большее распространение. Поскольку основные черты процедурно-ориентированных языков уже нашли свое описание в предыдущих пунктах, мы не будем останавливаться сколько-нибудь подробно на особенностях языка PL-1, тем более, что для такого описания потребовалось бы достаточно много места. Поэтому отметим лишь некоторые особенности, отличающие этот язык от уже описанных. Прежде всего, в PL-1 помимо операций над скалярами употребляются также операции над массивами и другими структурами данных.

Разумеется, с помощью подпрограмм такие операции можно определять и в других языках. Однако в PL-1 они вводятся более естественным путем, позволяя, в частности, строить наряду с обычными скалярными выражениями выражения (формулы) над структурами данных. В PL-1 значительно расширены средства управления вычислительным процессом. В частности, имеются средства для выделения ветвей процессов, которые могут выполняться независимо (асинхронно) от породивших их процессов, а затем снова сливаться (синхронизироваться) с ними. Заметим также, что файлы в этом языке в некоторых русских публикациях, посвященных языку PL-1, называются *теками*. Язык управления данными в PL-1 также отличается богатством своих средств, заимствованных из практики работы с современными операционными системами.

5.9.5. Проблемно-ориентированные языки программирования. Среди проблемно-ориентированных языков программирования отметим прежде всего *языки для моделирования сложных систем*. Они занимают как бы промежуточное положение между процедурно- и собственно проблемно-ориентированными языками. В отличие от обычных входных языков высокого уровня, они располагают большим набором средств для описания одновременно протекающих «внешних» по отношению к программе (обычно стохастических) процессов, например таких, как сборка изделий на конвейере, обслуживание покупателей в магазине, движение автомобилей по дороге и т. п.). Эти процессы по отношению к ведущей их программе выполняют как бы роль данных специального вида — динамически развертывающихся в некотором собственном, так называемом *системном* времени. Второе отли-

чие — это разнообразные средства для оформления результатов моделирования в виде графиков, гистограмм и т. п.

Описанная на языке моделирования сложная система с помощью специального программного обеспечения как бы «приводится в движение» внутри ЭВМ. Меняя условия, в которых работает моделируемая система, получают представление о ее поведении в реальной действительности. Меняя параметры системы, а иногда и ее конфигурацию, выбирают из множества возможных реализаций систему, поведение которой наилучшим образом отвечает заданным критериям.

Вначале наиболее широкое развитие получили языки моделирования чисто дискретных процессов. Это зарубежные языки *симула, симскрипт*, отечественный язык *слэнг* и др. В 70-е годы получили распространение языки моделирования, содержащие также средства описания непрерывных процессов, прежде всего процессов, описываемых системами обыкновенных дифференциальных уравнений. В качестве примера такого языка укажем отечественный язык *недис*.

Как правило, истинно проблемно-ориентированные языки являются гораздо более специализированными. В качестве примеров таких языков приведем отечественные языки *киев*-67 и *киев*-70. Они предназначены для описания процессов управления электронным (или ионным) лучом с целью обработки плоской поверхности. Команды в этих языках состоят из трех частей. В первой части указывается наименование геометрического объекта (набора точек, линии или плоской фигуры), который надлежит обработать. В число таких объектов включаются точечный растр, отрезок прямой, дуга окружности, прямоугольник, круг и т. п. Во второй части команды приводятся числовые значения параметров, характеризующих размеры и положение объекта в пределах разрабатываемого растра, например радиус круга и координаты его центра. Третья часть содержит информацию чисто технологического порядка (шаг перемещения луча, параметры импульсов и др.).

Более сложными являются языки, предназначенные для описания процессов обработки трехмерных деталей на станках с программным управлением. Имеются и многие другие языки «технологического» направления.

5.10. Технология программирования

Сложность процесса программирования растет опережающими темпами по мере увеличения размеров составляемых программ. Небольшие прикладные программы, составляемые одним человеком и помещающиеся полностью в ОЗУ, не требуют, как правило, особо сложной технологии. Программа пишется на одном из

входных языков *высокого уровня* (процедурно- или проблемно-ориентированных) с использованием библиотеки стандартных подпрограмм, транслируется, отлаживается и запускается в работу.

Отладка программы представляет собой важный элемент в общем технологическом цикле создания и эксплуатации программного продукта. Целью отладки является выявление и устранение ошибок, которые могут возникнуть при составлении программы. Технология отладки программы предусматривает использование для всей программы или отдельных ее блоков тех или иных *тестов*. В простейшем случае тестом может служить применение отдельных программных блоков к системам исходных данных, подобранных таким образом, чтобы результат работы блока был очевиден или, по крайней мере, легко вычислим без ЭВМ. Например, для блока, осуществляющего вычисление по формуле

$$z = e^x \cdot \sqrt{x^2 + y^2},$$

можно выбрать в качестве исходных данных пары значений x, y (0, 0), (0, 1), (1, 0).

При отладке может использоваться специальный режим *покомандного* (или поблочного) исполнения программы, при котором после испытания очередной команды (или блока) происходит остановка процесса и вывод результатов для анализа их программистом. Для устранения потерь машинного времени на анализ результатов поблочных тестов подобный метод отладки должен осуществляться в мультипрограммном режиме, или еще лучше,— в режиме разделения времени.

При более сложных системах тестирования программисту может выдаваться распечатка (так называемый *дамп*) полного содержимого рабочих ячеек ОЗУ и состояния центрального процессора.

Следует отметить, что большое число ошибок, полученных при написании программ на входных языках (особенно на языках высокого уровня), выявляются специальными программами *синтаксического контроля*, применение которых предшествует процессу трансляции. Места ошибок при этом точно указываются программисту. Правда, таким образом выявляются только так называемые *синтаксические ошибки*, т. е. нарушение правил грамматики алгоритмического языка, на котором написана анализируемая программа. Что же касается *содержательных ошибок* (например, неправильный адрес перехода и др.), то их выявление требует совместной работы ЭВМ и программиста.

Для простых программ перечисленных средств оказывается достаточно. Однако сложные программы объемом в десятки и даже сотни тысяч машинных команд требуют для своей разработки и эксплуатации дополнительных средств. Необходимость в

таких средствах возникает уже на самом первом этапе — *формализации* и *первичной алгоритмизации* процессов, которые предстоит запрограммировать.

Задача этого этапа состоит прежде всего в том, чтобы распределить работу по написанию отдельных частей программы между коллективами программистов и обеспечить последующую стыковку разработанных ими участков программы. Для решения этой задачи чаще всего используются так называемые *блок-схемы* будущей программы.

В блок-схеме программы отдельные блоки программы изображаются обычно прямоугольниками, связи между блоками — стрелками, а условия, определяющие выбор той или иной стрелки,— кружками или ромбами. Пример простейшей блок-схемы, состоящей из двух блоков A и B и условия α, изображен на рис. 5.1. Стрелки на блок-схеме показывают передачи управления и направления движения информации между отдельными программными блоками. Руководитель проекта должен не только нарисовать блок-схему будущей программы, но и сформулировать задачи, решаемые отдельными блоками, сущность программных условий, а также точно определить входные и выходные данные каждого блока. После этого написание программ, решающих задачи всех блоков, а также программ, осуществляющих проверку всех условий, может быть поручено разным программистам.

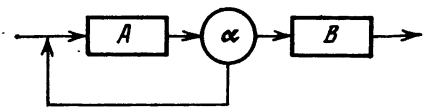

Рис. 5.1

В общем случае различные блоки могут быть запрограммированы в различных входных языках, которые могут иметь неодинаковые средства описания данных. Поэтому при их трансляции необходим специальный процесс *редактирования связей*. Программы, обеспечивающие этот процесс, включаются в состав программных средств автоматизации программирования. Разумеется, чтобы связи сработали правильно, руководитель проекта должен обеспечить унификацию обозначений данных.

Недостатком обычного метода блок-схемы является малый уровень формализации задач, решаемых программными блоками. В так называемой *R-технологии программирования* этот недостаток устраняется за счет изменения содержания, вкладываемого в само понятие блок-схемы. Блок-схема в *R*-технологии используется только для указания маршрутов движения данных и передач управления; что же касается операторов, обрабатывающих эти данные, то они могут накладываться на уже готовую блок-схему позднее. При этом степень подробности блок-схемы такова, что она охватывает все циклы и передачи управления в программе, за исключением тех, которые включены в состав отдель-

ных операторов. Благодаря этому программные блоки в R-технологии имеют вид простых последовательностей элементарных операторов используемого языка (без переходов и циклов) и могут варьироваться произвольным образом без нарушения общей логической структуры программы. В результате создается возможность разделить процессы написания и даже отладки логических схем программы от содержательного наполнения этих схем. Сама логическая схема в R-технологии также имеет не вполне традиционный вид. Она изображена в виде направленного *R-графа*, подобного тому, который изображен на рис. 5.2. На стенках (дугах) этого графа с одной стороны изображаются условия (α_i), при выполнении которых передачи управления происходят от этой стрелки. С другой стороны стрелки помещаются операторы или последовательности операторов A_i, которые должны быть выполнены в этом случае. Стрелка, помеченная на рис. 5.2 буквой N означает передачу описания логической схемы на другой лист бумаги с номером N. Таких передач на другие листы в схеме может быть несколько.

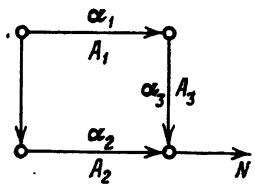

Рис. 5.2

При использовании традиционных входных языков логические схемы программ даже средней сложности могут оказаться слишком громоздкими для того, чтобы быть спроектированными и отлаженными одним человеком за разумное время. В R-технологии это противоречие разрешается за счет использования в качестве входного языка автокода некоторой воображаемой ЭВМ (так называемой *R-машины*). В состав команд R-машины помимо операций с обычной (прямо адресуемой) памятью вводятся операции со стековой и некоторыми другими специальными видами ЗУ (пока не используемыми в реальных ЭВМ). Благодаря этому операторы *R-языка* (языка R-машины) делаются весьма емкими, выражаясь порой многими десятками команд обычных ЭВМ.

Самое же главное состоит в том, что внутрь таких операторов «втягиваются» многие циклы и переходы, возникающие при программировании различных процессов обработки нечисловой информации. Тем самым резко упрощаются логические схемы программ для таких процессов, составление и отладка подобных программ приобретают такую же простоту и ясность, как и для случая вычислительных программ при условии использования входных языков, ориентированных на вычислительные процедуры (алгол, фортран и др.).

Другой подход к методике проектирования больших программ дает так называемый *метод формализованных технических заданий*. В упрощенном виде суть этого метода состоит в том, что

при проектировании сложных программ создается ряд проблемно-ориентированных алгоритмических языков A_1, A_2, ..., A_m с последовательно разукрупняющимися операторами и логическими условиями. Руководитель проекта пишет полный текст программы на языке A_1 и разделяет его на несколько программных блоков, поручая дальнейшее их разукрупнение в языке A_2 уже не одному, а группе программистов (выделяя соответствующие блоки и контролируя связи между ними). Процесс этот продолжается, пока на очередном этапе не будет получена запись программы на каком-либо известном входном языке A_m, трансляция с которого может быть выполнена автоматически. Не исключено, разумеется, что последняя запись будет написана не на одном входном языке A_n, а на нескольких.

Все языки на предыдущих этапах формализуются лишь в той степени, чтобы записи на них однозначно понимались всеми членами работающей над задачей бригады программистов. Как правило, эти языки проблемно-ориентированны. Например, в языках верхних уровней, предназначенных для написания шахматных программ, могут использоваться в качестве операторов известные алгоритмы проведения шахматных эндшпилей (мат одинокому королю королем и ладьей и т. п.), а в качестве логических условий — наличие слабого поля, открытых линий и т. п.

Описанный метод представляет собой пример технологии проектирования программ сверху вниз, т. е. от более укрупненных к более и более детализированным описаниям. Другой тип технологии предполагает обратное направление процесса проектирования программ снизу вверх. При этом вначале создаются наборы подпрограмм и программные модули, решающие отдельные части программируемой задачи. После построения таких заготовок из них собирается полная программа. Подобная методика носит название *модульного программирования.*

5.10.1. Структурное программирование. Практика программирования показывает, что источником наибольшего количества ошибок и наибольших трудностей при отладке является использование операторов безусловного и условного переходов. Большое число таких операторов часто делает логическую схему программы практически необозримой и сильно затрудняет разбиение программы на части для работы над ней бригады программистов. Автором еще в 1965 г. было показано, что любую программу можно привести к виду, не использующему операторов перехода. Программы такого вида впоследствии получили наименование *структурированных программ.* В них помимо обычного последовательного выполнения элементарных операторов используются составные операторы (также имеющие вид структурированных программ) в виде условных операторов и операторов цикла (см. гл. I).

Логические схемы структурированных программ отличаются большой простотой и наглядностью. Технология их написания и отладки, получившая наименование *структурного программирования*, обеспечивает сокращение времени подготовки программ в два раза и более. Разумеется, эта технология не сводится только к исключению из арсенала программистов операторов перехода. Заметим лишь, что, как было показано в 1965 г. автором, структурированные программы можно рассматривать как формулы в некоторой алгебре. Зная правила преобразования выражений в такой алгебре, можно осуществлять глубокие формальные (и потому полностью автоматизируемые) преобразования программ. Тем самым в системе автоматизации программирования появляется возможность иметь гораздо более сильные, чем обычно, средства автоматической оптимизации программ.

5.10.2. Документирование и ведение программ. В настоящее время программы для ЭВМ представляют собой один из видов общественного продукта. Составленная программа не является «личным инструментом» составившего ее программиста. Она, как правило, используется более или менее широким кругом других пользователей, которые при этом должны, разумеется, понимать особенности работы с нею. Кроме того, программы в процессе эксплуатации подвергаются различного рода изменениям и усовершенствованиям. Необходимость внесения изменений в программы (даже в том случае, когда они написаны на входных языках высокого уровня) вызывается изменением типов и конфигураций используемых ЭВМ. Таким образом, программный продукт нуждается не только в начальной разработке и отладке, но и в четко организованном постоянном *ведении*.

Все сказанное убедительно свидетельствует о том, что составление и последующие изменения программного продукта нуждаются в тщательно продуманной *системе документирования*. В такую систему помимо собственно текстов программ на входных и машинных языках должны включаться их достаточно подробные описания, инструкции для пользователей и другие документы. Поскольку, как уже отмечалось выше, документация может подвергаться изменениям, наилучшая система документирования предполагает безбумажный способ ее ведения. При этом все документы хранятся на машинных носителях, а специальная инструментальная программная система позволяет работать с документацией как лицам, ответственным за ее ведение, так и пользователям. В системе обеспечивается возможность быстро вносить нобходимые изменения в хранимые документы, а также в случае необходимости получать оперативно в нужных количествах их копии как в машинной, так и в традиционной бумажной форме.

5.11. Обеспечение надежности машинной обработки информации

Огромная сложность технических и программных средств, применяемых в современных системах обработки данных, остро ставит вопрос об их *надежности*. Надежность современных электронных схем, достаточно высокая сама по себе, дополняется развитыми системами *машинного контроля*. В местах возможного возникновения кратковременных сбоев или выхода из строя отдельных элементов применяются так называемые *коды с обнаружением ошибок*, а для особо ответственных схем — *коды с (автоматическим) исправлением ошибок*. В таких кодах (обычно двоичных) имеются дополнительные — так называемые *контрольные разряды*.

О простейшем виде *контроля по четности* уже говорилось выше (в гл. III). Единственный контрольный разряд в этом виде контроля выбирается таким образом, что во всем коде (вместе с контрольным разрядом) имеется четное число единиц (0, 2, 4, 6 и т. д.). Нарушение этого правила в каком-либо месте контролируемой схемы вызывает сигнал прерывания с тем, чтобы возникшая ошибка не повлияла на результат работы исполняемой программы. По этому сигналу управление передается обычно специальной *тестовой программе*. Она устанавливает характер возникшей ошибки (случайный кратковременный сбой или постоянная неисправность). В первом случае исполнение прерванной программы возобновляется. При обнаружении неисправности длительного действия тестовая программа локализует место ее возникновения и выводит необходимую информацию на пульт *системного оператора*, управляющего работой ЭВМ (*системный пульт*), для организации ее устранения.

Идея (двоичного) кода с исправлением одиночной ошибки состоит в следующем. Пусть исходный (незащищенный) код имеет m (двоичных) разрядов. Выберем $n > m$ так, чтобы $2^n/(n+1) \geqslant 2^m$. Полученный *защищенный код* будет иметь $k = n - m$ *контрольных разрядов*, причем $2^k \geqslant m + k + 1 = n + 1$. Каждому из n разрядов присваивается номер от 1 до n. Далее для любого значения защищенного (n-разрядного) кода составляется k *контрольных сумм* $S_1, \ldots, S_k$ по модулю 2 значений специально выбранных разрядов этого кода. Для S_i выбираются разряды, для которых двоичные коды номеров имеют в i-м разряде единицу. Для суммы S_1 это будут, очевидно, разряды с номерами 1, 3, 5, 7, ..., для суммы S_2 — разряды с номерами 2, 3, 6, 7, 10, 11, ... и т. д.

При любом исходном коде контрольные разряды могут быть выбраны так, чтобы все контрольные суммы были равны нулю. Проверочный (двоичный) код $S = S_k S_{k-1} \ldots S_1$ при этом будет ну-

левым. При таком условии защищенный код называется *правильным*. При возникновении одиночной ошибки в этом коде в любом (безразлично — основном или контрольном) разряде проверочный код будет отличен от нуля. При этом в силу способа выбора контрольных сумм значение проверочного кода S будет представлять собой двоичный код номера разряда защищенного кода, в котором возникла ошибка. Для исправления этой ошибки достаточно изменить значение указанного разряда на противоположное (если оно равно 0, то на 1, а если 1 — то на 0). Соблюдение неравенства $2^k \geqslant n+1$ обеспечивает устранение ошибки в любом разряде защищенного кода.

Описанный процесс исправления одиночной ошибки может выполняться автоматически специальным *корректирующим блоком*, который может быть выполнен в виде как комбинационной, так и последовательностной схемы (см. гл. I). Размещая такие блоки в различных трактах передачи информации ЭВМ, можно обеспечить правильность ее работы даже при выходе из строя многих ее элементов (отдельных разрядов регистров или ячеек ЗУ, отдельных шин передачи информации и т. д.).

Из неравенства $2^n/(n+1) \geqslant 2^m$, которое можно переписать в виде $2^k \geqslant m+1+k$, легко находим, что для исправления одиночной ошибки к 16-разрядному коду достаточно добавить 5 контрольных разрядов, к 32-разрядному 6 и т. д. Возможно строить коды с исправлением не одной, а любого заданного числа ошибок. Однако при этом число контрольных разрядов должно быть соответственно увеличено. Заметим также, что коды с исправлением ошибок позволяют обнаруживать большее число ошибок, чем они способны исправить.

Помимо кодов с обнаружением и исправлением ошибок применяются и другие методы *аппаратной защиты* от возможных ошибок. Один из наиболее простых состоит в использовании вместо одного какого-либо блока (а иногда и целой ЭВМ) трех совершенно одинаковых параллельно работающих блоков. На выходах этих блоков ставится *схема мажоритарного выбора*, формирующая на своем выходе тот сигнал, который выдает большинство присоединенных к ней блоков (т. е. 2 или 3). Тем самым обеспечивается правильная работа даже при полном выходе из строя одного из блоков.

Использование методов аппаратной защиты от ошибок позволяет строить в принципе сколь угодно надежные электронные схемы любой сложности. На практике существуют ЭВМ, электронная часть которых может работать безошибочно в течение нескольких лет. Разумеется, наличие столь мощной защиты сильно удорожает ЭВМ. Поэтому на практике аппаратную надежность ЭВМ сегодня доводят обычно до нескольких сотен или нескольких тысяч часов безошибочной работы. Остальная часть га-

рантии получения безошибочных конечных результатов возлагается на средства программной защиты.

Часть этих средств включается в операционную систему. Современные ОС, как уже отмечалось выше, автоматически придают программам пользователей те или иные периферийные устройства. При неисправности в любом из таких устройств (надежность которых, как правило, существенно ниже надежности электронных схем) ОС автоматически обеспечивает его отключение и замену другим (исправным) устройством. ОС обнаруживает и предотвращает ошибки, которые могут возникнуть от неправильных действий программы пользователей (нарушение установленных форматов данных, попытки обращения к отключенным устройствам и т. д.). В ряде современных управляющих ЭВМ, предназначенных для управления особо ответственными процессами, ОС позволяет по мере выхода из строя отдельных элементов и узлов автоматически перестраивать структуру машины, отключая часть менее важных программ, с тем чтобы обеспечить максимальную живучесть машины в смысле правильного исполнения ею наиболее ответственных функций.

Значительные возможности обеспечения защиты от ошибок имеются и у самих пользователей. Составляя программы, они могут предусматривать в них повторный счет (с мажоритарным выбором результата), различные контрольные проверки (подобно тому, как это делается в обычных расчетах, производимых вручную) и т. д.

Разумная комбинация всех перечисленных средств защиты может обеспечить любую требуемую гарантию надежности результатов, получаемых с помощью ЭВМ.

5.11.1. Защита памяти. При мультипрограммном режиме работы ЭВМ возникает новый источник ошибок, а именно, возможность записи информации одними программами в зоны памяти, отведенные для других программ. В таком случае может произойти непоправимая порча данных, с которыми оперируют программы, а иногда и самих программ. Для защиты от подобных ошибок вводится так называемый *ключ защиты памяти*. При работе ЭВМ ОС разбивает оперативную память на блоки, каждому из которых присваивается некоторое значение ключа защиты этого блока. Число этих значений выбирается равным $n+1$, где n — число программ, одновременно исполняемых в ЭВМ (в мультипрограммном режиме).

Распределяя блоки памяти между программами, диспетчер-планировщик составляет таблицу, в которой каждому блоку отводится номер (от 1 до n) программы, использующей этот блок. Нулевое значение ключа соответствует незащищенным блокам (обычно таковыми являются блоки, не закрепленные ни за одной программой). При исполнении программы присвоенный ей

номер запоминается в УУ центрального процессора в специальном регистре ключа защиты памяти, который включается в слово состояния программы. При записи информации в ОЗУ содержимое этого регистра сравнивается со значением ключа, приписанного блоку ОЗУ, в который происходит обращение. Всякая попытка записи в блок, защищенный ключом другой программы, автоматически блокируется.

То же самое происходит и в том случае, когда участок данной программы исполняется не центральным, а канальным процессором. Номер исполняемой программы запоминается в УУ канала (в адресном слове канала) и в случае попытки записи в ОЗУ сравнивается с ключом защиты соответствующего блока ОЗУ. Срабатывание защиты памяти (при попытках записи информации в «чужие» блоки ОЗУ) как при работе ЦП, так и канала не только блокирует запись, но и формирует соответствующий сигнал прерывания для возможности включения программ ОС, помогающих устранить возникший конфликт. Необходимость приписывания блокам ОЗУ ключей защиты вызывает появление команд: *установить ключ* и *прочитать ключ*.

Заметим, что защита памяти предназначена для блокирования лишь несанкционированной записи. Чтение «чужой» информации при этом не блокируется. Для защиты от несанкционированного чтения секретной информации используются другие средства, описываемые в следующей главе. Оба вида защиты от *несанкционированного доступа* (записи или чтения) могут применяться не только к ОЗУ, но и ко всем другим видам памяти, прежде всего к памяти на магнитных лентах и дисках.

5.12. Операционные системы ЕС ЭВМ

Поскольку операционные системы представляют собой весьма сложные программные комплексы (содержащие многие сотни тысяч машинных команд), они обычно наращиваются постепенно. В результате возникает целое семейство операционных систем с разными наборами возможностей. Для ЕС ЭВМ первоначально был создан относительно простой вариант так называемой *дисковой операционной системы* (ДОС ЕС). Затем последовательно вводились все более и более совершенные *версии* операционной системы, различаемые номером, который состоит из двух цифр, разделенных точкой (например, ОС ЕС 4.0).

Следует подчеркнуть, что с машиной фактически поставляется не сама ОС, а лишь некоторая ее заготовка, которую еще необходимо настроить на конкретную *конфигурацию* ЭВМ (номер модели, объем ОЗУ, состав периферийного оборудования и др.). При настройке планируются также размер и состав библиотек стандартных подпрограмм, стандартные назначения вводно-вывод-

ных устройств и устройств внешней памяти для нужд самой ОС и др. Процесс настройки называют также *процессом генерации* конкретной ОС (той или иной версии) для ЭВМ данной конфигурации.

Заметим, что в ЕС ЭВМ в состав ОС принято включать программное обеспечение системы автоматизации программирования. Развитые версии ОС, помимо языка ассемблера, обеспечивают возможность использования в качестве входных языков фортран, кобол, PL-1. Возможно использование и других языков, которым не нашлось места в нашем изложении. В своей управляющей части развитые версии ОС ЕС предоставляют пользователю все средства, которые описаны в настоящей главе, и многие дополнительные средства, не нашедшие в ней отражения.

В состав ОС ЕС ЭВМ включаются также различного рода служебные программы, осуществляющие изменение форм представления данных и особенно массивов и файлов. В их числе отметим особо *программы сортировки*, осуществляющие *упорядочение* массивов (файлов), например, упорядочение массива чисел в порядке их роста или убывания.

Глава VI
БАЗЫ ДАННЫХ

6.1. Общие сведения

В начальный период развития ЭВМ организацией данных занимались прикладные программисты. Каждая прикладная программа снабжалась своими собственными данными, которые вводились в ЭВМ вместе с программой. Пока на ЭВМ решались отдельные, не связанные друг с другом задачи, такое положение было естественным. Однако начиная с 60-х годов и особенно в 70-е годы основная масса применений ЭВМ получила другое, так называемое *системное* направление.

В системных применениях задачи, решаемые на ЭВМ, взаимосвязаны в единое целое общей целью, например, управлением тем или иным сложным объектом, обработкой регулярных потоков данных и т. п. Имея дело с единым объектом, работающие в ЭВМ программы используют и формируют ту или иную *систему данных* об этом объекте, которую мы будем называть его *информационной моделью*.

На первых порах и в системных применениях преобладал *позадачный подход*, при котором каждая программа использовала свои данные, формируемые в изолированном от других задач процессе. При этом приходилось повторять многократный ввод и вывод по существу одних и тех же данных. Например, при управлении предприятием определенные данные о его производственном персонале используются не только в системе кадрового учета, но и при решении задач планирования, начисления зарплаты и т. д. Естественно поэтому ввести эти данные в ЭВМ один раз, создав *базу данных* о кадрах предприятия и предоставив возможность любым прикладным программам черпать необходимые для них данные из этой базы.

Выгодность подобного подхода имеет несколько аспектов. Во-первых, достигаемая при этом *одноразовость ввода* резко уменьшает нагрузку на устройства ввода и подготовки данных, являющиеся наиболее узким местом безбумажной информатики как в части пропускной способности, так и трудоемкости. Второе пре-

имущество заключается в достижении *независимости* процесса сбора и обновления или, как принято говорить, *актуализации* данных от процесса их использования прикладными программами.

Это позволяет в свою очередь, во-первых, разделить работу по автоматизации этих процессов между разными коллективами программистов и тем самым ускорить процесс создания соответствующих автоматизированных систем. Во-вторых, устраняется опасность ошибок из-за возможной разновременности актуализации данных в различных программах. И, наконец, в-третьих, достигается бо́льшая гибкость как в изменении и расширении состава прикладных программ, так и в отношении совершенствования организации самой базы данных.

Последнее преимущество предполагает *независимость* прикладных программ от *физической организации* базы данных. Такая независимость достигается с помощью специального (системного) программного обеспечения, которое интерпретирует так называемый *язык манипулирования данными*. Будучи процедурно, а не машинноориентированным, подобный язык, при использовании его в качестве дополнения к процедурно- или проблемно-ориентированным языкам программирования, позволяет писать прикладные программы, не вникая в фактическое расположение данных во внешней памяти.

Универсальные базы данных, обслуживающие любые запросы прикладных программ, вместе с соответствующим программным обеспечением принято называть *банками данных*. Подобно тому как обычный банк является хранилищем вкладов, банк данных может рассматриваться как хранилище информации (баз данных), принадлежащей в общем случае многим пользователям. Владельцы информации определяют *правила доступа* к ней других пользователей, правила ее актуализации, длительность хранения и др. Для реализации этих возможностей банк данных снабжается специальным программным обеспечением. Кроме того, функционирование банка предполагает наличие одного или нескольких человек, выполняющих функции *администрации банка*. Не являясь формально владельцем содержимого банка, администрация несет полную ответственность за правильность его функционирования.

6.2. Справочно-информационные системы

Первые банки данных были использованы для автоматизации справочно-информационной деятельности. Здесь понятие банка данных выступает в наиболее чистой форме ввиду фактического отсутствия прикладных программ. В *автоматизированных информационных системах* (АИС) их заменяет интерпретатор *языка*

запросов, позволяющий формулировать требования к выдаче тех или иных *справок*. В простейшем случае такая справка представляет собой одну или несколько записей из некоторого *файла* (см. § 1.8).

6.2.1. Фактографические АИС. Различают *фактографические* АИС, у которых базы данных составляются из *форматированных* (формализованных) *записей*, и *документальные* АИС, записями которых могут служить различные неформализованные документы (статьи, письма и т. п.). В простейшем случае фактографическая АИС имеет в качестве базы один файл с записями *фиксированного формата* (все записи имеют одну и ту же длину). Примером форматированных записей могут служить, скажем, записи об операциях по приему и выдаче денег в сберкассе; запись имеет четыре основных атрибута: дата, характер операции (принято, выдано), сумма, остаток вклада.

В качестве форматированной записи может рассматриваться кадровая анкета (личный листок по учету кадров). Правда, такие ее разделы, как «прежняя работа», «поездки за границу» и др. в обычной анкете не до конца формализованы и, главное, имеют переменную длину, поэтому при автоматизации кадрового учета необходимы некоторые доделки. В частности, полная формализация раздела «прежняя работа» требует составления классификатора предприятий и организаций (причем не только в настоящем, но и в прошлом). Кроме того, обычно бывает целесообразно фиксировать максимальное количество позиций в каждом разделе и тем самым выравнивать длину записей (у многих записей при этом могут возникнуть позиции с пустым заполнением). С этими дополнениями мы будем рассматривать систему кадрового учета как базовый пример простейшей фактографической АИС.

Среди атрибутов (форматированных) записей обычно существует атрибут, который однозначно *идентифицирует* запись. В системе кадрового учета таким атрибутом является *учетный номер* работника, которому принадлежит данная запись (анкета). Подобный атрибут называется *основным* (или *первичным*) *ключом*. По нему определяется (с помощью специальной программы или таблицы) *адрес записи* во внешней памяти.

Одной из важнейших задач АИС является быстрый подбор записей, обладающих теми или иными *свойствами*, например — выбрать все анкеты людей данного возраста или данной профессии. Атрибуты, которыми задаются эти свойства, как правило, идентифицируют не одну запись, а некоторое множество записей. Они называются *дополнительными* (или *вторичными*) *ключами*.

Поиск нужных записей по дополнительному ключу обычно разбивается на два этапа. На первом этапе определяются значения основного ключа, отвечающие записям с заданным значени-

ем дополнительного ключа. На втором этапе по найденным значениям основного ключа находят адреса записей, а затем и сами записи. Для быстрого выполнения первого этапа (без последовательного просмотра всех записей) используются так называемые *инвертированные списки.* Каждый такой список состоит из пар значений дополнительного и соответствующего им множества значений основного ключа, упорядоченных по дополнительному ключу.

Пусть, например, первоначальный файл состоит из записей с тремя атрибутами: учетный номер, год рождения, код профессии. Возьмем для простоты лишь 12 записей (табл. 6.1).

Таблица 6.1

Учетный номер	1	2	3	4	5	6	7	8	9	10	11	12
Год рождения	1950	1945	1950	1947	1945	1950	1951	1953	1949	1950	1957	1946
Код профессии	03	01	05	03	02	04	03	03	02	01	08	09

Список, инвертированный по ключу год рождения, приведен в табл. 6.2.

Таблица 6.2

Год рождения	1945	1946	1947	1949	1950	1951	1953	1957
Учетный номер	2, 5	12	4	9	1, 3, 6, 10	7	8	11

Список, инвертированный по ключу код профессии, приведен в табл. 6.3.

Таблица 6.3

Код профессии	01	02	03	04	05	08	09
Учетный номер	2, 10	5, 9	1, 4, 7, 8	6	3	11	12

Объединение инвертированных списков по всем дополнительным ключам составляет так называемый (полностью) *инвертированный файл.* Имея такой файл, легко найти записи с данными

атрибутами. Например, в нашем случае профессию с кодом 03 и одновременно год рождения 1950 имеет лишь работник с учетным номером 1. Если инвертированные списки не перекрывают все множество ключей, то говорят о *частично инвертированном файле.*

6.2.2. Документальные АИС. Основной задачей, решаемой в документальных АИС, является поиск документов по их *содержанию.* Если язык запросов (как и язык самих документов) является обычным (неформализованным) человеческим языком (возможно, с тем или иным профессиональным уклоном), то *полное* решение задачи поиска требует понимания системой *смысла* запросов. Такая задача тесно связана с проблемой создания так называемого *искусственного интеллекта* и на сегодняшний день еще достаточно далека от полного решения.

Поэтому в практике документальных АИС употребляются упрощенные способы поиска. Простейшим из них является использование так называемых *дескрипторов*, под которыми понимается некоторое фиксированное множество слов (в том числе специальных профессиональных терминов), которые, по мнению разработчика данной конкретной АИС, в наибольшей степени характеризуют содержание ее документального фонда. Например, если фонд составляют статьи по медицине, то в качестве дескрипторов естественно использовать названия болезней, лекарств и лечебных процедур, а также такие термины, как лечение, диагностика и др.

Наиболее прямолинейное решение задачи поиска при использовании фиксированного *словаря дескрипторов* заключается в следующем: АИС просматривает текст запроса (на обычном, неформализованном языке) и фиксирует все встречающиеся в тексте дескрипторы. Например, в запросе «найти все статьи по диагностике рака» выделяются дескрипторы «диагностика» и «рак». После этого система просматривает полные *тексты* всех документов (статей) и отбирает из них те, которые содержат оба выделенных дескриптора.

При реализации подобной процедуры нужно дополнительно учесть то обстоятельство, что как в тексте запроса, так и в тексте документа дескрипторы могут трансформироваться (и притом по-разному) в соответствии с требованиями грамматики используемого языка. В приведенном примере дескриптор «рак» встречается в родительном падеже с окончанием «а», а в тексте статьи он может встретиться в других падежах. Поэтому идентификация дескрипторов должна проводиться с точностью до окончаний (а иногда и до суффиксов).

Следует иметь в виду также и то, что просмотр полных текстов документов требует много времени. Требования повышения производительности АИС заставляют вместо просмотра самих

документов просматривать их *поисковые образы*. В качестве таковых в простейшем случае выступают просто перечни встречающихся в документах дескрипторов. Поисковые образы составляются заранее либо вручную, либо автоматически (в результате одноразового просмотра текстов специальной программой). Обычно эти образы хранятся отдельно от текстов самих документов, имея в своем составе *ссылку на адрес* соответствующего документа. Аналогичным образом для запроса составляется *поисковый образ запроса*. В процессе поиска происходит сравнение поисковых образов документа и запроса на основе *критерия смыслового соответствия*, фиксированного для системы, либо указываемого при формулировке запроса. При этом документ считается *релевантным* запросу, если условие сравнения выполняется. В качестве критерия смыслового соответствия может выступать, например, условие совпадения множеств дескрипторов поисковых образов документа и запроса, включение множеств друг в друга, их пересечение и др. Поскольку наиболее дешевым способом хранения информации являются сегодня микрофиши, то для хранения полных текстов документов используют обычно их. В памяти же ЭВМ (внешней) хранятся лишь поисковые образы документов.

Специальные устройства автоматического поиска и копирования в библиотеке микрофишей позволяют после нахождения адреса документа в считанные секунды получить его так называемую *твердую копию*. Такая копия представляет собой обычный бумажный документ со шрифтом нормального (ориентированного на человека) размера.

Заметим, что документальная АИС с простыми дескрипторными поисковыми образами может рассматриваться как фактографическая система с булевыми (да — нет) атрибутами, число которых равно полному числу используемых дескрипторов. Однако такое представление является экономичным лишь при относительно небольшом числе дескрипторов.

Несовершенство описания содержания простой системой дескрипторов заключается прежде всего в том, что в обычных языках имеется много средств для выражения одного и того же смысла. В результате может оказаться, например, что в статье по диагностике той или иной болезни само слово «диагностика» не будет встречаться ни разу. Наоборот, наличие в тексте статьи термина «диагностика» еще не означает, что статья посвящена именно диагностике. Это может быть всего-навсего случайная ремарка. Поэтому при поиске по дескрипторам могут быть извлечены так называемые *нерелевантные* документы, т. е. не имеющие отношения к рассматриваемому запросу, и, наоборот, некоторые релевантные документы могут оказаться не найденными. В первом случае говорят о *неточности* (*информационном шуме*) рассматриваемой АИС, во втором — о ее *неполноте*.

Применительно к каждому конкретному запросу определяются (экспертным путем) два коэффициента. *Коэффициент полноты* представляет собой отношение числа выданных по запросу релевантных документов к их общему числу в поисковом массиве. *Коэффициент точности* есть отношение релевантных (в данной выдаче) документов к общему числу выданных (релевантных и нерелевантных) документов. Для многих запросов система может характеризоваться как средними значениями этих коэффициентов, так и их минимальными (наихудшими) значениями.

Несовершенство простого языка дескрипторов проявляется также и в том, что на нем в принципе не могут быть выражены некоторые важные нюансы смысла. Например, наличие в документе дескриптора «поставка» и дескрипторов — имен A и B поставщика и потребителя ничего не говорит о том, кто кому поставляет ($A \rightarrow B$ или $B \rightarrow A$). Для выражения подобных отношений в множество дескрипторов должны быть включены служебные слова (прежде всего предлоги) и построена формализованная грамматика, превращающая это множество в некоторый формальный язык. В зависимости от мощности изобразительных средств такого языка можно в той или иной степени улучшить информационные характеристики АИС. Однако за такое улучшение приходится расплачиваться усложнением поиска, а следовательно, уменьшением производительности и увеличением стоимости системы.

6.3. Организация последовательных файлов

Последовательным файлом называется именованная линейно упорядоченная последовательность записей одного и того же типа. Именно в таком виде файл представляется прикладному программисту. Однако в зависимости от вида используемой памяти реальное *физическое* размещение в ней отдельных записей может сильно отличаться от подобного представления. При работе с файлами возникает необходимость как *последовательного просмотра* записей, так и *прямой адресации* записи по ее ключу.

Последовательный просмотр записей проще всего обеспечивается соответствующим (последовательным) их размещением в памяти. Так поступают, как правило, при использовании ЗУ с последовательным доступом (магнитных лент). В случае ЗУ других типов записи часто размещаются в другом порядке, определяемом прежде всего удобствами их прямой адресации. Для возможности перехода от очередной записи к следующей за ней пользуются встроенными в записях *указателями*. Наиболее быстрый поиск следующей записи происходит, если в качестве указателя используется истинный физический адрес ее начала. Одна-

ко при этом указатели становятся зависимыми от положения файла в памяти. При его перемещениях указатели надо менять. Более удобными в этом отношении являются *относительные адреса*, показывающие положение в ЗУ следующих по порядку записей по отношению к данным записям. Возможны и более сложные способы формирования указателей, обеспечивающие полную машинную независимость процедур последовательных просмотров файлов.

Заметим, что при порче какого-либо указателя часть записей может оказаться потерянной для программ последовательного сканирования файлов. Обеспечение проверки и восстановления искаженных или утерянных указателей возможно при введении обратных указателей (от данной записи — к предшествующей ей). Такое восстановление делается специальными программами обеспечения *сохранности файлов*.

Прямая адресация записей (по их ключам) выполняется различными типами процедур. В случае ЗУ с последовательным доступом прямая адресация так или иначе требует последовательного сканирования файла со сравнением ключа каждой очередной записи с заданным (для поиска) значением ключа. В ЗУ других типов могут использоваться более быстрые процедуры адресации. Если записи в ЗУ упорядочены по ключу, то можно использовать, например, *блочный метод*. При нем сначала просматриваются записи в начале крупных блоков записей с целью фиксации блока, в котором находится искомая запись. После нахождения нужного блока поиск в нем можно организовать либо последовательным сканированием, либо снова блочным способом (с менее крупными блоками). Подобный прием *иерархического блочного поиска* используется человеком при поиске нужного слова в энциклопедии (сначала фиксируется том, затем страница и, наконец, сканируется найденная страница).

Если в иерархическом блочном поиске на каждом шаге используются лишь два блока (равных с точностью до единицы длины), то приходим к наиболее быстрому (в классе блочных методов) методу *дихотомического* (или *двоичного*) *поиска*. При его применении количество просматриваемых записей при файле из N записей имеет порядок $\log_2 N$.

Индексный метод адресации использует специальную таблицу, называемую *индексом*, которая соотносит различным значениям ключа адреса соответствующих записей или блоков таких записей. Если таким образом идентифицируется лишь адрес блока, а внутри блока поиск делается последовательным сканированием, то метод именуется *индексно-последовательным*.

Наконец, еще один способ использует те или иные алгоритмы для преобразования ключа в адрес. Эти алгоритмы могут использовать специальные свойства ключа. Например, если ключ при-

нимает все значения от 1 до n, все записи имеют одинаковую длину l ячеек ЗУ и помещаются в последовательные ячейки, начиная с p-й, то адрес A_k записи с ключом k $(1 \leqslant k \leqslant n)$ вычисляется по формуле $A_k = p + l(k - 1)$.

Заметим, что подобный способ можно в принципе применять не только к числовым, но и к буквенным ключам, предварительно заменив буквы их цифровыми кодами. Однако при этом многим числовым значениям в диапазоне изменения ключа не будут соответствовать никакие используемые его значения. Поэтому простые формулы адресации типа только что рассмотренной дадут плохое заполнение памяти. Чтобы избежать такой ситуации, используется специальный прием, называемый *перемешиванием* (hashing). Его смысл заключается в подборе алгоритма, отображающего множество используемых значений ключа в множество адресов с достаточно плотным заполнением памяти (порядка 80%). Перемешиванием этот прием называется потому, что используемые им алгоритмы располагают записи ЗУ в случайном порядке (а не в порядке их расположения в файле).

К сожалению, подыскать достаточно простой алгоритм, дающий не только плотное, но и взаимно однозначное отображение f множества значений ключа в множество адресов, удается далеко не всегда. Поэтому может оказаться, что при перемешивании разным значениям ключа будет соответствовать один и тот же адрес. Чтобы выйти из затруднения, при перемешивании определяется адрес целого блока ячеек — так называемого *пакета*, в котором могут размещаться несколько записей. Если какой-либо пакет оказывается полностью заполненным, а в нем требуется поместить новые записи, то эти записи направляются в первый из свободных *пакетов переполнения*, адреса которых известны программе поиска.

Среди алгоритмов нахождения адреса A_k пакета, в который следует поместить запись со значением ключа, равным k, отметим один из наиболее простых и эффективных. Он основан на выборе числа m, не имеющего малых делителей (желательно простого) и вместе с тем близкого к выбранному числу n пакетов. Тогда адрес A_k определяется как остаток от деления k на m. Если l — число записей, размещаемых в одном пакете, а N — общее число записей в файле, то для хорошего заполнения памяти должно соблюдаться условие $nl \approx N$.

При поиске записи по ключу k поиск прежде всего производится в пакете, адрес A_k которого вычисляется с помощью правила: A_k есть остаток от деления k на m. Если записи с заданным значением ключа там не окажется, то производится последовательное сканирование пакетов переполнения. Подбирая должным образом параметры n и l, можно добиться того, что при достаточно хорошем заполнении памяти поиск большинства запи-

сей потребует лишь одного сканирования (без обращения к пакетам переполнения).

Рассмотрим еще некоторые особенности работы с файлами при размещении их в ЗУ различных типов.

6.3.1. Файлы на магнитных лентах. Как уже отмечалось выше, при адресации записей на магнитных лентах (МЛ) используется последовательное сканирование файла. Обмен между МЛ и оперативной памятью идет блоками (см. § 3.2), которые обычно содержат не одну, а несколько (иногда достаточно много) записей. Поиск записи в блоке, переданном в ОЗУ, может производиться как последовательным сканированием, так и описанными выше блочными методами. Актуализация файла на МЛ делается обычно достаточно большими порциями. Изменения накапливаются в специальном актуализационном файле и в соответствии с утвержденным расписанием переносятся в основной файл, который при этом фактически переписывается заново.

6.3.2. Файлы на магнитных дисках. Наиболее распространенной формой организации файлов на магнитных дисках (МД) является *индексно-последовательный метод*. Если файл размещается на одном цилиндре (см. п. 3.3.3), то одна из дорожек отдается под *индекс дорожек*. Под *область переполнения* отводится обычно также одна дорожка. На остальных дорожках организуются *блоки записей*, упорядоченные по ключу: например, на первой дорожке помещаются записи со значениями ключа от 1 до 120, на второй — от 120 до 300 и т. д. На дорожке записи размещаются в произвольном порядке.

В индексе дорожек хранятся граничные (минимальные или максимальные) значения ключа для каждой дорожки с указателями на соответствующие дорожки. Имеется также указатель на область переполнения.

Если файл подвержен частым изменениям, то целесообразно при его начальном формировании не заполнять полностью дорожки. Поэтому, когда файл пополняется новыми записями, специальная программа, сверяясь с индексом, направляет его на соответствующую дорожку. И лишь при отсутствии свободных мест на дорожке запись направляется в область переполнения. При обращении к записи по ее ключу сначала идет ее поиск (после обращения к индексу) на основной дорожке. В случае отсутствия требуемой записи поиск продолжается в области переполнения. Поскольку все операции выполняются в пределах одного цилиндра, перемещения блока головок не требуется. Переключение же дорожек (в одном цилиндре) осуществляется с помощью электронных схем с большой скоростью.

Если файл не помещается на одном цилиндре, то он продолжается (обычно на соседние цилиндры) в пределах данного *пакета дисков*, который, будучи заполнен данными, носит обычно

наименование *тома*. Заполнение цилиндров идет (как и в случае дорожек) в порядке роста (или убывания) ключа, по которому упорядочен файл. По аналогии с цилиндром в томе создается своя область переполнения и *индекс цилиндров*. Если файл выходит за пределы одного тома, то создается *индекс томов*, а в случае необходимости и еще одна дополнительная область переполнения.

Обращение к записям начинается со старшего индекса (с целью определения нужного тома) и продолжается далее по иерархии индексов, пока не будет найдена нужная дорожка. Поэтому указатели старших индексов адресуют следующие по старшинству индексы, и лишь самый младший индекс (индекс дорожек) адресует дорожки с записями. В случае, когда нужная запись на основной дорожке не найдена, происходит переход к сканированию областей переполнения в возрастающем порядке иерархии (сначала для цилиндра, затем для тома и, наконец, для множества томов).

Заметим, что в случае магнитных барабанов и дисков с фиксированными головками понятия тома и цилиндра совпадают. *Томом* называют также и магнитную ленту, заполненную информацией.

Актуализация файлов с индексно-последовательной организацией может производиться в истинном масштабе времени: при получении очередной новой записи (или изменении имеющейся) соответствующие коррективы вносятся в файл немедленно (обычно после окончания очередного цикла работы прикладных программ с файлом). При большой продолжительности времени существования файла, как и в случае файлов на магнитных лентах, могут производиться его периодические перестройки. Такая перестройка (выполняемая с помощью специальных программ) затрагивает, разумеется, не только сам файл, но и соответствующие ему индексы.

6.3.3. Файлы в ЗУ с произвольным доступом. Если размер файла позволяет разместить его в ЗУ с произвольным доступом, то он чаще всего организуется таким образом, чтобы была возможность прямой адресации записей по тому или иному ключу. Для этой цели может использоваться любой из описанных выше способов. Актуализация таких файлов производится, как правило, в истинном масштабе времени.

6.3.4. Файлы в ассоциативных ЗУ. Наиболее просто прямая адресация записей по ключу производится в ассоциативной памяти, которая специально приспособлена для этой цели. При использовании ассоциативных ЗУ нет необходимости прибегать к каким-либо алгоритмам преобразования ключа в адрес. Адресация (по шинам признака) производится непосредственно кодом значения ключа. До начала 80-х годов ассоциативная память для

операций с файлами не получила сколько-нибудь широкого распространения. Однако успехи микроэлектроники делают такую перспективу в ближайшем будущем достаточно реальной.

6.4. Организация иерархических файлов

В большинстве приложений наряду с последовательными файлами приходится иметь дело с *иерархическими файлами*, в которых записи связаны в древовидные структуры, описанные в § 1.8. Заметим, что даже отдельная запись, как, например, кадровая анкета, может рассматриваться как иерархическая структура. Для этого достаточно найти атрибут, состоящий из более мелких элементов. В нашем случае таким атрибутом может служить дата рождения, разлагающаяся на три элемента: год, месяц, число.

При размещении иерархического файла в ЗУ используется несколько разных способов. Одним из наиболее старых является последовательное размещение файла в виде так называемой *левосписковой структуры*. Ее смысл легко уяснить из следующего примера. Пусть необходимо разместить файл, изображенный на рис. 6.1. Тогда начинают с корня и идут по самой левой ветви, доходя, наконец, до листа. Затем слева направо проходят все листья, принадлежащие одному и тому же узлу B_1. После этого переходят к узлу B_2 и перечисляют его листья и т. д. В результате получают последовательность записей $AB_1C_1C_2C_3B_2C_4C_5B_3C_6C_7$.

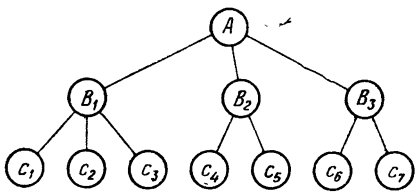

Рис. 6.1

Следующий способ — использование множественных *встроенных указателей*. В нашем случае это означает, что в записи A должны иметься указатели на записи B_1, B_2, B_3, в записи B_1 — на записи C_1, C_2, C_3 и т. д. Помимо такой системы указателей на *порожденные записи* могут использоваться также указатели на *исходные записи*: в записях C_1, C_2, C_3 — на B_1, в записи B_1 — на A и т. д. Наконец, наибольшее удобство при работе с иерархическими файлами достигается в случае дополнения систем указателей двух описанных типов указателями на *следующую подобную запись*. В нашем случае это означает, что запись B_1 снабжается указателем на B_2, B_2 — на B_3, C_1 — на C_2, C_2 — на C_3, C_4 — на C_5 и т. д.

В ряде случаев оказывается удобным вместо встроенных указателей использовать специальные *справочники связей*. В таком

справочнике помещаются только *идентификаторы записей*, т. е. их основные ключи, а также все необходимые указатели. Благодаря тому, что справочник достаточно компактен, его (или некоторую достаточно большую его часть) можно поместить в оперативную память и осуществлять поиск с большой скоростью.

Таблица 6.4

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
B_1	1	1	1	0	0	0	0
B_2	0	0	0	1	1	0	0
B_3	0	0	0	0	0	1	1

Помимо обычного способа задания связей в справочнике с помощью системы указателей иногда используются так называемые *битовые отображения* (или *битовые таблицы*). Смысл этого способа легко уяснить из конкретного представления битовым отображением связей между записями B_i, C_j иерархического файла (табл. 6.4), изображенного на рис. 6.1. Наличие связи от B_i к C_j характеризуется в таблице единицей на пересечении строки B_i со столбцом C_j. В случае же отсутствия связи на соответствующем пересечении стоит нуль. Битовые таблицы позволяют вести наиболее быстрый поиск связей в файле.

Заметим, что в случае необходимости прямого поиска записи по ее ключу используются те же самые механизмы, что и в случае последовательных файлов. Что же касается механизмов, описываемых в этом параграфе, то их назначение аналогично назначению последовательного сканирования записей в последовательных файлах.

6.5. Организация сетевых файлов

Иерархические файлы оказываются достаточными в большинстве приложений. Однако существуют важные приложения, в которых требуется обобщение этого понятия. Это имеет место, например, при организации управления на основе так называемых *сетевых графиков* (описываемых в последующих главах). Обобщение, о котором идет речь, снимает ограничение о наличии у каждой записи не более одной исходной записи и допускает *произвольные связи* между записями. Возникающие таким образом структуры называются *сетевыми файлами*. Простейший пример такого файла приведен на рис. 6.2.

При организации сетевых файлов используются уже описанные выше методы встроенных указателей, справочников и битовых отображений. Заметим также, что при введении *избыточности* за счет многократного повторения некоторых записей в базе данных сетевые структуры могут быть сведены к иерархическим. Такое сведение для файла, изображенного на рис. 6.2, показано на рис. 6.3. В рассматриваемом случае достаточно двукратного повторения записи 3.

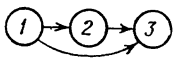

Рис. 6.2

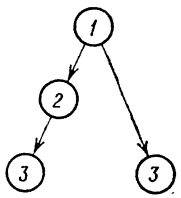

Рис. 6.3

Заметим, что в сетевых файлах довольно часто приходится *именовать связь*, а иногда и сопровождать дополнительными данными, называемыми *данными пересечения* двух записей, соединяемых этой связью.

Простейший пример подобных *именованных связей* дают родственные связи между людьми (отец — сын, дядя — племянник и т. п.). С подобной ситуацией приходится иметь дело не только в сетевых, но и в некоторых иерархических файлах. Отношения типа «женаты», «разведены» и др. могут сопровождаться данными о времени установления соответствующего отношения.

6.6. Языки для описания данных

Графическое представление связей в базе данных сохраняет простоту и наглядность лишь в том случае, когда представляемые схемы несложны. В случае сложных схем стрелки могут скорее запутать, чем помочь. Кроме того, при вводе описаний в ЭВМ двумерное графическое представление все равно заменяется одномерным (ввиду линейной структуры ЗУ). Поэтому для описания данных создаются специальные языки. Простейшие из них встраиваются в алгоритмические языки. Примерами могут служить средства для описания данных в алголе-60 и коболе, рассмотренные в предыдущей главе. Заметим, что средствами кобола описываются иерархические структуры данных.

В ряде программных систем для банков данных, получивших достаточно широкое распространение (американская система IMS, советская Ока и др.), используются лишь иерархические структуры данных. Языки описания данных в этих системах строятся по тому же принципу, что и раздел описания данных в языке кобол.

Американская ассоциация по банкам данных Codasyl предложила другой язык, позволяющий описывать не только иерархические, но и многие сетевые структуры. Основой языка codasyl является понятие *набора*. Набор представляет собою именован-

ное двухуровневое дерево. Его исходная запись (первый уровень) называется *владельцем набора*, а порожденные записи (второй уровень) — *членами* этого набора.

При описании данных, разумеется, приводятся не сами записи, а их *типы*, называемые также *схемами*. Примером двухуровневого набора может служить структура, в которой записью-владельцем является счет вкладчика сберкассы, а записями-членами — операции с этим счетом. В схемах записей указываются имена атрибутов, а также типы и форматы их значений (например, имя атрибута — фамилия, имя и отчество вкладчика, тип — алфавитный, формат — 30 букв). Описание схемы набора включает в себя имя набора, после чего объявляется и описывается сначала его владелец, а затем — члены.

Каждый из членов одного набора может быть объявлен владельцем другого набора. Продолжая этот процесс, можно, как нетрудно видеть, описать любую иерархическую структуру. Кроме того, не запрещаются и другие способы комбинаций наборов, благодаря чему создается возможность описывать непосредственно (без предварительного сведения их к иерархическим структурам) достаточно сложные сетевые структуры. Можно, например, создать два набора с одинаковыми членами, но разными владельцами, и т. п. В целом процесс описания схем данных с помощью наборов напоминает процесс сборки изделия из готовых конструктивных элементов.

Для указания *способов вхождения* в базу данных используется понятие так называемого *сингулярного набора*. Владельцем каждого такого набора объявляется *система* (имеется в виду операционная система ЭВМ, через которую прикладные программы взаимодействуют с базой данных). В отличие от других наборов, у которых схема набора и описываемый ею двухуровневый иерархический файл различны между собой (в схеме даны имена атрибутов, а в файле — их значения), для сингулярного набора схема и есть сам файл.

Смысл введения сингулярного набора состоит в следующем. Если прикладная программа хочет обратиться к какой-то записи, то для ее *полного именования* (с целью правильной адресации) следует использовать *последовательность имен записей*, начиная с члена одного из сингулярных наборов, которые связаны между собой отношением владелец — член. Например, если в базе данных записана информация об операциях по счетам вкладчиков не в одной, а в нескольких сберкассах, то простейшая схема структуры данных этой базы в принципе задается двумя наборами: в первом владельцем является сберкасса, а членом — вкладчик, во втором владелец — вкладчик, а член — операция. В языке codasyl к этим наборам нужно добавить набор, владельцем которого является система, а членом — сберкасса. Этим предопре-

деляется тот факт, что прикладная программа, желая обратиться к какой-то конкретной записи о той или иной операции, должна именовать это обращение последовательностью имен сберкассы, вкладчика и, наконец, операции. В противном случае обращение может оказаться неоднозначным.

Описание базы в языке codasyl начинается с объявления имени схемы. Далее следует объявление имени области, за которым следуют описания записей, принадлежащих этой области. В одну область проектировщик базы данных помещает те записи, которые, по его мнению, будут часто использоваться совместно. Используя подобное разбиение по областям, программы, управляющие размещением данных в памяти, выполняют это размещение так, чтобы данные, принадлежащие к одной области, размещались в ЗУ близко друг к другу (в смысле времени перехода). Описание записей делается средствами, близкими к коболу.

К каждому типу записи (или ее элементу) может быть приписан так называемый *замок секретности*, в качестве которого могут употребляться специальные кодовые слова — *пароль* или специальная прикладная программа (составляемая владельцем информации). Такая программа определяет некоторую процедуру, разрешающую доступ к данным (например, серию вопросов, на которые программа, обращающаяся к данным, должна дать правильные ответы). При неправильных ответах (в простейшем случае — при незнании пароля) доступ к данным будет закрыт. Замками секретности могут снабжаться также и другие части описаний: схемы, области, наборы (члены набора) и др.

В конце описания описываются наборы (связи между данными). Для каждого набора указывается его имя и указание на характер размещения его членов (они могут помещаться в память либо в случайном порядке, либо упорядочиваться по ключу, указываемому в описании члена каждого типа). Далее указывается имя владельца набора, а затем — имена членов (с указанием ключа).

6.6.1. Общие требования к языкам описания данных. Язык описания данных в общем случае должен позволять описывать различные типы элементарных данных и их форматы, а также именовать их. То же самое должно иметь место и в отношении вводимых типов структур данных (запись, файл, область и др.). Язык должен позволять специфицировать ключи и способ упорядочения данных по ключам. Он должен давать возможность определять любые отношения (связи) между записями (или другими группами данных), именовать эти отношения и, если необходимо, описывать характеризующие их данные. Кроме того, могут специфицироваться диапазоны возможных значений элементов данных и длины файлов. Язык также должен иметь возможность описывать замки секретности и другие средства *защиты данных*.

В случае системных применений полезно также иметь средства для описания *режимов актуализации данных*.

Все указанные возможности (которыми не располагает целиком ни один из разработанных пока языков) относятся к описанию *логических структур данных*, т. е. тех структур, в виде которых данные представляются пользователю. Для описания *физических структур данных*, т. е. их реального расположения в памяти и методов доступа, нужны дополнительные средства, которые (в неформализованном виде) частично уже приводились выше в разделах, посвященных организации файлов.

6.7. Языки манипулирования данными (ЯМД)

Такие языки должны давать прикладному программисту машинно-независимые средства работы с базами данных. Они дополняют обычные языки программирования новыми операторами (макрокомандами). Частично подобная задача решается современными операционными системами. Поэтому системы интерпретации языков манипулирования данными в банках данных можно (и должно) рассматривать как расширение операционной системы. Хороший ЯМД должен иметь возможность использовать для общения с базой данных все стороны имеющихся средств описания данных. Разработанные к настоящему времени ЯМД в полной мере этому условию не удовлетворяют.

Приведем в качестве примера список операторов ЯМД codasyl, представляющий собой расширения средств манипулирования данными, которые встроены в кобол.

Оператор open (открыть) открывает (т. е. делает доступным прикладной программе) файл или набор, имя которого указывается после имени оператора. Поскольку открытие наборами файла влечет за собой приведение в готовность соответствующего механизма доступа к нему (описанного в предыдущей главе), а этот механизм может потребоваться другим программам, то после исчерпания потребности в открытом файле (наборе) он закрывается оператором close (закрыть).

Оператор find (найти) находит в базе данных запись, имя которой приводится в составе оператора, и делает ее *текущей записью* программы. Оператор get (взять) передает в рабочую область программы значения указанных элементов поименованной записи.

Оператор modify (модифицировать) изменяет значение определенных элементов данных поименованной записи. Новые значения данных берутся при этом из рабочей области программы. Оператор insert (вставить) вставляет в указанный набор codasyl'а (в качестве нового члена) запись из рабочей области программы. Обратный оператор remove (удалить) удаляет из набора

поименованный экземпляр записи. Оператор store (заполнить) выполняет вставку в базу данных новой записи с *организацией* всех необходимых связей. В частности, таким образом может быть вставлена новая запись — владелец набора. Обратный оператор delete (устранить) удаляет из базы данных указанную запись и уничтожает все ее связи.

Оператор keep (удерживать) сохраняет настройку механизма доступа к записи с целью убыстрения повторных обращений. Обратный оператор free (освободить) отменяет действие оператора keep.

Наконец, с помощью оператора order (упорядочить) записи в поименованном файле или наборе упорядочиваются по указанному ключу в убывающей или возрастающей последовательности его значений.

6.8. Системы управления базами данных (СУБД)

Так именуется специальное (системное) программное обеспечение, которое позволяет прикладным программам работать с базами данных без знания конкретного способа размещения данных в памяти ЭВМ, т. е. *физической структуры данных*. Система управления при этом пользуется тремя видами описаний данных. С одной стороны — это описание *логической структуры данных* (сделанное прикладным программистом) в *исполняемой прикладной программе*. С другой стороны — это описание *логической структуры базы данных*, выполненное разработчиками этой базы (системными аналитиками и программистами). Наконец, третий вид описания — это описание *физической структуры базы данных*, также выполняемое системными аналитиками и программистами — разработчиками базы данных.

Заметим, что наряду с системными программистами в разработке баз данных должны, как правило, участвовать и *системные аналитики*, в задачу которых входит анализ предполагаемых способов использования проектируемых баз данных с целью *оптимизации* их логических и физических структур.

Программы, осуществляющие управление базой данных, обеспечивают в общем случае два интерфейса. Прежде всего это интерфейс между логической и физической структурами базы данных. Программы, обеспечивающие этот интерфейс, осуществляют преобразование логических имен данных в физические адреса, интерпретируют операторы языка манипулирования данными и осуществляют фактический обмен информацией между базой данных и рабочими областями прикладных программ. Последние две операции выполняются программным обеспечением базы данных совместно с операционной системой ЭВМ, на которой реализована эта база.

Второй интерфейс — это интерфейс между логической структурой данных, описанной в прикладной программе, и логической структурой самой базы. В банках данных первого поколения эта задача решалась весьма просто, поскольку прикладным программистам разрешалось пользоваться лишь *подсхемами* схемы базы данных. В банках данных второго поколения прикладным программистам разрешается использовать произвольные, удобные для них логические структуры данных (предварительно описав их на принятом в системе языке описания данных). Разумеется, эти структуры могут использовать лишь те элементы данных, которые содержатся в базе данных. Однако форматы записей и связи между данными могут очень сильно отличаться от стандартов, принятых в базе данных.

Поскольку в этом случае прикладной программист в языке манипулирования данными использует не стандартные, а свои собственные представления о логических структурах данных, требуется еще один уровень интерпретации операторов ЯМД. Эти задачи полностью ложатся на специальное программное обеспечение соответствующего банка данных. Методы установления интерфейса между различными логическими структурами данных (составленных из одних и тех же элементов) требуют привлечения новых средств. Один из наиболее перспективных подходов к решению такой задачи дают так называемые *реляционные структуры данных*, о которых будет рассказано ниже.

Помимо уже описанных задач, на программное обеспечение (ПО) банков данных возлагаются и другие функции. Среди них отметим прежде всего задачи обеспечения *секретности* и *сохранности* данных. ПО банка данных выполняет процедуры проверки *права на доступ* к данным, объявленным по тем или иным причинам секретными. О сущности этих процедур уже говорилось выше.

Заметим, что использование в качестве замков секретности не просто паролей, а программ, составляемых самими пользователями, позволяет пользователям конструировать сколь угодно хитрые замки. Например, поскольку операционная система ведет учет текущего времени (дат, часов, минут и секунд), то пользователь может предусмотреть в программе систему паролей, меняющихся в зависимости от времени.

В целом, машинная (безбумажная) форма хранения информации может обеспечить гораздо более совершенные и надежные системы обеспечения секретности, чем традиционная бумажная.

Помимо *защиты от неавторизованного доступа* к секретным данным важнейшее значение имеет защита любых (не обязательно секретных) данных от уничтожения или искажения. Подобная опасность может быть вызвана самыми различными причинами.

Во-первых, это случайные или преднамеренные *физические воздействия* (в основном с помощью электромагнитных полей) на носители (магнитные ленты, диски и др.), содержащие те или иные данные. Защита от таких воздействий предполагает специальные режимы хранения машинных носителей с данными, экранировку лентотек и дискотек и т. п. Кроме того, предусматривается необходимый уровень контрольного дублирования данных для возможности их восстановления на основных (рабочих) носителях в случае их уничтожения или порчи.

Второй возможный источник искажений данных — программы пользователей, которые могут по ошибке записать новые данные не в ту область памяти, в которую нужно. Чтобы избежать подобной опасности, вводится специальная система паролей, приводящаяся в действие только при записи. Не зная пароля, прикладная программа может читать (несекретные) данные, защищенные подобным паролем, но не может изменить их.

Процедуры защиты и *восстановления* данных приводятся в действие и контролируются специальным лицом — *администратором* банка данных, несущим всю полноту ответственности за сохранность находящейся в банке информации и соблюдение режима секретности. В число таких процедур, помимо уже описанных, могут включаться процедуры *периодической инвентаризации* содержимого банка с непременным соблюдением всех режимных требований. Для инвентаризационных и восстановительных процедур также создается специальное программное обеспечение.

Важное место в программном обеспечении банков данных занимают, как правило, процедуры *актуализации* (обновления) данных. В составе ПО банка такие процедуры могут отсутствовать лишь в том случае, когда используемая база данных является простой суммой баз пользователей. При таком положении задача актуализации базы данных возлагается на прикладные программы пользователей. В случае *системных применений* значительная часть базы данных (если не вся целиком) обеспечивается не зависящей от прикладных программ системой актуализации. Программное и административное обеспечение актуализации входит в этом случае в состав системы управления данных и попадает под юрисдикцию ее администратора.

Под *административным обеспечением* здесь понимаются специальные должностные инструкции для лиц, являющихся источниками актуализации, о порядке, сроках подачи и формах документирования актуализационной информации, а также о порядке ввода этой информации в систему. При больших размерах базы данных назначаются администраторы отдельных ее частей, несущие ответственность за сохранность подведомственных им частей, а также за своевременную и правильную их актуализацию.

Собственно обработка данных из баз данных осуществляется прикладными программами, составляемыми на алгоритмических языках. При этом для обеспечения взаимодействия прикладных программ и СУБД должны создаваться *интерфейсы* между соответствующими системами программирования и СУБД, на которые возлагаются функции по согласованию программной связи и связи по данным. Такие интерфейсы чаще создаются для систем программирования, ориентированных на обработку данных, таких, как кобол, PL-1 и др., и обычно включаются в СУБД.

6.9. Реляционные базы данных

Реляционный подход к описанию структур данных основывается на использовании для их описания языка предикатов (см. § 1.5), или, как в этом случае принято говорить, произвольных (n-местных) *отношений*, причем отношений только между элементарными данными. Любое подобное отношение может быть представлено *двумерной таблицей* (табл. 6.5).

Таблица 6.5

a	b	c	...	x
a_1	b_1	c_1	...	x_1
a_2	b_2	c_2	...	x_2
a_3	b_3	c_3	...	x_3
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
a_n	b_n	c_n	...	x_n

На этой таблице представлено отношение с именем A над элементарными данными с именами a, b, c, ..., x. Каждая строка таблицы представляет собой набор значений этих данных, находящихся между собой в данном отношении (иными словами, отношение A для них *истинно*). Строки таблицы принято называть *кортежами*, а столбцы — *доменами* (используется также термин *атрибут*). Если количество столбцов (или, что то же самое — длина кортежей) равно m, то говорят, что отношение A имеет *степень* m или является m-местным (m-арным) отношением.

Считая кортежи отдельными записями, легко представить заданное отношение (таблицу) в виде простейшего последовательного файла. Организация таких файлов в ЭВМ была подробно изложена в § 6.3.

Главная идея реляционного подхода состоит в том, чтобы представлять произвольные структуры данных в виде совокупностей описанных таблиц (отношений). Процесс такого представления (точнее, переход от других форм представления к описанному) называется *нормализацией*, а само представление — *реляционной структурой*. Если на отношения реляционной структуры не накладывается никаких ограничений, кроме отсутствия одинаковых строк в таблице, то говорят, что она представлена в *первой нормальной форме*.

Очевидным способом может быть проведена нормализация любого иерархического файла, записями которого являются элементарные данные. Для этого достаточно «размножить» узлы файла так, как это показано на рис. 6.4.

После такого размножения фактически уже получено представление файла в виде требуемой таблицы. Достаточно лишь именовать домены и присвоить таблице имя соответствующего файла.

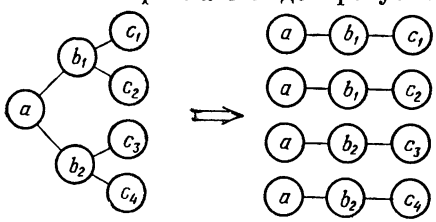

Рис. 6.4

Если записи сами представляют собой иерархические структуры, то тот же самый процесс «размножения» узлов можно применить и к ним. В результате, как легко видеть, файл будет приведен к виду плоской таблицы. В случае сетевых структур тем же самым приемом «размножения» узлов они приводятся сначала к иерархической, а затем и к реляционной (табличной) форме.

Приведенные приемы нормализации очень просты, но приводят к большой избыточности в логическом представлении данных, которая, однако, не означает избыточности хранения этих данных. Однако наибольшая неприятность при этом состоит в том, что актуализация таких отношений может привести к так называемым *аномалиям манипулирования*, когда происходит разрушение и/или потеря данных. Действительно, если в отношении, приведенном в табл. 6.6, исключить выделенную запись, то

Таблица 6.6

Ф.И.О.	Образование	Шифр законченного учебного заведения	Адрес учебного заведения
Иванов И. И.	высшее	КГУ	г. Киев

это приведет к потере информации об адресе учебного заведения КГУ, если в отношении нет больше записей с данным шифром учебного заведения. Чтобы не допустить возникновения аномалий, прибегают к декомпозиции отношений, о чем подробнее говорится ниже. Для рассмотренного примера отношение целесообразно разбить на два, как показано ниже (табл. 6.7 и 6.8).

К недостаткам можно отнести также и то, что результатом нормализации является отношение высокой степени, в то время

как на практике могут представлять особый интерес и даже требовать специального именования отношения меньшей степени *). Поэтому процесс нормализации обычно продолжается дальше в направлении *расщепления* сложных (многоместных) отношений на более простые.

Таблица 6.7

Ф.И.О.	Образование	Шифр законченного учебного заведения
Иванов И. И.	высшее	КГУ

Таблица 6.8

Шифр учебного заведения	Адрес учебного заведения
КГУ	г. Киев

6.9.1. Алгебра отношений. Огромным преимуществом реляционного представления структур данных является то, что на множестве отношений легко определяются различные операции, превращающие это множество в *алгебру отношений* (*реляционную алгебру*). Прежде всего можно, выделяя какую-либо часть доменов (столбцов) данного отношения, получить отношение меньшей степени. При выполнении этой операции, называемой *проекцией*, предполагается, что в получающейся таблице устраняются возможные повторения строк.

Операция *объединения* (соединения), обозначаемая звездочкой (*), позволяет «склеить» несколько отношений в одно истинное тогда и только тогда, когда истинны все объединяемые отношения. При этом помимо имен объединяемых отношений должно быть указано, какие домены из этих отношений нужно включить в результирующее отношение. Заметим, что при такой операции число строк может уменьшиться и даже свестись к нулю (что означает получение в результате *пустого* отношения). Например, при объединении отношения A, задаваемого кортежами $(a_1b_1c_1)$, $(a_2b_2c_1)$, $(a_3b_1c_1)$, и отношения B с кортежами (b_1c_2), (b_2c_1) по всем трем доменам, которые мы обозначим через a, b, c, возникнет отношение C с единственным кортежем $(a_2b_2c_1)$. Символически это может быть выражено формулой $A * B(a, b, c) = C$.

Возможны и другие операции над отношениями. Если в состав программного обеспечения реляционной базы данных включить программы, которые автоматизируют выполнение этих операций, то они могут осуществить отображение одной логической структуры данных в другую. При этом, разумеется, мы предпо-

*) Этот недостаток иногда оборачивается и преимуществом, поскольку одними лишь бинарными отношениями, используемыми иерархическими и сетевыми структурами, нельзя выразить всех требуемых отношений.

лагаем, что обе структуры описаны в виде схем отношений (таблиц). Программы перевода производят заполнение таблиц второй структуры, используя таблицы первой структуры. Тем самым автоматизируется интерфейс между различными логическими структурами данных, т. е. решается задача, о которой говорилось в § 6.8.

На администрацию системы при этом ложится задача задания (в описании данных) формул реляционной алгебры, с помощью которых из отношений логической структуры базы данных получаются отношения структуры, используемой прикладным программистом. Более высокий уровень автоматизации интерфейса предполагает использование лишь описаний связываемых структур данных на специальном языке описания данных, расширенном с помощью так называемого *исчисления отношений* (*реляционного исчисления*). В этом случае система сама определяет пути реализации такого «описательного» отображения.

6.9.2. Исчисление отношений. В реляционном исчислении главным средством описания новых отношений исходной структуры является символическое обозначение вида $A(x_1 \cdot y_1, x_2 \cdot y_2, \ldots)$. Оно означает отношение с именем A, полученное в результате проекции кортежей, удовлетворяющих булевскому выражению B, на домены x_i, y_i, где x_i — имя отношения (исходной структуры), y_i — имя домена из этого отношения. Различают *квантор существования* $\exists$ (сокращение для слова «существует») и *квантор всеобщности* $\forall$ (сокращение для слов «для всякого»). Двоеточие (:) используется для сокращенного представления слов «такой, что». В качестве примера задания отношения в языке реляционного исчисления рассмотрим выражение

A(К. ФИО, К. зарплата) : К. цех = 3 зарплата 150.

Оно задает таблицу (отношение) с именем А с двумя доменами из таблицы (отношения) К (кадры). Первый домен содержит фамилии, имена и отчества рабочих, второй — размер месячной зарплаты. При этом в таблицу А включаются только рабочие третьего цеха с размером зарплаты, большей, чем 150 рублей в месяц.

6.9.3. Вторая и третья нормальные формы. Если в отношении какой-либо атрибут или группа атрибутов X полностью определяют другой атрибут (или группу атрибутов) Y, то говорят, что Y *функционально зависит* от X. Слова «полностью определяют» означают, что любому значению X соответствует лишь одно, а не несколько значений Y. Атрибут (или группа атрибутов) Y называется *полно зависимым* от группы атрибутов X, если Y функционально зависит только от всего множества X, а не от какого-либо его собственного подмножества.

Назовем *возможным ключом отношения* атрибут или группу атрибутов, значения которых однозначно идентифицируют строку

задающей отношение таблицы. Атрибут, входящий в состав какого-либо возможного ключа, будем называть *основным* (*первичным*).

Говорят, что отношение задано во *второй нормальной форме*, если оно находится в первой нормальной форме и каждый атрибут, не являющийся основным, полно зависит от любого возможного (вообще говоря, составного) ключа в этом отношении.

Если любой такой ключ является *простым ключом* и X не зависит функционально от Y или Y — от Z, говорят о *неполной транзитивной зависимости* Z от X.

Преобразование второй нормальной формы в *третью нормальную форму* состоит в *расщеплении* отношения R на два отношения R и Q с атрибутами X, Y и Y, Z.

В качестве примера рассмотрим отношение R с тремя атрибутами: номер работника, номер выполняемого им задания, расчетный срок выполнения задания. Здесь третий атрибут неполно транзитивно зависит от первого атрибута, поскольку один и тот же работник может выполнять несколько заданий. Поэтому после приведения отношения R к третьей нормальной форме представляющая его таблица распадется на две. В первой таблице номеру работника соотносятся номера выполняемых им заданий, во второй — номеру задания соотносится расчетный срок его выполнения.

В программное обеспечение реляционных баз данных обычно включаются программы для автоматической нормализации. Значит, при разработке базы данных достаточно задать все необходимые атрибуты и отношения между ними в произвольной форме. ПО базы данных выполняет после этого приведение к третьей нормальной форме, обращаясь к разработчику лишь для *согласования имен*, возникающих в процессе нормализации *новых отношений*. Что же касается логической структуры базы данных, то наиболее удобно оказывается использовать для нее третью нормальную форму.

6.9.4. Проектирование реляционных баз данных. Проектирование реляционных баз данных основывается на трех принципах: эквивалентных преобразований схемы базы данных, уничтожения аномалий манипулирования данными и минимальной избыточности данных. В общем случае при проектировании ставится задача получения такой схемы базы данных, которая в определенном смысле эквивалентна и лучше исходной.

В настоящее время известны два подхода к определению понятия эквивалентности схем баз данных: эквивалентность по зависимостям и эквивалентность по данным. В определении *эквивалентности по зависимостям* требуется, чтобы схемы содержали одни и те же зависимости: если Γ_1 и Γ_2 — множества зависимостей двух схем, то эти схемы эквивалентны по зависимостям, при

$\Gamma_1^+ = \Gamma_2^+$, где Γ^+ — замыкание Γ по зависимостям. В определении *эквивалентности по данным* требуется, чтобы обе базы данных содержали одни и те же данные: две базы данных со схемами S_i, $i = \overline{s, k}$ и T_j, $j = \overline{1, n}$, соответственно содержат одни и те же данные, если $S_1 * S_2 * \ldots * S_k = T_1 * T_2 * \ldots * T_r$, где $*$ — операция естественного соединения отношений.

Как уже отмечалось, одной из целей проектирования реляционных баз данных является уничтожение аномалий манипулирования данными. Существенный вклад в решение этой проблемы дала теория нормализации баз данных. Введенная Коддом третья нормальная форма позволила существенно уменьшить (но не исключить полностью) аномалии манипулирования. При этом важно отметить, что третья нормальная форма сохраняет эквивалентность как по зависимостям, так и по данным. В теории нормализации рассмотрены и другие нормальные формы, в том числе полностью исключающие аномалии манипулирования, однако, каждая из них либо теряет эквивалентность по данным или зависимостям, либо допускает аномалии.

Как и в случае эквивалентности, можно дать два определения избыточности: избыточность по зависимостям и избыточность по данным. Система отношений S с множеством зависимостей Γ *избыточна по зависимостям* Γ_i, если $\left(\bigcup_{j=1}^{n} \Gamma_j\right)^+ = \left(\bigcup_{\substack{j=1 \\ j \neq i}}^{n} \Gamma_j\right)^+$.

Система S *избыточна по данным* отношения S_i, если $S_1 * S_2 * \ldots * S_n = S_1 * S_2 * \ldots * S_{i-1} * S_{i+1} * \ldots * S_n$. Известно, что для любого отношения существует такая третья нормальная форма, которая не избыточна по функциональным зависимостям.

В связи с двумя подходами к определению эквивалентных преобразований схем баз данных существуют два метода *проектирования схем*: синтез и декомпозиция. *Синтез* предполагает сохранение эквивалентности по зависимости, а *декомпозиция* — по данным. В алгоритмах синтеза достижима только третья нормальная форма, в то время как в алгоритмах декомпозиции — любые. При синтезе нельзя получить такие отношения, в которых полностью снимались бы аномалии манипулирования, в то время как при декомпозиции можно. В алгоритмах синтеза можно достичь минимальной избыточности по функциональным зависимостям, а в алгоритмах декомпозиции — по данным.

Отсутствие единого решения проблемы проектирования реляционных баз данных в рассмотренной постановке приводит к необходимости на практике определять и отслеживать возможные нарушения в процессе ведения и преобразования баз данных с целью их предотвращения.

6.9.5. Преимущества и недостатки реляционных структур. Помимо уже разобранного выше преимущества, заключающегося в

возможности для различных прикладных программ использовать различные представления логической структуры данных, реляционные структуры обладают и рядом других преимуществ. Это, во-первых, простота понимания и работы с базой (особенно для не очень опытных пользователей), поскольку большинство людей имеет практику работы с двумерными таблицами. Следует особо подчеркнуть, что графическое представление структур данных выглядит ясным и наглядным лишь для структур малой сложности: при большой сложности структур представление их в виде таблиц обеспечивает бо́льшую ясность и точность.

Наряду с простотой представления данных, работа с реляционными базами данных поддерживается языками манипулирования высокого уровня (уровня исчислений), избавляющими пользователя от необходимости программирования поисковых процедур на базах данных. Для этого достаточно правильно указать в запросе, что необходимо найти, а не как найти. Этим достигается полное исключение необходимости прикладного программирования поисковых процедур при работе с базами данных.

Во-вторых, реляционную базу проще развивать и дополнять. Процедуры актуализации также, как правило, упрощаются. В-третьих, упрощается контроль секретности — секретные части отношений всегда могут быть выделены и изолированы от остальных (несекретных) частей. В-четвертых, упрощается физическая организация данных и ее интерфейс с логической структурой, благодаря чему банк данных легче адаптируется к возможным изменениям его технической базы.

Пятое преимущество состоит в том, что в реляционной структуре естественным образом выражаются отношения любой степени, а не только бинарные отношения, как это имеет место в случае иерархических и сетевых структур. Это преимущество становится сразу ощутимым даже в том случае, когда в структуре данных требуется выразить, например, множественное отношение, которое выделяет из списка людей отдельные семьи или какие-либо другие коллективы. Его задание возможно и в иерархических (а тем более сетевых) структурах, однако оно требует введения дополнительных записей и соответствующих ссылок.

Наконец, одно из важнейших преимуществ реляционных структур состоит в том, что отношения являются строго определенным математическим понятием и поэтому могут служить объектом строгой математической теории. Сюда же относится и уже описанная выше возможность введения алгебраических операций над отношениями, что придает реляционным структурам особую гибкость и возможность относительно простой адаптации к самым разнообразным требованиям.

К числу недостатков реляционных структур можно отнести большую сложность программного обеспечения логических ин-

терфейсов, особенно при использовании реляционного исчисления. Определенный недостаток состоит также и в некоторой избыточности двумерных таблиц по сравнению с иерархическими и особенно сетевыми структурами. Эта избыточность наглядно иллюстрируется на рис. 6.4. Правда, в процессе нормализации расщепление отношений может заметно уменьшить избыточность представления и тем самым улучшить использование памяти.

6.10. Целостность баз данных

Выше (в § 6.8) мы уже познакомились с понятием *сохранности* базы данных. Понятие *целостности* представляет собой дальнейшее развитие этого понятия. Дело в том, что в системных применениях база данных, как правило, описывает некоторый реальный объект (или группу объектов). Иными словами, базы данных в этом случае выступают в качестве *информационных моделей* реальных объектов. Объекты же в большинстве случаев являются не статическими, а *динамическими*. Иными словами, они, а следовательно, и описывающие их данные меняются с течением времени. Отслеживание этих изменений в базе данных составляет сущность процесса ее *актуализации*, о чем уже не раз говорилось выше.

Если режимы актуализации различных частей базы данных не взаимоувязаны в *единую систему*, может возникнуть ситуация, когда данные, описывающие реальный объект, относятся к *разным моментам времени*. Иногда в этом может не быть никакой особой беды. Однако во многих случаях это ведет к серьезным искажениям и даже к появлению информационных моделей объектов, которые в действительности просто не могут существовать.

Простейший случай искажения возникает, например, при изменении стажа, разряда или должности. Если соответствующее изменение имело место и даже внесено в кадровый файл, но не внесено в файл, которым пользуется бухгалтерия при начислении зарплаты, то может возникнуть случай неправильного начисления зарплаты.

Более сложный пример дает практика автоматизированного проектирования в диалоговом режиме со многими проектантами, работающими на общей информационной базе. Предположим, что в процессе проектирования, скажем завода, один проектант переместил в определенное место какую-то единицу оборудования, а другой проектант еще не убрал с этого места «свое» оборудование. В этом случае база данных в соответствующий момент времени внутренне противоречива, ибо представляет объект, который в действительности не может существовать.

В обоих приведенных примерах нарушается *целостность базы данных*. Не вдаваясь далее в детализацию этого понятия, заме-

тим, что наиболее полное понятие целостности базы данных основывается на глубоком, содержательном знании природы отображаемого ею объекта. Поэтому в общем случае задача автоматического контроля целостности базы данных и, тем более, восстановления целостности после ее случайной утери связана с задачей создания искусственного интеллекта и пока еще далека от полного решения.

Сказанное, разумеется, не исключает наличия автоматического контроля и исправления простых нарушений целостности баз данных того типа, который приведен в первом из двух рассмотренных выше примеров. В случае, когда для автоматического контроля и исправления некоторых нарушений целостности базы данных используются элементы содержательного знания об описываемом ею объекте, мы будем говорить о наличии элементов «интеллектуальности» этой базы данных (точнее — об интеллектуальности ее программного обеспечения).

6.11. Телекоммуникационные методы доступа

При работе с базами данных или с отдельными файлами с удаленных терминалов, помимо традиционных задач организации доступа, возникают многие дополнительные задачи. Необходимо организовать управление передачей, приемом и анализом сообщений, циркулирующих между ЭВМ и терминалами, проверять доступность и занятость не только терминалов, но и линий связи и т. д. Программы, обеспечивающие *телекоммуникационный доступ*, организуют опрос терминалов о наличии у них сообщений для передачи и определяют порядок такой передачи. В случае *режима соперничества*, когда готовые для передачи терминалы борются за монопольное обладание линией связи, программы теледоступа следят за тем, чтобы не оставлять в течение сколько-нибудь длительного времени те или иные терминалы необслуженными.

Методы теледоступа могут использовать ЭВМ для коммутации сообщений, которыми обмениваются друг с другом удаленные терминалы. Они обычно также дают возможность *дистанционного управления вводом заданий* (программы ДУВЗ), так что с удаленных терминалов оказывается возможным решать большие задачи, программы и данные для которых были заранее предоставлены в распоряжение ЭВМ.

Методы теледоступа, обеспечиваемые стандартными ОС, предназначены в основном для позадачного подхода. При использовании баз данных дополнительно возникает задача установления интерфейса между СУБД и программами, обеспечивающими теледоступ. Задача состоит в организации так называемых *транзакций*, т. е. единичных актов взаимодействия в системе «база

данных — передача данных» (БД — ПД), выполняемых по запросам удаленных пользователей. Необходимость обрабатывать поток запросов в *реальном масштабе времени* отличает данную ситуацию от обычной ситуации обеспечения интерфейса между базой данных и прикладными программами, которые работают в *пакетном режиме* (и потому могут спокойно ждать своей очереди). Следует особо подчеркнуть, что различные транзакции могут обслуживаться различными прикладными программами, работой которых также приходится управлять в реальном масштабе времени.

Подобный *системный теледоступ* позволяет создавать информационно-справочные системы (с удаленными и близкими терминалами), работающие в реальном масштабе времени, осуществлять быструю коммутацию сообщений между терминалами и др. Для ЕС ЭВМ функции системного теледоступа осуществляет программный комплекс Кама, работающий совместно с ОС ЕС и СУБД Ока.

6.12. Первичные и вторичные файлы в базах данных

При выборе форм представления файлов в базах данных возникает естественное противоречие между простотой и скоростью их актуализации, с одной стороны, и простотой и быстродействием их интерфейса с прикладными программами — с другой. Простота актуализации файла требует, чтобы структура его записей соответствовала принятым формам первичного учета. Например, записи в кадровом файле должны в свете этого требования содержать все атрибуты, принятые в утвержденной форме личного листка по учету кадров. В файле, описывающем оборудование, записи должны отражать принятые формы паспорта на отдельные единицы оборудования и т. д. Подобные файлы, подогнанные к задачам первичного учета, мы будем называть *первичными файлами*. При полноте учета первичные файлы составляют *полную информационную модель* характеризуемого ими объекта.

В то же время различные прикладные программы могут пользоваться только небольшой частью данных, содержащихся в том или ином конкретном первичном файле. Например, программе для начисления зарплаты сотрудникам не нужно большинство атрибутов кадровых анкет. То же самое можно сказать о программе планирования сменных заданий (доводимых до каждого рабочего) и др. Разумеется, СУБД могут снабжать прикладные программы данными непосредственно из первичных файлов. Ясно, однако, что при этом каналы обмена ВЗУ с ОЗУ будут загружены передачей большого числа лишних данных, потребуется

неоправданный расход ячеек ОЗУ под буферы, снизится быстродействие интерфейса.

Поэтому в базах данных наряду с первичными файлами часто целесообразно иметь *вторичные файлы*, формы записей которых, а также способ упорядочения записей в файле ориентированы не на *первичный учет*, а на *прикладные программы*. Например, для задач планирования сменных заданий записи вторичного кадрового массива должны кроме учетного номера и имени рабочего содержать лишь данные о его квалификации и, возможно, стаже работы. При этом порядок следования записей в файле должен отражать принятую организационную структуру, т. е. распределение рабочих по цехам, участкам, бригадам и пр. С таким файлом задачи планирования будут решаться гораздо эффективнее как с точки зрения скорости работы, так и с точки зрения расходования ресурсов ЭВМ (прежде всего памяти и каналов).

Разумеется, вторичные файлы можно включить в базу данных, полностью уравняв их в правах (т. е. в организации процесса актуализации и интерфейса с прикладными программами) с первичными файлами. Однако на практике в большинстве случаев оказывается удобным поступать иным способом. Суть его состоит в том, что первичные файлы объединяются в особую (первичную) базу данных. Актуализация этих файлов приводится в соответствие с процедурами первичного учета в системе. В идеале такая актуализация ведется в *истинном масштабе времени*. Иными словами, изменения, поступающие с рабочих мест первичного учета, сразу же вносятся в соответствующие первичные файлы. На практике, разумеется, обычно довольствуются компромиссным решением: поступающие изменения накапливаются в специальных буферах и по мере их наполнения разносятся по соответствующим (первичным) файлам. Процесс «разнесения» поступивших изменений по файлам может регулироваться либо жестко (в установленное время), либо в зависимости от числа поступивших изменений. Система актуализации первичной базы данных обеспечивает тем самым главное ее назначение — быть *информационным портретом* (с той или иной разумной задержкой во времени) характеризуемого этой базой реального объекта.

Вторичные файлы могут существовать в виде *статических* (постоянно существующих) и *динамических* (возникающих лишь на нужный период времени) *файлов*. Процесс актуализации статических файлов и создания динамических файлов управляется не динамикой процессов первичного учета, а динамикой выхода «на счет» (исполнение) работающих с этими файлами прикладных программ. При этом используется то обстоятельство, что в большинстве системных применений ЭВМ выход на счет прикладных программ представляет собой не полностью *случайный*,

а в определенной мере *детерминированный*, упорядоченный процесс. Закономерности этого процесса используются специализированной частью операционной системы, расширяющей систему управления заданиями в обычных ОС (рассчитанных на случайный поток заданий). Результатом такого использования является *заблаговременная подготовка* (актуализация или создание заново) к нужному моменту необходимых вторичных файлов. Подобная специализированная ОС занимает промежуточное положение между обычными (пакетными) ОС и ОС реального времени.

Тем самым вторичные файлы как бы выделяются в особую (вторичную) базу данных, отличающуюся от первичной базы данных прежде всего способом организации процесса актуализации *). При этом первичные файлы используются в качестве своеобразных эталонов, по которым выверяются (или создаются заново) вторичные файлы. Так что программы актуализации вторичной базы данных могут строиться с помощью СУБД первичной базы, дополненной в случае необходимости программами «урезания» и переупорядочивания (*сортировки*). Ситуация здесь в какой-то мере напоминает организацию измерений на производстве: непосредственно для измерений употребляются *рабочие* приборы и инструменты, для контроля и настройки которых служит специальная *эталонная* аппаратура.

Способ ведения вторичных файлов (статический или динамический) определяется конкретными обстоятельствами. Часто обновляемые, но редко используемые вторичные файлы целесообразно иметь в динамическом виде, редко же обновляемые, но часто используемые — в статическом виде. Выбор одного из этих способов осуществляется обычно в результате моделирования проектируемой системы обработки данных.

Кроме того, в процессе работы системы, использующей банк данных, можно осуществлять те или иные процедуры статистических оценок частоты обращения к различным элементам данных. Полученные оценки используются для оперативной перестройки вторичной базы данных с целью повышения эффективности системы, использующей эту базу.

Практика работы современных информационно-справочных систем и систем оперативной обработки данных (в частности, автоматизированных систем управления) показывает, что обычно более 80% запросов относятся менее чем к 20% данных, находящихся в первичной базе данных.

---

*) Создание динамических файлов можно трактовать как частный случай актуализации, когда актуализируемый файл в начале процесса актуализации был пуст.

6.13. Распределенные базы данных

До сих пор рассматривались базы данных и СУБД, размещаемые и работающие на одной ЭВМ. Такие базы данных и СУБД называют *локальными*. Практически все чаще возникает необходимость в решении задач с *распределенными базами данных*, т. е. таких задач, для которых данные размещены по месту своего возникновения или наиболее эффективного использования, на ЭВМ, удаленных друг от друга на большие расстояния. При этом предполагается, что на каждой из ЭВМ данные управляются своими локальными СУБД.

Основу для решения таких задач составляют *сети передачи данных*, позволяющие соединять по каналам связи различные ЭВМ, обеспечивать техническую и программную поддержку обмена данными между ними, образуя тем самым *сеть* ЭВМ.

Обработка распределенных данных обычно строится так, что функции по выдаче в сеть транзакций на обмен данными с локализацией нужной ЭВМ сети и локальных баз данных возлагаются на обрабатывающую программу. Это, с одной стороны, значительно усложняет процесс прикладного программирования и, с другой — делает прикладные программы зависимыми от локализации данных в сети. С целью устранения этих недостатков при работе с распределенными данными создаются *системы управления распределенными базами данных* (СУРБД). Они строятся таким образом, чтобы максимально обеспечить соблюдение принципа независимости прикладных программ от локализации данных в сети, при котором логическое представление распределенной базы данных и манипулирование данными для прикладной программы ничем не отличаются от соответствующего локального варианта базы. Такие СУРБД оснащены каталогами, в которых хранятся структура сети, информация о локальных СУРБД и базах данных, а также программным обеспечением, которое на основе этой информации управляет взаимодействием прикладной программы и конкретной локальной базой данных сети.

Сложность управления распределенными базами данных во многом зависит от того, поддерживаются ли они однотипными локальными СУРБД, взаимодействие между которыми осуществляется просто, так как для этого используется один общий язык. Если же, например, часть распределенной базы данных находится под управлением СУРБД сетевого типа, а другая — под управлением СУРБД иерархического типа, то вопрос управления такой распределенной базой существенно усложняется. Один из путей разрешения возникающей сложности состоит в использовании некоторой промежуточной интерфейсной СУРБД (чаще всего реляционного типа), через которую на основе соответствующих отображений обеспечивается интерфейс между СУРБД.

Глава VII

МНОГОМАШИННЫЕ КОМПЛЕКСЫ И СЕТИ ЭВМ

7.1. Многомашинные комплексы

В практике применения ЭВМ их нередко приходится объединять в *комплексы*, состоящие из двух и более машин. Необходимость в таком объединении вызывается различными обстоятельствами. Прежде всего, довольно часто в случае особенно ответственных применений (как правило, в режиме реального времени) приходится иметь так называемый *горячий резерв*. В этом случае чаще всего используются две совершенно идентичные ЭВМ, подключаемые через коммутатор к обслуживаемому объекту. В качестве последнего может выступать любой производственный объект, управляемый при помощи ЭВМ, терминальная сеть оперативной информационной службы и др.

В обе ЭВМ вводится необходимый набор программ и баз данных. Оперативная информация, поступающая от объекта в случае, когда она не меняет баз данных (например, запросы в информационную систему), направляется для обработки в первую ЭВМ. Вторая машина находится во включенном («горячем») состоянии, но оперативную информацию не обрабатывает. В случае выхода из строя первой ЭВМ поток оперативной информации немедленно направляется на вторую машину так, чтобы обслуживаемый объект не почувствовал произведенной подмены.

Для большинства ЭВМ помимо остановок, вызванных выходом машины из строя, предусматриваются периодические плановые остановки для профилактических осмотров. Чтобы и в этом случае всегда иметь готовый к действию горячий резерв, в комплекс может быть включено не две, а три (и даже более) машины. При этом комплекс остается *однородным*, т. е. составленным из одинаковых машин.

Необходимость в *неоднородных* вычислительных комплексах может возникнуть в системах, соединяющих в себе несколько различных функций. Например, одна или несколько машин могут выполнять в системе функции приема и передачи информации в линии связи, тогда как другие машины будут специализировать-

ся на обработке поступающей информации. В так называемых *интегрированных системах* управления сложными объектами необходимость создания многомашинных комплексов обуславливается логикой самого управления. Отдельные ЭВМ, управляя отдельными составными частями объекта, нуждаются в обмене данными и *координации управления* в интересах всего объекта в целом. Как правило, в этом случае приходится строить неоднородные комплексы, хотя иногда они могут свестись к однородному случаю.

Наконец, необходимость в комплексировании ЭВМ может возникнуть не только в режиме реального времени, но и в режиме пакетной обработки при решении особо сложных задач. Здесь также могут использоваться как однородные, так и неоднородные комплексы.

Во всех случаях при создании многомашинных комплексов приходится решать две основные задачи. Во-первых, это организация *информационного обмена* между отдельными ЭВМ, входящими в комплекс, а также между ними и внешним миром. Вторая задача — это организация *управления комплексом*.

Имеется несколько способов организации информационного обмена между отдельными ЭВМ в комплексе. Наиболее радикальный из них — организация общих информационных шин, к которым подсоединяются центральные и канальные процессоры, а также блоки ОЗУ всех машин с возможностью организации произвольных связей между любой парой этих устройств. При этом мы получаем уже фактически не комплекс, а *мультипроцессорную вычислительную систему*. Реализация подобного объединения ЭВМ требует глубоких изменений их операционных систем. Кроме того, в случае неоднородного комплекса необходимы достаточно сложные согласователи интерфейсов, позволяющие центральным процессорам ЭВМ одних типов обращаться к ОЗУ и каналам ЭВМ других типов. В случае объединения ЭВМ, сильно разнящихся друг от друга, работа по столь глубокому их объединению сравнима по сложности с разработкой новой вычислительной системы и потому практически малооправданна.

Более практичным является такой способ, когда ЭВМ объединяются через специально подсоединяемый к ним всем блок (или несколько блоков) ОЗУ. Такой блок служит как бы своеобразным «почтовым ящиком», куда каждая ЭВМ может засылать *сообщение*, адресованное любой другой ЭВМ, и считывать сообщения, адресованные ей самой.

В этом случае также требуется определенная доделка операционных систем и организация интерфейсов с «почтовым ящиком». Однако объем этих доделок существенно меньше, чем в предыдущем случае, благодаря чему указанный метод чаще находит применение на практике.

Еще более простым является установление связей между ЭВМ через их *канальные процессоры*. Особенно просто эта задача решается при объединении двух ЭВМ. Для этого достаточно соединить какой-либо канал одной машины с каким-либо каналом другой через так называемый *адаптер канал — канал*. Такое соединение дает возможность для каждой ЭВМ обращаться к ОЗУ другой ЭВМ тем же способом, каким они осуществляют обращение к своим собственным устройствам внешней памяти. Этот способ требует минимальных доделок операционных систем объединяемых машин и потому применяется на практике наиболее часто. При объединении таким способом нескольких машин выделенные каналы подсоединяются к специальному *групповому коммутатору*, способному играть роль адаптера канал — канал между любой парой подсоединенных к нему ЭВМ.

Заметим, что каналы даже двух совершенно одинаковых ЭВМ нельзя подсоединить друг к другу непосредственно (без адаптера). Это происходит в силу несимметричности канала: интерфейс канала с ОЗУ отличен от его интерфейса с подключаемыми к нему периферийными устройствами. Задача адаптера канал — канал как раз и состоит в том, чтобы обращение к «чужому» ОЗУ выглядело бы как обращение к «своему» периферийному устройству.

Имея в своем распоряжении достаточное число каналов и даже простые (парные) адаптеры канал — канал (а тем более — групповые коммутаторы), можно в принципе организовать сколь угодно сложные *структуры связей* между отдельными ЭВМ комплекса при любом их числе.

Простейшая из таких структур — так называемая *кольцевая структура* — изображена на рис. 7.1. Для ее организации у каждой из объединяемых ЭВМ под межмашинные связи выделяется по два канала. Кружками на этой структуре обозначены адаптеры канал — канал. Их число в этом случае равно числу объединяемых ЭВМ.

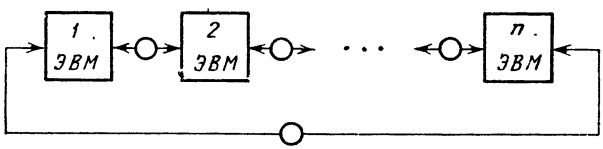

Рис. 7.1

По организации управления вычислительным процессом многомашинные комплексы делятся на комплексы с *централизованным* и *децентрализованным управлением*. В первом случае в комплексе выделяется специальная *машина-диспетчер*, управляющая работой всех других машин. Во втором случае все машины комплекса равноправны и могут организовывать свою совместную работу под

управлением любой из них. Централизованное управление несколько легче в реализации, но обладает меньшей надежностью, поскольку выход из строя машины-диспетчера нарушает работу всего комплекса. При децентрализованном управлении, обеспечивающем высокую надежность и живучесть комплекса, помимо большей сложности реализации надо считаться еще с дополнительными потерями времени и ресурсов (прежде всего памяти) на взаимные согласования и многочисленные передачи управления.

Возможно и комбинирование двух указанных управлений за счет введения нескольких управляющих машин, обязанности между которыми перераспределяются (в результате возможных неисправностей) на основе принципа децентрализованного управления.

Сигналы управления могут передаваться в комплексах по тем же самым каналам, по которым передается информация. В случае, когда у объединяемых машин имеются шины прямого управления (см. гл. V), как это имеет место в случае ЕС ЭВМ, управляющие сигналы выгоднее подавать именно по ним. Помимо большого быстродействия, подобный способ передачи хорош также тем, что УУ центрального процессора воспринимает сигналы, пришедшие по шинам прямого управления, как особый вид прерываний. Поэтому составляя соответствующим образом программы обработки этих прерываний, разработчик комплекса может осуществить различные взаимодействия объединяемых в комплекс ЭВМ без дополнительной переделки их ОС. При децентрализованном управлении подобные программы составляют основу *распределенной операционной системы комплекса* (ОСК). При централизованном управлении программы ОСК встраиваются в основном в машину-диспетчер.

Комплексирование ЭВМ, о котором шла речь в настоящем параграфе, предполагает, что объединяемые ЭВМ находятся на небольшом расстоянии друг от друга (в одном ВЦ). Комплексы ЭВМ, находящихся на большом расстоянии друг от друга, называются *сетями* ЭВМ. Объединение ЭВМ в сети производится с помощью *систем электрической связи.*

7.2. Системы связи

Для организации связи между удаленными друг от друга ЭВМ определяющее значение имеет то обстоятельство, что принятая форма (как потенциальная, так и импульсная) представления сигналов внутри современных ЭВМ предполагает большую крутизну их фронтов.

В результате даже при невысоких темпах посылки сигналов в их естественном (машинном) виде линия связи должна иметь

весьма широкую полосу пропускания. При *прямых связях* (без специальной аппаратуры усиления и восстановления сигналов) для передачи машинных сигналов требуются дорогостоящие высокочастотные кабели. Но даже и по таким кабелям дальность надежной прямой связи между ЭВМ не превышает обычно нескольких сотен метров.

Применяя на приемном конце специальную аппаратуру распознавания, усиления и восстановления формы сигналов, удается увеличить длину прямых связей до нескольких километров. Например, в ЭВМ СМ1 и СМ2 с помощью специальных расширителей интерфейса можно удалять периферийные устройства на расстояние до 3 км. Разработанная в Институте кибернетики АН УССР аппаратура Барс позволяет увеличить это расстояние до 20 км и притом без специальных высокочастотных кабелей.

Для передачи машинных сигналов на большие расстояния сегодня принято использовать существующие сети телеграфной и телефонной связи. Для обеспечения передачи информации по относительно узкополосным каналам в этих сетях используется несколько иная (отличная от принятых в ЭВМ) форма представления сигналов. Поэтому при использовании этих каналов для передачи машинных сигналов необходимо на их концах (подсоединенных к ЭВМ или терминалам) ставить специальные устройства для преобразования формы сигналов — так называемые *модемы*. Для понимания возникающих здесь проблем необходимо хотя бы в общих чертах познакомиться с принципами организации современной системы местной и дальней телефонной связи.

Прежде всего оказывается, что для достаточно качественной передачи обычной человеческой речи достаточна полоса пропускания в 3 кГц (точнее, диапазон частот от 300 до 3400 Гц). Этот стандарт частоты получил наименование *тональной частоты*. Заметим, что он выбран на основании частотных характеристик оконечных устройств (органов речи и слуха человека), а не каких-либо характеристик самой сети связи.

Сигналы, которые передаются в местных сетях при телефонных разговорах, представляют собой простые электрические копии соответствующих звуковых сигналов. Телефонные аппараты подсоединяются к местным телефонным станциям с помощью так называемых *абонентских линий*, представляемых физически в виде жил многожильного кабеля. Телефонная станция осуществляет коммутацию абонентских линий, позволяя подсоединить каждого из подключенных к ней абонентов к любому другому. Кроме того, определенная часть коммутируемых кабельных линий предназначается для связи данной телефонной станции с другими телефонными станциями, составляющими местную телефонную сеть.

Для дальней (междугородной) телефонной связи подобная организация использования кабелей, при которой одна физическая

линия (пара жил кабеля) используется для передачи одного телефонного разговора, была бы слишком неэкономной. Поэтому для такой связи каждая физическая линия используется для одновременной передачи многих телефонных разговоров. С этой целью с помощью специальной *каналообразующей аппаратуры* тональной частотой модулируются высокочастотные сигналы, разнесенные по спектру частот так, чтобы возникающие в результате полосы частот не перекрывали друг друга. Если все эти сигналы передаются по одной физической линии одновременно, то с помощью специальных фильтров на приемном конце их можно разделить и демодулировать в исходную тональную частоту. Для надежного разделения несущие частоты должны быть разнесены по спектру с промежутками, несколько превышающими тональную частоту: в качестве такого промежутка у нас выбрана частота в 4 кГц.

В зависимости от полосы пропускания той или иной физической линии связи употребляются различные степени уплотнения исходного канала (линии связи), т. е. количество виртуальных телефонных каналов, каждый из которых может быть использован (одновременно с другими виртуальными каналами) для передачи телефонного разговора по данной физической линии.

Принятый у нас стандарт предусматривает несколько групп уплотнения. Первичная 12-канальная группа охватывает полосу частот от 60 до 108 кГц. Пять первичных групп составляют вторичную 60-канальную группу с полосой частот от 312 до 552 кГц. Пять таких групп объединяются в 300-канальную группу с полосой частот от 812 до 2044 кГц и т. д.

Поскольку для телеграфной связи требуется меньшая полоса пропускания (в принятом у нас стандарте 140 Гц), то для улучшения использования линии при передаче телеграфных сообщений используется аппаратура *вторичного уплотнения*, образующая в одном телефонном канале 12 телеграфных каналов.

В качестве физических линий при дальней связи кроме устаревшей проводной связи используются высокочастотные кабельные линии и радиосвязь в УКВ, дециметровом и сантиметровом диапазонах*). Поскольку в этих диапазонах устойчивая связь обеспечивается лишь в пределах прямой видимости (без огибания радиоволнами поверхности Земли), то линия связи представляет собой ряд вышек с прямопередающими в виде направленных зеркал антеннами, каждая из которых снабжается аппаратурой для необходимого усиления передаваемых сигналов (так называемые *радиорелейные* линии связи).

*) Приведенные выше частоты относятся к кабельным линиям. В случае радиосвязи несущие частоты значительно выше.

Такими же усилительными станциями (расположенными на определенном расстоянии друг от друга) снабжаются проводные и кабельные линии дальней связи.

В последние годы все большее распространение получают космические (спутниковые) линии связи, использующие в качестве ретрансляционной станции высоколетящие спутники. Особенно удобны так называемые *стационарные спутники*, период обращения которых вокруг Земли равен периоду обращения Земли вокруг своей оси, а плоскость орбиты совпадает с плоскостью экватора. Такой спутник будет все время висеть (на высоте порядка 40 000 км) над одной и той же точкой земного экватора, не меняя своего положения относительно поверхности Земли.

Спутниковая связь обладает двумя большими преимуществами: уменьшает количество промежуточных ретрансляционных станций (которые устанавливаются на спутниках) до одной или в крайнем случае до двух; в спутниковой связи передаваемый сигнал только малую часть расстояния проходит в непосредственной близости к поверхности Земли, в зоне, насыщенной радиопомехами от гроз и разнообразных технических источников (например, систем зажигания автомобильных двигателей). Благодаря этому надежность спутниковой связи оказывается более высокой по сравнению с другими системами связи.

Как уже отмечалось выше, стандарты, принятые для телефонных каналов, определяются частотными характеристиками оконечных устройств (органов речи и слуха человека). То же самое имеет место и в случае телеграфной связи. Казалось бы, естественно, чтобы при организации связи между ЭВМ использовался тот же самый принцип. Тогда задача системы связи сводилась бы к созданию временных каналов соответствующей пропускной способности так, чтобы соединяемые ЭВМ могли обмениваться информацией в естественном для них темпе. Создание сети ЭВМ, удаленных друг от друга на любое расстояние, в принципе, не отличалось бы от создания комплекса ЭВМ, расположенных в пределах прямых связей. Отличиями были бы только необходимость модулирования и демодулирования несущих частот системы дальней связи машинными сигналами и дополнительные требования по борьбе с возникающими в линиях связи помехами.

В действительности развитие сетей ЭВМ пошло по линии вписывания системы передачи данных между машинами в уже существующие стандарты сетей телефонной и телеграфной связи. Поэтому возникает необходимость создания многоуровневой системы преобразования потоков данных, которыми обмениваются ЭВМ через сеть связи, и соответствующих систем стандартов представления информации и управления для каждого уровня — так называемых *протоколов*.

Задачей самого нижнего, *физического уровня* является преобразование стандартных однобитовых сигналов (каждый из которых принимает лишь 2 значения: 0 и 1) в различимые друг от друга сигналы тональной частоты, а также обратное преобразование. Эта задача решается специальными устройствами модуляции и демодуляции — *модемами*. Простейшие модемы с *частотной модуляцией* используют для представления 0 и 1 две различные частоты в пределах полосы 300 — 3400 Гц. Другие типы модемов используют *амплитудную* или *фазовую модуляцию*. Возможны также и более сложные комбинированные системы.

Скорость передачи дискретной информации с помощью модемов разных типов определяется прежде всего используемым каналом связи. Различают *низкоскоростные* модемы, использующие телеграфные каналы; *среднескоростные*, использующие стандартные телефонные каналы, и *высокоскоростные*, использующие первичные 12-канальные группы. Внутри этих групп установлены стандарты скорости передачи, зависящие от уровня помех в каналах связи и используемого метода модуляции.

Для низкоскоростной передачи приняты стандарты 50, 100 и 200 бод (бит/с), для среднескоростной — 600, 1200, 1800, 2400, 3000, 3600 и 4800 бод, для высокоскоростной — 19,6, 48 Кбод и выше. Для каналов с малым уровнем помех в принципе возможно дополнительное увеличение скорости передачи данных, приходящейся на 1 Гц полосы канала. Так, для низкошумящего стандартного телефонного канала (с полосой 3,1 кГц) удается передавать данные со скоростью 9600 бит/с.

7.3. Сети ЭВМ

Для понимания специфики преобразований информации в сетях ЭВМ необходимо хотя бы в общих чертах познакомиться с принципами организации (или, как принято говорить, *архитектурой*) подобных сетей. Пример сети достаточно простой архитектуры изображен на рис. 7.2.

В узлах этой сети, изображенных квадратами, расположены так называемые *коммутационные процессоры* (КП), соединяемые между собой (через модемы) линиями дальней связи. Для подсоединения терминалов обычно используются коммутационные процессоры специального вида, называемые *терминальными* (ТКП). Кружками изображены ЭВМ, а треугольниками — терминалы пользователей. Они подсоединяются либо прямо к ЭВМ, либо к территориально близким к ним узлам сети (через модемы или, в пределах прямых связей, без них) с помощью местных абонентских линий или (с модемами) через местную телефонную сеть (включая *автоматическую* междугородную связь). В местах, где

сосредоточено много терминалов, могут устанавливаться местные *концентраторы* и *мультиплексоры передачи данных*. Они позволяют использовать одну линию связи для обслуживания нескольких низкоскоростных терминалов одновременно.

Коммутационные процессоры могут осуществлять функцию *коммутации каналов* между *абонентами сети* (ЭВМ и терминалами), организуя на все время работы друг с другом двух абонентов постоянный канал связи между ними. Однако подобный способ связи приводит обычно к весьма неэкономному использованию каналов связи. Дело в том, что во время сеанса связи между отдельными сообщениями, которыми обмениваются абоненты, могут быть промежутки ожидания, когда канал, будучи занятым, не используется для фактической передачи информации. Даже при непрерывной передаче информации канал может оказаться недогруженным ввиду малой скорости передачи (что, как правило, имеет место при передаче данных с терминалов).

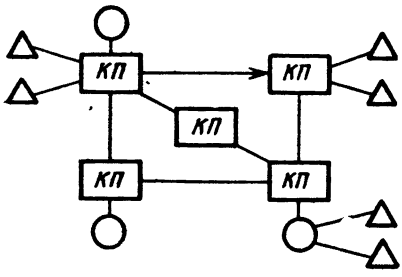

Рис. 7.2

Для более плотной загрузки каналов употребляется другой способ коммутации — *коммутация сообщений*. Суть его состоит в том, что сообщения, которыми обмениваются абоненты сети, могут временно запоминаться в узлах сети (в памяти коммутационных процессоров) и передаваться от узла к узлу наиболее экономным способом (как правило, на большой скорости), обеспечивая гораздо лучшее (по сравнению со способом коммутации каналов) использование дорогостоящих каналов дальней связи.

Однако коммутация произвольных сообщений имеет и свою отрицательную сторону: при варьировании длины сообщений в широких пределах память КП приходится рассчитывать на самые длинные из них. В результате происходит неоправданное увеличение стоимости памяти КП при малом среднем коэффициенте ее загрузки.

Для устранения этого недостатка в современных сетях употребляется разбивка сообщений на *пакеты* стандартной длины, передаваемые независимо друг от друга и зачастую даже по различным путям. Тем самым в задачу управления сетью добавляется управление разборкой на пакеты и сборкой из пакетов сообщений, оптимизация маршрутов передачи пакетов в зависимости от загрузки каналов и узлов сети и многое другое. Коммутаторы сообщений и пакетов выступают в качестве концентраторов информации и потому в некоторых сетях носят наименование *концентра-*

торов. Задержки, вызываемые ими, как правило, весьма малы (не превышают долей секунды).

7.3.1. Сетевые протоколы. Опишем теперь основные задачи управления и стандарты представления информации (*протоколы*) на различных уровнях, начиная с нижнего — так называемого *линейного протокола* или *протокола управления информационным каналом* (сокращенно ИКУП). Этот протокол может реализоваться частично в КП, но большей частью в специальных устройствах, называемых *адаптерами линии передачи данных* (ЛПД).

Главная задача ИКУП — обеспечение достоверности передачи информации даже при недостаточно надежном канале связи. Эта задача может быть решена многими способами. Один из них — использование кодов с обнаружением и исправлением ошибок, описанных в предыдущей главе. Напомним, что простейший из таких кодов — код с контролем по четности, который используется и в ЭВМ. Смысл его состоит в том, что к каждой посылке из заданного числа k бит (в ЭВМ обычно $k=8$) добавляется еще один двоичный сигнал (0 или 1) так, чтобы общее число единиц в полученной $(k+1)$-битовой посылке было четным. Задачами ИКУП в этом случае будут контроль по четности на приемном конце и посылка ответного сигнала на предыдущий конец о результате этой проверки. Если проверка подтвердила правильность приема, то передается следующая посылка, если нет — то осуществляется повторная передача неправильно принятой посылки. Именно так осуществляется контроль правильности передачи информации между центральной и периферийной частями ЭВМ. В сетях связи, однако, такой простой способ не годится. Причина заключается в том, что для сетей связи наиболее характерны так называемые *групповые помехи*, которые искажают несколько бит посылки подряд.

Для обнаружения, а тем более для исправления ошибок в случае групповых помех обычные способы потребовали бы неоправданно большого числа контрольных разрядов. Поэтому способ контроля видоизменяется в соответствии с идеей, показанной на рис. 7.3. Здесь на рисунке К разр.— контрольные разряды, позволяющие обнаружить и исправлять ошибки в каждой строке. Однако передача происходит не по строкам, а по столбцам.

Если длина групповой помехи не превосходит одного столбца, а вероятность появления помехи достаточно мала, то даже при наличии такой помехи, как правило, будет искажаться не более одного разряда в каждой строке, что позволяет на приемном конце не только обнаруживать, но и исправлять ошибки при относительно небольшом числе контрольных разрядов.

Необходимость изменения порядка следования сигналов в посылках до и после передачи сильно усложняет ИКУП. Поэтому на практике обычно употребляют более простой способ контроля

правильности передачи посылки: передав посылку от пункта A в пункт B, немедленно передают ее обратно в пункт A, где сравнивают с запомненной копией переданной посылки. В случае совпадения в B передается сигнал правильности передачи и начинается передача следующей посылки. В противном случае процесс передачи посылки повторяется столько раз, сколько потребуется для правильной передачи. Оптимальная длина посылки

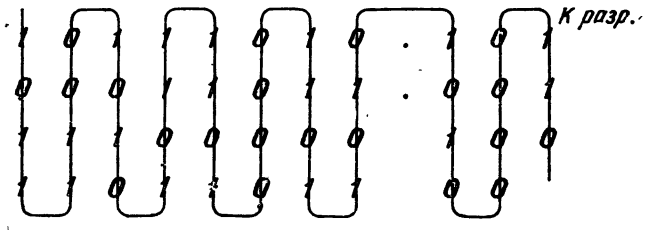

Рис. 7.3

(обеспечивающая максимум пропускной способности канала) зависит от вероятности искажения передаваемых сигналов. Если эта вероятность равна 10^{-2} (весьма ненадежный канал), то оптимальная длина посылки равна 5 бит. При этом за счет необходимости многократных повторений передачи посылок истинная пропускная способность канала составит всего 1,4% номинальной. При вероятности ошибок 10^{-4} (канал средней надежности) оптимальная длина посылки равна 57 бит, а коэффициент использования канала поднимается примерно до 50%. Для каналов высокой надежности (вероятность ошибки 10^{-6}) соответствующие величины будут равны 610 бит и 75%. Важно подчеркнуть, что правильно построенный ИКУП (и соответствующая аппаратура передачи данных) позволяет осуществлять надежную передачу данных (хотя и с пониженной скоростью) по сколь угодно плохим каналам.

Еще один вопрос, связанный с ИКУП: как отличать одну посылку от другой? Один из широко употребляющихся способов решения этой проблемы заключается в использовании специальных маркеров в начале и в конце каждой посылки. Например, маркером может служить комбинация 01111110, т. е. группа из 6 последовательных единиц, обрамленных нулями. Чтобы исключить возможность случайного появления такой комбинации в теле посылки, аппаратура передачи данных производит автоматическую вставку (на передающем конце) и исключение (на приемном конце) символа 0 в теле посылки после каждой очередной группы из 5 символов. Тем самым исключается возможность появления внутри посылки группы из 6 единиц подряд.

Следующий за ИКУП уровень управления передачей информации в сетях ЭВМ регламентируется *протоколом пакетной комму-*

тации и соответствующим математическим обеспечением, встроенным в КП сети. Протокол во-первых определяет длину пакета (обычно от 256 до 4096 бит), а также содержание и формат *заголовка пакета*, в который обязательно включаются адреса абонентов, обменивающихся пакетом. Анализируя эти заголовки, КП определяют, какому из узлов сети (соседнему с данным) переслать пакет.

При этом часть *операционной системы сети* (ОСС), встроенная в КП, решает задачу оптимизации движения пакетов от узла к узлу так, чтобы обеспечить наилучшую загрузку линий связи и самих КП, с одной стороны, и наиболее быструю передачу пакетов конечным адресатам — с другой. Для точного решения этой задачи необходим специальный управляющий процессор, непрерывно получающий информацию о состоянии всех узлов сети и всех линий связи.

Такое централизованное управление загружает сеть передачей большого количества служебной информации и предъявляет повышенные требования к надежности управляющего процессора. Поэтому на практике используют обычно упрощенные приемы приближенной локальной оптимизации, когда решение о направлении пересылки вырабатывается КП, в котором этот пакет находится, на основании информации о непосредственно прилегающих к нему участках сети. Возможен также вариант распределенного управления, при котором каждый КП действует на основе информации о состоянии всей сети, оптимизируя потоки передаваемой им информации. В последние годы этот вариант (в чистом виде или в комбинации с централизованным управлением) получает все большее распространение.

Еще одна задача, решаемая на этом уровне,— задача идентификации и аннулирования дублей пакетов, которые могут возникнуть в результате множественности маршрутов и возможности зацикливания. С этой целью в заголовке пакета должен содержаться идентификатор породившего его сообщения. Обычно он составляется из сетевого номера пославшей сообщение ЭВМ, номера процесса в этой ЭВМ (а иногда и кода пользователя), инициировавшего данное сообщение, и, наконец, идентификатора самого сообщения в этом процессе.

Следующий уровень управления имеет дело с задачей сборки и разборки сообщений на отдельные пакеты с формированием и последующим уничтожением соответствующих заголовков пакетов. Этот уровень управления регламентируется так называемым *транспортным протоколом* и реализуется частью ОСС, называемой обычно *процессом абонентских сообщений* (ПАС) или транспортной станцией. В первых сетях ПАС встраивался обычно в КП. В более поздних сетях (например, во французской Cyclades) ПАС реализуется в абонентских ЭВМ (сокращенно АВМ) и управля-

ется их операционными системами как одна из задач пользователя. Такое решение обеспечивает бóльшую стандартность сети и ее независимость от типа подключаемых к ней АВМ. При подсоединении АВМ к КП стандартная часть интерфейса реализуется в КП, а нестандартная — в самой АВМ и специальных сетевых адаптерах.

Еще более высокий уровень управления (реализующийся в АВМ) организует *потоки сообщений* от процесса к процессу, включая задачу формирования заголовков сообщений на передающем конце и их упорядочение в правильный поток (упорядоченный по номерам сообщений). На этом уровне реализуются макрокоманды создания и уничтожения так называемых *виртуальных каналов*, связывающих между собой удаленные (реализованные в удаленных друг от друга АВМ) процессы. Соответствующая часть ОСС (встраиваемая в АВМ) носит название *супервизора виртуальных каналов*.

Дальнейшая детализация управления решается на уровне протокола процесс — процесс и соответствующей части ОСС (встроенной в АВМ) — так называемым *сетевым супервизором процессов*. На этом уровне организуется связь *портов*, в качестве которых могут выступать некоторый массив данных (файл), устройство ввода-вывода, некоторая точка программы, генерирующая или потребляющая сообщения. Это примеры *индивидуальных портов* (принадлежащих определенному процессу). Примером *коллективных портов* могут служить входы в системы управления базами данных (СУБД) с последующим уточнением адресата (файла или записи) при помощи средств самих СУБД.

Каждый порт снабжается *сетевым идентификатором*, состоящим из сетевого идентификатора АВМ, идентификатора пользователя этой АВМ, идентификатора, присвоенного процессу, и, наконец, идентификатора порта в процессе. Порт снабжается необходимой буферной памятью для размещения формируемых и принимаемых сообщений. На уровне протокола процесс — процесс реализуются макрокоманды (связанные с ОС АВМ): открыть порт, закрыть порт, соединить (разъединить) порт с портом (создать или аннулировать соответствующий виртуальный канал).

Еще более высоким уровнем ОСС и соответствующих протоколов является *уровень управления* решением задач на сети. Основная решаемая здесь задача — создание *виртуальных сетей процессов* (в различных АВМ). Обычно этот уровень реализуется в виде совокупности *мониторов сетевых абонентских служб*, каждый из которых предназначен для исполнения сетью той или иной определенной функции (например, справочно-информационной). Такие мониторы могут встраиваться в АВМ, либо в специальные *информационно-диспетчерские* ЭВМ (ИДВМ). В таких ИДВМ может, например, реализоваться функция безадресного обращения

любого пользователя сети к распределенному (между абонентскими ЭВМ) банку данных через соответствующую систему сетевых каталогов. В задачу мониторов абонентских служб входит планирование и управление процессом решения задач на сети с учетом загрузки отдельных АВМ и линий связи. Все эти функции регламентируются специальными протоколами.

Помимо уже перечисленных протоколов в ряде действующих сетей используются и другие протоколы, как, например, специальные протоколы для пересылки файлов, для удаленного ввода заданий, протокол виртуального терминала и др. Отметим также, что ОСС ведет постоянный учет объема услуг, предоставляемых ею каждому пользователю, и начисляет соответственно суммы к оплате.

В связи с развитием многих сетей национального и даже межнационального масштаба с различными системами протоколов возникает специальная задача организации *межсетевого взаимодействия.* Для решения этой задачи обычно устанавливается общий (межсетевой) протокол процесс — процесс, фиксируются единый формат сообщений и единый язык управления обменом сообщениями. Для преобразования подобных межсетевых сообщений в форматы, принятые в отдельных сетях, некоторые узлы (КП) объединяемых сетей соединяются друг с другом линиями связи через специальные устройства (обычно универсальные ЭВМ с адаптерами), называемые *шлюзами.*

7.4. Вычислительные центры коллективного пользования

Простейшим примером сети ЭВМ служит так называемая *терминальная сеть.* Она состоит из единственного вычислительного центра (ВЦ) и сети удаленных терминалов. Вычислительный центр в этом случае принято называть ВЦ *коллективного пользования* (ВЦКП). Он оснащается, как правило, достаточно мощной ЭВМ или, чаще всего, многомашинным вычислительным комплексом. Удаленные терминалы, как правило, объединяются в группы, каждая из которых замыкается на *терминальный концентратор.* В простейшем случае он представляет собой обычный мультиплексор передачи данных. В более сложных случаях в качестве концентратора может использоваться мини-ЭВМ, способная не только накапливать сообщения от терминалов для лучшей загрузки каналов связи, но и выполнять более сложные функции, связанные с обработкой информации. Сложность самих терминалов также может варьироваться в значительных пределах. Наряду с простейшими терминалами в виде телетайпов или алфавитно-цифровых дисплеев, в терминальную сеть могут включаться более сложные терминалы — АРМ, т. е. автоматизированные рабочие места (см. § 4.10) и абонентские пункты (см. § 4.9). Как уже

отмечалось в гл. IV, под такими названиями могут фигурировать целые комплексы вводно-выводных устройств, включая устройства внешней памяти на магнитных лентах и магнитных дисках. Для подключения подобных комплексов с высокопроизводительными устройствами требуются высокоскоростные каналы и широкополосные модемы.

Заметим, что термин «ВЦ коллективного пользования» не очень удачен в том смысле, что подчеркивает лишь одну, так сказать, юридическую сторону этого понятия, а не технологию предоставления услуг (удаленный доступ). А суть сегодня состоит именно в этой технологии, поскольку коллективное пользование ЭВМ и ВЦ в чисто юридическом плане практиковалось с самых первых шагов развития вычислительной техники. Современные же ВЦКП — это центры с удаленным доступом, работающие либо в режиме разделения времени, либо в режиме удаленного управления заданиями. Самое важное при этом, что ВЦ предоставляет свои услуги пользователям, находящимся на своих рабочих местах, а не приходящим за их получением в ВЦ. Именно такое понимание термина ВЦКП укоренилось в сегодняшней нашей практике, и именно так мы его будем понимать в дальнейшем.

7.5. Распределенные банки данных

Первоначально сети ЭВМ предоставляли своим пользователям только вычислительные услуги (точнее, услуги по обработке информации). Базы данных (в случае необходимости) создавались самими пользователями для своего индивидуального потребления. Однако существует огромное число данных, которые интересуют большое число самых различных пользователей. Это, прежде всего, научно-техническая информация, различного рода справочные данные и др. Поэтому многие сети ЭВМ стали предоставлять своим абонентам (пользователям) доступ не только к вычислительным мощностям, но и к накапливаемой в сетях (в безбумажной форме) разнообразной информации.

Во многих случаях эта информация не концентрируется в какой-либо одной ЭВМ, а распределена по всем ЭВМ сети. Доступ в подобные *распределенные базы* (*банки*) *данных* осуществляется специальными сетевыми СУБД. Такие СУБД дают возможность *безадресного обращения* к данным, подобно тому как это делается в случае обычных баз данных, реализованных на одной ЭВМ (см. гл. VI). Зная *логическую структуру* базы данных сети, абонент формирует запрос к ней (на языке манипулирования данными), не заботясь о том, в каких именно ЭВМ сети расположены интересующие его данные.

Интерфейс с реальной *физической структурой* данных осуществляется СУБД автоматически через систему машинных ка-

талогов. При этом не исключено, что окончательный ответ на запрос абонента будет сформирован из данных, хранящихся не в одной, а в нескольких (удаленных друг от друга) ЭВМ сети. Формирование ответа предусматривает таким образом многократные обмены между различными ЭВМ и автоматическое редактирование текста ответа. Вся эта работа производится под управлением операционной системы сети.

7.6. Некоторые иностранные сети ЭВМ

7.6.1. Сеть ARPA. Одной из первых действующих сетей ЭВМ была сеть ARPA в США, созданная вначале в интересах управления перспективными исследованиями при Министерстве обороны США (Advanced Research Project Agency). Первая очередь этой сети была введена в эксплуатацию в 1969 г. В настоящее время она охватила всю территорию США, а через английский и норвежский узлы связи в сеть включены многие ВЦ и пользователи в Европе. В сети ARPA помимо обычных коммутационных процессоров (КП), обеспечивающих соединение между собой ВЦ сети, используются также так называемые *терминальные коммутационные процессоры* (ТКП). Через каждый из них к сети могут подключаться до 63 терминалов и несколько ЭВМ.

В сети используются машины многих типов почти всех крупных американских фирм, в том числе специализированная мультипроцессорная система Illiac IV, выполняющая на векторно-матричных операциях порядка 200 млн. оп/с. Сообщения длиной до 8064 бит передаются из ЭВМ в КП, где разбиваются на пакеты длиной 1008 бит. В операционной системе сети имеются средства автоматического переформатирования не только элементарных данных, но и файлов при обменах между ЭВМ с разными форматами данных. Встроенный в терминальный КП *протокол виртуального терминала* преобразует сообщения из кодов терминалов в единый код сети ARPA, осуществляет связь терминалов с удаленными ЭВМ и управляет работой терминалов. Операционная система сети имеет средства для автоматического распределения ресурсов сети, в частности — в закреплении терминалов за отдельными ЭВМ. Различные ЭВМ сети специализируются при этом на определенных видах обслуживания, что повышает эффективность их использования.

Во второй половине 70-х годов в сеть включаются модульные многопроцессорные КП, допускающие подключение 20 ЭВМ и несколько сотен терминалов. Они способны использовать наземные и спутниковые линии связи до 1,3 Мбит/с.

7.6.2. Сеть Cybernet. Сеть принадлежит американской фирме СДС (Контрол Дейта). Она, как и сеть ARPA, начала функционировать в 1969 г. Ее пользователями являются 350 крупнейших

корпораций США и многие крупные научно-исследовательские центры. Особенностью сети является широкое использование коммутации каналов вместо коммутации сообщений. Это удешевляет сеть и при наличии широкополосных коммутаторов обеспечивает абонентов приемлемым временем коммутации. В сети используются сообщения длиной в 8192 бит.

7.6.3. Сеть фирмы General Electric. Сеть начала работать в 1969 г., а через 10 лет выросла в одну из крупнейших мировых сетей. Ее главной опорой является мощный ВЦ в Кливленде (США). Свыше 50000 пользователей сети в 250 городах США, Европы и Японии поключаются к ней с помощью местной телефонной сети через удаленные концентраторы. Удаленные концентраторы построены на базе мини ЭВМ. Каждый из них подключен к 48 телефонным каналам местной телефонной сети. Использование дополнительно мультиплексоров с разделением по частоте позволяет подсоединять к каждому концентратору одновременно до ста медленных терминалов (типа телетайпа), либо до 10 видеотерминалов.

Кроме того, имеются еще 16 центральных концентраторов, три из которых расположены в Европе. Они соединены с опорным ВЦ в Кливленде через трансатлантические кабели и через стационарный спутник связи. Все компоненты сети дублированы, что обеспечивает высокую надежность ее работы. В центральных концентраторах хранятся таблицы, указывающие, в каких ЭВМ сети пользователи хранят свои программы и данные. Эти таблицы используются для определения маршрутов передачи сообщений в начале каждого сеанса связи. Периодически (не реже, чем раз в месяц) вычислительные мощности перераспределяются между абонентами сети с тем, чтобы обеспечить равномерную загрузку. Поскольку зона действия сети охватывает много часовых поясов, она имеет выровненный суточный график нагрузки.

Заметим, что развитая сеть автоматической междугородной телефонной связи существенно видоизменяет понятие местной телефонной сети. Поэтому удаленные концентраторы сети не обязательно ставятся в каждом городе, в котором есть абоненты сети. Абоненты могут использовать концентраторы в любых близлежащих городах с достаточно дешевыми тарифами междугородной связи. Для обеспечения равномерной нагрузки сетей связи в системе могут осуществляться периодические перезакрепления абонентов за различными концентраторами. При использовании местной (не сетевой) междугородной связи абонент оплачивает ее услуги помимо оплаты услуг собственно сети (за передачу информации от удаленного концентратора в ВЦ и обратно, а также за обработку и хранение информации).

Как и во всякой коммерческой сети, в сети General Electric величина оплаты услуг сети насчитывается каждому абоненту

операционной системой сети автоматически. Сеть предоставляет широкий круг услуг как в режиме разделения времени (системы MARK I, II и III), так и в удаленной пакетной обработке. Кроме того, в последние годы непрерывно растет возможность предоставления сетью различного рода информационных услуг, т. е. доступа к базам данных коллективного пользования.

7.6.4. Сеть Cyclades. Национальная французская сеть Cyclades (Циклад) обслуживает широкий круг пользователей, в числе которых важное место занимают правительственные органы (как центральные, так и местные). Первая очередь сети в составе 16 ЭВМ и 5 узлов коммутации вступила в строй в 1974 г. Сеть строилась на основе усовершенствования технических решений, принятых в ARPA. Усовершенствования направлены на обеспечение большей гибкости и простоты подсоединения к сети ЭВМ и терминалов.

С этой целью в каждой ЭВМ функции *транспортной станции* (ТС) осуществляются программой, работающей как программа пользователя. Благодаря этому переделки в операционных системах, подключаемых к сети ЭВМ, сводятся к минимуму. Транспортные станции реализуют прежде всего протокол связи ЭВМ с *сетью пакетной коммутации* (СПК), т. е. транспортный протокол. Кроме того, на них возлагается задача реализации части протоколов более высоких уровней, обеспечивающих связь между различными ЭВМ.

В любой ЭВМ может располагаться несколько транспортных станций, каждая из которых управляет одной или несколькими линиями связи, соединяющими эту ЭВМ с узлами сети пакетной коммутации. Встраивая в ЭВМ специальную *транзитную* транспортную станцию, можно подсоединить ее к узлам различных сетей и использовать ее в качестве шлюза (см. п. 7.3.1), соединяющего эти сети. Таким образом, в сети Cyclades облегчена и задача объединения с другими сетями, причем объединение с сетью ARPA и английской национальной сетью NPL уже осуществлено.

Еще одно удобство сети Cyclades — гибкая система адресации, позволяющая практически неограниченно наращивать сеть. Вся сеть разбита на районы. Полный адрес любой транспортной станции (ТС) состоит из номера района и ее *локального номера* внутри этого района. Каждый район обслуживается своим узлом коммутации, причем коммутационный процессор узла хранит сведения лишь о ТС своего района. При выборе же им линии связи при отправке пакета в другой район используется только номер этого района.

Абонент сети идентифицируется (полным) адресом своей ТС и *локальным номером* внутри ТС. В одной ТС может размещаться до 2048 абонентов с 16 портами каждый, либо до 128 абонентов с 256 портами каждый. Порты связываются между собой *дуплекс-*

ными виртуальными каналами, т. е. такими каналами, которые допускают одновременную передачу сообщений в обоих направлениях. Размер пакета — до 2040 бит (такой же, как и в сети NPL), причем 72 бита из них отводятся на заголовок, в котором указываются номера адресата и отправителя, тип сети-адресата (Cyclades, ARPA или NPL), приоритет сообщения, к которому принадлежит пакет, и поля для специальных отметок служебных пакетов.

Сообщения в системе Cyclades называются *письмами*. Так называемое *регулярное письмо* может послать (по любому адресу) каждый абонент, причем для этого не надо использовать никаких дополнительных команд. Такие письма удобны для ведения диалога и координации процессов в различных ЭВМ сети. При намерении вести длительный обмен используется другой тип писем — тип *соединения*. Для них в процессе установления *устойчивого* виртуального канала фиксируется максимальный размер письма, которое может быть принято каждым партнером. После получения очередного письма данного типа адресат сообщает отправителю о его доставке и о том, какое число писем он может еще принять. В письмах при этом вместо полного адреса абонента может использоваться более короткий номер установленного виртуального канала. Помимо писем в сети могут циркулировать короткие (16-битовые) команды, называемые *телеграммами*. Они служат для обмена информацией между узлами сети, а также между узлами и ТС, с целью организации управления потоками пакетов.

Среднее время задержки пакета в сети Cyclades (как и в сети ARPA) равно 0,1 с. Вероятность ошибки при передаче пакетов — не более 10^{-10} на бит (10^{-4} на пакет). Подключение терминалов к сети производится через терминальные коммутационные процессоры. Однако использование терминалов со своими буферными устройствами позволяет в принципе их прямое подключение к узлам коммутации. Скорость передачи информации между узлами коммутации в сети Cyclades колеблется в пределах от 4,8 до 48 Кбит/с.

7.6.5. Другие иностранные сети. Помимо уже перечисленных сетей в мире сегодня действуют несколько сотен сетей ЭВМ различного назначения. Кроме универсальных сетей, способных предоставлять произвольные вычислительные и информационные услуги, существуют специализированные сети. К их числу относятся, например, банковские сети, позволяющие осуществлять безбумажные расчеты между клиентами территориально удаленных друг от друга банков. Другой пример — сети, связывающие ЭВМ и терминалы в системах продажи авиационных билетов.

К началу 80-х годов все технически развитые страны располагали сетями с развитыми системами терминального обслуживания. Многие страны создали или создают сети национального

масштаба для обслуживания правительственных органов. К их числу относятся не только такие страны, как США, Япония, Франция, ФРГ, Англия, но и многие малые государства, в том числе из числа развивающихся стран.

7.7. Государственная сеть вычислительных центров

Особо большую роль сети ЭВМ могут играть в управлении экономикой в национальных масштабах (что в полной мере возможно лишь в социалистических странах). Впервые вопрос о создании такой сети в Советском Союзе возник в 1962 г. На протяжении 1963 г. под руководством автора был создан эскизный проект *Государственной сети вычислительных центров* (ГСВЦ) для управления экономикой на всех уровнях (от цеха до Госплана СССР). По ряду причин практическая реализация этого проекта началась лишь в 70-е годы после решений 24-го съезда КПСС. Следует сразу подчеркнуть, что по своим масштабам ГСВЦ (в полном объеме) во много раз превосходит любую из известных в настоящее время сетей ЭВМ. Поэтому полная реализация проекта требует значительного времени.

Основу ГСВЦ должна составить *опорная сеть* особо мощных общегосударственных ВЦ коллективного пользования (ВЦКП), дислоцированных в крупных городах страны (практически во всех областных центрах). В состав каждого такого опорного ВЦКП входит прежде всего многомашинный вычислительный комплекс той или иной (в зависимости от класса ВЦ) мощности. Вторая составная часть оборудования — *региональный узел коммутации* (РУК). Третья компонента — *информационно-диспетчерский пункт* (ИДП).

Вся территория страны разбивается на регионы, на каждый из которых приходится по одному такому *опорному* ВЦКП. Все ВЦ и терминалы, обслуживающие предприятия и органы управления экономикой, находящиеся в данном регионе, подсоединяются к соответствующему региональному узлу коммутации с помощью относительно узкополосных (коммутируемых или закрепленных) каналов местной системы связи.

Опорные центры соединяются друг с другом с помощью широкополосных каналов (не менее 48—96 Кбит/с). В идеале пропускная способность этих каналов должна быть доведена до уровня пропускных способностей периферийных процессоров объединяемых ЭВМ. Для этой цели соответствующие широкополосные модемы должны быть включены в обход каналообразующей аппаратуры непосредственно в линии дальней связи с необходимой пропускной способностью. С целью уменьшения затрат могут быть использованы существующие линии передачи телевизионных программ. Поскольку в дневное и вечернее время такие линии заня-

ты своим прямым делом (они используются также для междугородных телефонных переговоров), опорные центры могли бы использовать их для своих нужд прежде всего в ночные и в ранние утренние часы.

Следует подчеркнуть, что переход на скорости обмена, соответствующие характеристикам периферийных процессоров объединяемых ЭВМ, фактически (исключая необходимость использования модемов) превращает сеть в многомашинный комплекс, объединяющий удаленные ЭВМ через их собственные канальные процессоры и адаптеры канал — канал. При этом устраняется необходимость многих сетевых протоколов, единственной причиной необходимости которых является несоответствие пропускных способностей источников и приемников информации с пропускной способностью соединяющих их каналов. К числу таких протоколов относится прежде всего транспортный протокол.

Заметим, что необходимость сложной системы протоколов в современных сетях ЭВМ обусловлена в первую очередь именно подобным несоответствием. Ведь существующие сети связи развивались применительно к двум основным источникам и приемникам: к человеку (точнее, к его голосу и слуху) и к буквопечатающим телеграфным аппаратам. Именно скоростными характеристиками этих оконечных устройств обусловлен выбор пропускных способностей двух основных стандартных каналов — телефонного (3,1 кГц) и телеграфного (140 Гц). Втискивание в «прокрустово ложе» этих стандартов оконечных устройств принципиально другого класса (ЭВМ) как раз и потребовало сложной системы протоколов. С подобным положением мы сталкиваемся, например, при попытке «втиснуть» телефонный разговор в рамки телеграфного канала. Подобные системы (получившие наименование *вокодеров*) требуют специальных протоколов (реализуемых в виде особых устройств) сжатия (на передающем конце) и восстановления (на приемном конце) обычной речевой информации. При этом в процессе передачи используется дискретное представление речевой информации.

Через региональные центры коммутации и опорную сеть ВЦ и терминалы пользователей любой ведомственной принадлежности, расположенные в любых частях страны, могут обмениваться сообщениями и осуществлять совместную работу так, как это делается в уже описанных выше сетях. При этом как внутри региональных (местных) сетей, так и при передаче сообщений между опорными ВЦКП могут использоваться помимо уже упомянутых региональных узлов коммутации и другие коммутационные узлы, использующие как принцип коммутации каналов, так и принцип коммутации сообщений (пакетов). При традиционных формах организации связи в сетях первый принцип используется в основном в местных сетях, а второй — в опорной сети. Однако при использо-

вании описанной выше широкополосной связи между опорными ВЦКП принцип коммутации каналов в ряде случаев может оказаться более выгодным и для опорной сети.

В описанных выше иностранных — так называемых *коммерческих сетях* организация совместной работы па сети различных абонентов требует их предварительной договоренности о содержании и формах этой работы. Сеть же ограничивается лишь предоставлением технических услуг для ее проведения. Подобный режим работы, разумеется, возможен и в ГСВЦ. Однако, в отличие от обычных коммерческих сетей, ГСВЦ имеет еще и другую (и притом основную) цель: обеспечивать управление экономикой на всех уровнях в соответствии с принятыми на данный момент организационно-юридическими принципами. В соответствии с этими принципами по запросу тех или иных органов (наделенных требуемыми правами) может осуществляться принудительное (диктуемое сетью) объединение ВЦ и терминалов сети в те или иные *временные конфигурации* для выполнения запрашиваемой работы.

Удовлетворение подобных запросов является главной задачей информационно-диспетчерских пунктов (ИДП) опорной сети. С этой целью каждый региональный ИДП ведет оперативный учет и контроль состояния технических средств, программного обеспечения и баз данных всех абонентов сети в данном регионе, а также планов работы их ВЦ и терминалов. Если в регионе имеются органы управления надрегионального (например, общесоюзного) уровня, то соответствующая задача учета и контроля возлагается на дополнительный ИДП соответствующего уровня. Тем самым сеть ИДП оказывается построенной в соответствии с организацией и дислокацией сети социально-экономического управления в стране. Управление же этой сетью должно осуществляться специальным союзнореспубликанским органом — министерством (или комитетом) *информатики*.

Специально подчеркнем, что в задачу этого органа никоим образом не входит управление экономикой. Он осуществляет лишь управление (в интересах в с е х органов социально-экономического управления) *техническими средствами* (ВЦ, РУК, ИДП и терминалами), а также *загрузкой общегосударственной сети передачи данных* (ОГСПД), с помощью которых осуществляется социально-экономическое управление (сама ОГСПД при этом может принадлежать министерству связи).

Простейшей формой запроса, который должна удовлетворять ГСВЦ, является обычный информационный запрос (например, сколько того или иного материала находится на предприятиях данной отрасли в данном регионе). Для удовлетворения подобных запросов сеть ИДП должна обладать монитором соответствующей абонентской службы. Важнейшей составной частью такого монитора является система управления распределенной базой данных

(СУРБД), включающей каталог баз данных как всех абонентов сети, так и специальных *региональных баз данных*, создаваемых в опорных ВЦКП. В региональные базы включаются прежде всего различные данные, требуемые для территориального управления (регистры населения, земельных угодий, природных ресурсов, дорог и т. п.). Кроме того, в монитор информационной службы сети должны входить специальные программы для проверки права на запрос той или иной информации, а также *определения очередности удовлетворения запросов*. Последнее зависит не только от уровня органа, запрашивающего информацию, но и от определяемой самим этим органом срочности запроса, а также от сложности запроса и технического состояния сети (прежде всего от степени загрузки различных ВЦ и сетей связи).

Более сложными являются задачи межведомственного планирования и управления, например, задачи согласования календарных планов производства и материально-технического снабжения по цепочкам потребителей и поставщиков. Решение подобных задач (более подробно описываемых в последующих главах) требует образования временных конфигураций ВЦ и терминалов, принадлежащих различным ведомствам, и эффективного управления ими с обязательным учетом планов их работы на свои собственные нужды. Учитывая огромное быстродействие ЭВМ и высокую цену даже незначительных простоев сложного сетевого оборудования, такая задача не может быть эффективно решена иначе, как под централизованным управлением сети ИДП. Заметим, что для решения этой задачи чрезвычайно важно оперативно отслеживать состояние технической, программной и информационной базы у всех вовлекаемых в работу абонентов сети. С этой целью должен быть организован автоматический обмен информацией между операционными системами абонентских ЭВМ и мониторами соответствующих абонентских служб, реализованных в сети ИДП. Управление совместной работой абонентских ВЦ еще более усложняется в случае, когда решаемые задачи требуют оперативного диалога с людьми. Соответствующие мониторы должны в этом случае учитывать расписание работы и скоростные возможности всех специалистов, вовлекаемых в диалог.

Сложность возникающих здесь задач легко оценить, если представить себе масштабы ГСВЦ. В законченном виде она будет объединять около 200 опорных ВЦКП, ИДП и РУК, несколько десятков тысяч ведомственных ВЦ и несколько миллионов терминалов. Заметим также, что абоненты ГСВЦ неоднородны в смысле объема требований, предъявляемых к сети. ВЦ многих органов управления, прежде всего общегосударственного уровня (Госплан СССР и др.) желательно подключить к соответствующим региональным узлам коммутации широкополосными каналами, тогда как ВЦ большинства предприятий и мелких организаций могут

довольствоваться каналами меньшей пропускной способности, так что в ГСВЦ помимо опорной сети может быть выделена по крайней мере еще одна высокоскоростная подсеть, содержащая несколько сотен особо важных абонентов. Необходимо помнить также, что наряду с быстрой автоматической связью в ГСВЦ для передач больших объемов несрочной информации может использоваться пересылка по почте соответствующих машинных носителей, прежде всего магнитных лент и гибких дисков.

Как уже отмечалось выше, в опорных ВЦКП будут храниться специальные региональные базы данных. Поэтому такие ВЦ естественно использовать прежде всего для решения крупных социально-экономических задач общерегионального и межрегионального характера. Наряду с такими задачами опорные ВЦКП могут взять на себя дополнительно решение задач для абонентов, не имеющих своих ВЦ, а также выступать в виде резерва мощности при решении особо крупных задач у абонентов, не имеющих такого резерва. Это позволит рассчитывать абонентские ВЦ не на пиковые, а на средние нагрузки и тем самым значительно повысить эффективность использования вычислительной техники, снизить необходимые капитальные затраты.

Более того, наличие сети ИДП в руках органа, наделенного правом диспетчирования работы ВЦ у всех абонентов сети (независимо от их ведомственной принадлежности), позволяет эффективно управлять всем вычислительно-информационным потенциалом страны путем оперативного перераспределения нагрузок (подобно тому, как это делается в электрических сетях).

Наконец, следуя общим принципам создания вторичных баз данных, которые были изложены в предыдущей главе, в ВЦКП следует создавать такие базы для более оперативного удовлетворения поступающих в сеть запросов. Как уже отмечалось выше, накопив менее 20% общего объема данных, имеющихся во всех ВЦ данного региона, опорный ВЦКП может удовлетворять более 80% поступающих запросов, не обращаясь непосредственно к этим ВЦ. Подобная служба вторичных массивов должна обеспечиваться соответствующим сетевым монитором, в функции которого входит изучение статистики поступающих в сеть запросов, своевременная корректировка состава вторичных массивов и обеспечение их своевременной актуализации.

В заключение отметим еще одну особенность, отличающую ГСВЦ от обычных коммерческих сетей. Речь идет о способах защиты от несанкционированного доступа. Помимо обычных средств *защиты информации*, применяемых не только в отдельных ЭВМ, но и в сетях, в ГСВЦ возникает специфическая задача защиты от попыток применить те или иные программы к тем или иным данным. При этом сами по себе данные и программы открыты пользователю, запрещено лишь приводить их в соприкосновение.

Глава VIII

МАТЕМАТИЧЕСКИЕ МЕТОДЫ ОПТИМИЗАЦИИ

8.1. Общие сведения об оптимизации

Одним из важнейших применений машинной (безбумажной) информатики является машинное управление и проектирование. Применение ЭВМ в этих областях позволяет перейти от выработки более или менее хороших решений к выработке наилучших или, как обычно принято говорить, *оптимальных* решений. Процесс выработки таких решений называется *оптимизацией*. Математические методы оптимизации начали развиваться еще в домашинную эпоху. Однако без ЭВМ они могли применяться на практике лишь в самых простых случаях. Широкий переход к машинным методам выработки управленческих и проектно-конструкторских решений вызвал бурное развитие теории оптимизации и обеспечил ее широкое практическое использование.

8.1.1. Классическая постановка задачи оптимизации. Обычная постановка задачи оптимизации (которую мы будем называть *классической*) состоит в следующем. В некотором пространстве S тем или иным способом выделяется некоторое непустое множество M точек этого пространства, называемое *допустимым множеством*. Далее фиксируется некоторая вещественная функция $f(x)$, заданная во всех точках x допустимого множества. Она называется *целевой функцией*. Задача оптимизации состоит в том, чтобы найти точку x_0 в множестве M, для которой функция $f(x)$ принимает экстремальное — максимальное или минимальное значение. В первом случае для всех точек x множества M удовлетворяется неравенство $f(x_0) \geqslant f(x)$, во втором случае — неравенство $f(x_0) \leqslant f(x)$.

Заметим прежде всего, что как при *максимизации*, так и при *минимизации* задача оптимизации может не иметь решения. Например, если допустимое множество M состоит из всех положительных вещественных чисел ($x > 0$), то для целевой функции $f(x) = x^2$ в множестве M не существует ни точки максимума, ни точки минимума. Действительно, для любого $x_0 > 0$ всегда найдутся положительные числа x_1 и x_2, удовлетворяющие неравенст-

вам $x_1 < x_0$ и $x_2 > x_0$. Тогда, очевидно, $f(x_1) = x_1^2 < x_0^2 = f(x_0)$ и $f(x_2) = x_2^2 > x_0^2 = f(x_0)$.

Наоборот, может оказаться, что экстремум достигается не в одной, а в целом множестве точек. Например, в области, заданной неравенствами $-1 \leqslant x \leqslant 1$, $-1 \leqslant y \leqslant 1$, функция $f(x, y) = x^2 + y^2$ достигает максимума (равного двум) в четырех точках с координатами $(-1, -1)$, $(-1, 1)$, $(1, -1)$, $(1, 1)$. Функция $f(x)$, заданная соотношениями $f(x) = 1$ при $x \leqslant 1/2$ и $f(x) = 2x$ при $x > 1/2$, в области, определяемой неравенством $x \geqslant 0$, принимает минимальное значение (равное единице) во всех точках отрезка $[0, 1/2]$.

Следует различать *абсолютный* и *локальный* экстремумы. В случае достижения функцией $f(x)$ в точке x_0 абсолютного экстремума, скажем, максимума, неравенство $f(x_0) \geqslant f(x)$ имеет место для всех точек допустимой области M. В случае же локального экстремума (в данном случае максимума) это неравенство выполняется только для всех достаточно близких к точке x_0 точек области M. Например, $f(x_0) \geqslant f(x)$ для всех x таких, что расстояние $r(x, x_0)$ точки x от точки x_0 меньше некоторого (обычно достаточно малого) положительного числа ε.

На рис. 8.1 функция $f(x)$, заданная на отрезке $[a, b]$, имеет абсолютный максимум в точке b и два относительных максимума в точках a и d. В точке c эта функция достигает локального минимума, а в точке e — абсолютного минимума.

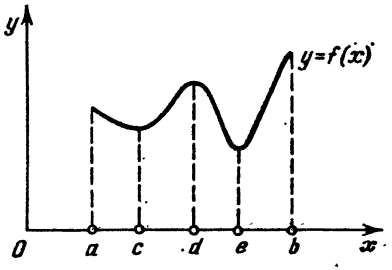

Рис. 8.1

При перемене знака целевой функции $f(x)$ все точки ее максимума превращаются, очевидно, в точки минимума и наоборот. Поэтому в теории достаточно рассматривать лишь какой-нибудь один из видов оптимума (максимум или минимум). В современной теории оптимизации чаще всего останавливаются на задаче нахождения минимума, или, что то же самое, на *задаче минимизации* функций. Все результаты этой задачи очевидным образом переносятся и на задачу нахождения максимума. Заметим, что хотя термин «минимизация функций» достаточно широко распространен, он не вполне точно отражает существо процесса минимизации. Ведь в таком процессе сама функция $f(x)$ остается неизменной.

Речь идет не о минимизации функции, а о минимизации ее значения (в результате выбора соответствующей точки в допустимом множестве значений ее аргумента).

8.1.2. Оптимизация функций и оптимизация функционалов. В практических оптимизационных задачах приходится иметь дело

с двумя основными видами их постановок. В первом случае задача ставится в обычном (евклидовом) пространстве конечной размерности. Точками x допустимого множества будут кортежи $x = \langle x_1, x_2, \ldots, x_n \rangle$ вещественных чисел, целевой же функцией $f(x) = F(x_1, \ldots, x_n)$ будет обычная вещественная функция от n вещественных аргументов (n — размерность пространства). Такую задачу мы будем именовать в дальнейшем *задачей оптимизации функций*. Во втором случае постановки оптимизационной задачи в качестве допустимого множества выступает некоторое множество M функций вещественных переменных $y(x_1, \ldots, x_m)$, а целевая функция есть некоторый *функционал* F, сопоставляющий каждой функции $y = y(x_1, \ldots, x_m)$ некоторое вещественное число $F(y)$. Такую задачу мы будем называть *задачей оптимизации функционалов* или *вариационной задачей*.

С абстрактной точки зрения обе указанные постановки совершенно идентичны: в обоих случаях речь идет о выборе точки в некотором пространстве, оптимизирующей значение целевой функции. Различие состоит лишь в природе самого пространства, однако этого достаточно, чтобы вызвать весьма существенные различия в практических методах оптимизации, применяемых в этих двух случаях. Заметим, что в принципе вторая постановка может быть с определенными допусками сведена к первой. Это возможно, если каждая функция $y(x_1, \ldots, x_m)$, составляющая допустимое множество, может быть с достаточной для практики степенью точности аппроксимирована множеством своих значений на некотором конечном множестве точек. Впрочем, в большинстве случаев число таких точек должно быть настолько велико, что сведение вариационной задачи к задаче оптимизации (обычных) функций становится мало практичным.

8.1.3. Многокритериальная оптимизация. В практике оптимизации часто приходится встречаться со случаем, когда вместо одной целевой функции $f(x)$ задано несколько целевых функций (критериев) $f_1(x), \ldots, f_k(x)$. Возникающая в этом случае задача *многокритериальной оптимизации* имеет несколько постановок. В одной из них требуется оптимизировать один из критериев, скажем $f_1(x)$, удерживая остальные критерии в заданных границах: $a_i \leqslant f_i(x) \leqslant b_i$ $(i = 2, 3, \ldots, k)$. В этом случае фактически речь идет об обычной *однокритериальной оптимизации*. Что же касается неравенств, ограничивающих другие критерии, то их можно рассматривать как дополнительные ограничения на допустимую область M. Правда, на практике в ряде случаев подобная задача решается несколько иными методами, чем обычная однокритериальная задача.

Вторая постановка состоит в *упорядочении* заданного множества критериев и в последовательной оптимизации по каждому из них. Иными словами, произведя оптимизацию по первому крите-

рию $f_1(x)$, мы получаем некоторое множество $M_1 \subset M$, на котором функция $f_1(x)$ принимает оптимальное (экстремальное) значение. Принимая его за новое допустимое множество, производим оптимизацию по второму критерию, получая в результате новое допустимое множество $M_2 \subset M_1$. Продолжая этот процесс, получим после оптимизации по последнему критерию $f_k(x)$ множество M_k, которое и будет конечным результатом многокритериальной оптимизации. Ясно, что если на некотором шаге i $(i < k)$ множество M_i сведется к одной точке, то процесс оптимизации можно закончить, поскольку $M_i = M_{i+1} = \ldots = M_k$. Не исключено, разумеется, что, как и в случае обычной однокритериальной оптимизации, задача может вообще не иметь решения.

Третья постановка применяет процесс сведения многих критериев к одному за счет введения априорных *весовых коэффициентов* λ_i для каждого из критериев $f_i(x)$. В качестве таких коэффициентов могут быть в принципе выбраны любые вещественные числа. Их значения выбираются исходя из интуитивного представления о степени важности различных критериев: более важные критерии получают веса с бóльшими абсолютными значениями. После установления весов λ_i многокритериальная задача сводится к однокритериальной с целевой функцией $f(x) = \lambda_1 f_1(x) + \ldots + \lambda_k f_k(x)$.

Вместо простой линейной комбинации исходных критериев могут использоваться и более сложные способы формирования из них нового критерия для однокритериальной оптимизации. Ниже будет показан другой путь многокритериальной оптимизации в рамках так называемой *системной оптимизации*.

8.1.4. Стандартные формы задач оптимизации функций. В стандартных формах объектом оптимизации является непрерывная функция $f(x)$ вещественных переменных $x = \langle x_1, \ldots, x_n \rangle$, допустимая область M задается конечной системой равенств и неравенств вида $p(x) = 0$, $q(x) \geqslant 0$, $r(x) \leqslant 0$ с непрерывными левыми частями. Если при этом область M ограничена, то в ней обязательно существует по крайней мере одна точка абсолютного максимума и по крайней мере одна точка абсолютного минимума функции $f(x)$.

Поскольку перемена знака у левых частей неравенств $q(x) \geqslant 0$ и $r(x) \leqslant 0$ меняет знаки этих неравенств на противоположные, можно ограничиться одним из этих двух типов неравенств. Обычно при максимизации пользуются неравенствами вида $q(x) \geqslant 0$, а при минимизации — неравенствами вида $r(x) \leqslant 0$. Таким образом, возникают две стандартные формы постановки задачи оптимизации функций.

В *задаче максимизации* требуется найти максимум функции $f(x)$ в области M, заданной системой равенств $p_i(x) = 0$ $(i = 1, 2, \ldots, l)$ и неравенств $p_i(x) \geqslant 0$ $(i = l+1, \ldots, m)$.

В *задаче минимизации* требуется найти минимум функции $f(x)$ в области M, заданной системой равенств $p_i(x)=0$ $(i=1, 2, \ldots, l)$ и неравенств $p_i(x)\leqslant 0$ $(i=l+1, \ldots, m)$.

Легко видеть, что при одновременной перемене знаков у функций $f(x)$ и $p_i(x)$ $(i=1, \ldots, m)$ задача максимизации переходит в задачу минимизации и наоборот. Поэтому разработку методов оптимизации достаточно вести применительно лишь к одной из этих постановок.

Заметим также, что *ограничения типа неравенств* можно заменить *ограничениями типа равенств* и простыми координатными неравенствами, вводя дополнительные (вещественные) переменные z_i. При этом ограничение вида $p_i(x)\geqslant 0$ заменится парой ограничений $p_i(x)-z_i=0$, $z_i\geqslant 0$, а ограничение $p_i(x)\leqslant 0$ — парой ограничений $p_i(x)+z_i=0$, $z_i\geqslant 0$. Этот прием в дальнейшем будет именоваться стандартным приемом *элиминации нетривиальных неравенств*. Его особенно удобно применять в том случае, когда по смыслу задачи все точки допустимой области имеют неотрицательные координаты. В результате его применения при этих условиях возникает третья стандартная форма постановки *задачи оптимизации функций*:

Найти оптимум (максимум или минимум) функции $f(x)$ в области M, заданной системой равенств $p_i(x)=0$ $(i=1, \ldots, m)$ и простых (координатных) неравенств $x_j\geqslant 0$ $(j=1, \ldots, n)$.

Во всех перечисленных постановках может присутствовать дополнительное требование о том, чтобы все координаты точки оптимума были целыми числами. Это требование превращает задачу *непрерывной оптимизации* в задачу *целочисленной оптимизации*. В случае, когда допустимая область M ограничена, в ней может располагаться лишь конечное множество точек с целочисленными координатами. Поэтому задача целочисленной оптимизации в ограниченной области в принципе может быть решена простым *методом перебора*, т. е. вычисления значения целевой функции (критерия) $f(x)$ во всех *допустимых точках* и выбора из них точки (или точек) с оптимальными значениями критерия.

8.2. Гладкая оптимизация. Общие свойства

В случае, когда функция цели $f(x)$ и функции $p_i(x)$, задающие ограничения, являются дифференцируемыми (гладкими), для решения задач оптимизации используется понятие градиента. Для любой дифференцируемой функции $g(x)$ ее *градиентом* $\nabla g(x)$ в точке x называется вектор

$$\nabla g(x)=\begin{pmatrix}\frac{\partial g(x)}{\partial x_1}\\ \cdots\\ \frac{\partial g(x)}{\partial x_n}\end{pmatrix}. \tag{8.1}$$

Вектор градиента $\nabla g(x)$ задает (в заданной точке x) направление наискорейшего роста функции $g(x)$, а обратное ему направление $-\nabla g(x)$, называемое *антиградиентом*,— направление наискорейшего убывания этой функции. Разумеется, эти направления определяются лишь в том случае, когда вектор градиента отличен от нуля. Точки, в которых градиент функции обращается в нуль, называются ее *стационарными точками*. Если экстремум дифференцируемой функции $f(x)$ достигается внутри допустимой области (не на ее границе), то в точке оптимума a ее градиент обращается в нуль, т. е. имеет место система равенств

$$\left.\frac{\partial f(x)}{\partial x_1}\right|_{x=a} = 0, \ldots, \left.\frac{\partial f(x)}{\partial x_n}\right|_{x=a} = 0. \qquad (8.2)$$

Эти равенства (*условия стационарности* функции) имеют место не только для абсолютных, но и для локальных экстремумов. Вместе с тем условия стационарности не являются достаточными. Иными словами, при их выполнении в некоторой точке x_0 она не обязательно будет экстремальной точкой. Возможно, что это так называемая *седловая точка*, пример которой (точка A) дается на рис. 8.2. В одних направлениях от такой точки происходит *подъем* (увеличение значения функции), в других — *спуск* (уменьшение значения функции), причем касательная плоскость в точке A горизонтальна, что будет, очевидно, в любой стационарной точке. Для отличия седловых точек от экстремальных требуется исследовать значения вторых производных (а иногда и производных более высоких порядков) в стационарной точке.

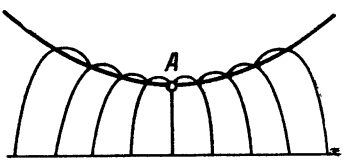

Рис. 8.2

8.2.1. Условия Куна — Таккера. В случае, когда экстремальная точка a лежит на границе допустимого множества, она не обязана быть стационарной. Однако при некотором дополнительном условии удается, заменив целевую функцию $f(x)$ так называемой *функцией Лагранжа* $L(x) = f(x) + \sum_{i=1}^{m} \lambda_i p_i(x)$ (где $p_i(x)$ — левые части всех граничных условий), добиться того, чтобы граничные экстремальные точки функции $f(x)$ были стационарными точками функции Лагранжа $L(x)$ при подходящих значениях параметров λ_i.

Условие, о котором идет речь,— так называемое *условие регулярности*, выполняется на практике. Его суть состоит в линейной независимости в данной точке $x = a$ градиентов всех граничных функций $p_i(x)$, для которых $p_i(a) = 0$.

При выполнении этого условия для любой точки максимума x^* в задаче «найти $\max f(x)$ при $p_i(x) = 0$ $(i = 1, \ldots, l)$, $p_i(x) \geqslant 0$

$(i=l+1, \ldots, m)$» и для любой точки минимума x^* в задаче «найти $\min f(x)$ при $p_i(x)=0$ $(i=1, \ldots, l)$, $p_i(x) \leqslant 0$ $(i=l+1, \ldots, m)$» должны соблюдаться следующие три условия, называемые *условиями Куна — Таккера*:

1) x^* лежит в допустимом множестве;

2) $\lambda_i p_i(x^*)=0$ при $i=1, \ldots, m$;

3) $\nabla f(x^*) + \sum\limits_{i=1}^{m} \lambda_i \nabla p_i(x^*) = 0.$

Здесь λ_i — некоторые вещественные числа (*множители Лагранжа*), произвольные для всех условий типа равенства $(i=1, \ldots, l)$ и неотрицательные для всех условий типа неравенства $(i=l+1, \ldots, m)$.

Заметим, что этими условиями определяются лишь точки возможных (локальных и абсолютных) экстремумов, исключая из рассмотрения все точки, в которых таких экстремумов заведомо быть не может. Однако их выполнения, вообще говоря, недостаточно, чтобы такие экстремумы действительно имели место. Что касается абсолютного экстремума, то он, как правило, легко находится путем фактического вычисления и последующего сравнения между собой значений целевой функции во всех точках, найденных из условий Куна — Таккера.

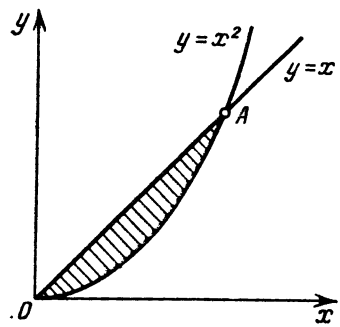

Рис. 8.3

Поясним сказанное на примере. Пусть требуется отыскать абсолютный максимум функции $z=x+y$ в области M, заданной неравенствами $x-y \geqslant 0$ и $y-x^2 \geqslant 0$. Эта область изображена на рис. 8.3: неравенство $x-y \geqslant 0$ задает множество точек, лежащих вправо от прямой $y=x$, а неравенство $y-x^2 \geqslant 0$ — множество точек, лежащих вверх от параболы $y=x^2$. Обозначая множители Лагранжа через λ и μ, из условий Куна — Таккера получим следующую систему соотношений:

$$x-y \geqslant 0,\ y-x^2 \geqslant 0,$$

$$\lambda(x-y)=0,\ \mu(y-x^2)=0,$$

$$\frac{\partial}{\partial x}(x+y+\lambda(x-y)+\mu(y-x^2))=0,$$

$$\frac{\partial}{\partial y}(x+y+\lambda(x-y)+\mu(y-x))=0,$$

$$\lambda \geqslant 0,\ \mu \geqslant 0,$$

или

$$x-y\geqslant 0,\ y-x^2\geqslant 0,\ \lambda(x-y)=0,$$
$$\mu(y-x^2)=0,\ 1+\lambda-2\mu x=0,\ 1-\lambda+\mu=0,$$
$$\lambda\geqslant 0,\ \mu\geqslant 0.$$

Легко проверить, что этим соотношениям удовлетворяет единственная точка $x=1$, $y=1$, которая, удовлетворяя условию регулярности, должна быть точкой абсолютного максимума (ибо в силу п. 8.1.4 такая точка в нашем случае обязательно существует). При этом $\lambda=3$, $\mu=2$.

Если искать точку минимума той же функции в той же области, то нужно изменить форму представления граничных условий: вместо неравенств $x-y\geqslant 0$ и $y-x^2\geqslant 0$ использовать эквивалентные им неравенства $y-x\leqslant 0$ и $x^2-y\leqslant 0$. Условия Куна — Таккера приводят к системе соотношений $y-x\leqslant 0$, $x^2-y\leqslant 0$, $\lambda(y-x)=0$, $\mu(x^2-y)=0$,

$$\frac{\partial}{\partial x}(x+y+\lambda(y-x)+\mu(x^2-y))=1-\lambda+2\mu x=0,$$

$$\frac{\partial}{\partial y}(x+y+\lambda(y-x)+\mu(x^2-y))=1+\lambda-\mu=0.$$

Этой системе удовлетворяет единственный набор значений, а именно: $x=0$, $y=0$, $\lambda=1$, $\mu=2$. Ясно, что точка $x=0$, $y=0$ в силу ее единственности и выполнения условия регулярности есть точка абсолютного минимума функции $z=x+y$ в области M.

Заменяя в последнем варианте задачи целевую функцию на $z=x-y$, придем к системе соотношений $y-x=0$, $x^2-y=0$, $\lambda(y-x)=0$, $\mu(x^2-y)=0$, $1-\lambda+2\mu x=0$, $-1+\lambda-\mu=0$. Они удовлетворяются лишь при $y=x$, $\lambda=1$, $\mu=0$. Поскольку во всех точках, для которых $y=x$, целевая функция $z=x-y$ принимает одно и то же значение $z=0$, а условие регулярности выполняется, приходим к выводу о том, что она достигает абсолютного минимума (в заданной области M) во всех точках отрезка, соединяющего начало координат с точкой $A(1, 1)$ (рис. 8.3).

Для демонстрации применения условий Куна — Таккера намеренно были взяты тривиальные примеры, для которых решение поставленных задач легко проводится на основе наглядных геометрических представлений. Ясно, однако, что возможность использования таких представлений имеется лишь в самых простых, исключительных случаях. Условия же Куна — Таккера работают во всех случаях, когда возможно явное аналитическое выражение градиента функции Лагранжа.

8.2.2. Достаточные условия для экстремума гладкой функции. В приведенных выше примерах нами фактически уже использовалось одно из достаточных условий экстремума гладкой функции:

точка $x = a$ будет точкой экстремума, если она является единственной точкой, удовлетворяющей условиям Куна — Таккера, а допустимая область ограничена.

Более общая форма достаточного условия экстремума получается при представлении функции Лагранжа $f(x) + \sum_{i=1}^{m} \lambda_i p_i(x)$ в виде функции $L(x, \lambda)$ от двух групп переменных: $x = (x_1, \ldots, x_n)$, $\lambda = (\lambda_1, \ldots, \lambda_m)$, с условиями $\lambda_1 \geqslant 0, \ldots, \lambda_m \geqslant 0$ (все ограничения предполагаются имеющими тип неравенств). Достаточное условие экстремума состоит в том, что для экстремальной точки $x = x^*$ существует допустимый набор $\lambda = \lambda^*$, для которого точка (x^*, λ^*) является *седловой точкой* функции Лагранжа. Это означает, что в случае, когда x^* есть точка максимума, имеют место соотношения

$$\max_{x \in X} L(x, \lambda^*) = L(x^*, \lambda^*) = \min_{\lambda \in \Lambda} L(x^*, \lambda), \tag{8.3}$$

а в случае точки минимума — соотношения

$$\min_{x \in X} L(x, \lambda^*) = L(x^*, \lambda^*) = \max_{\lambda \in \Lambda} L(x^*, \lambda). \tag{8.4}$$

При этом x пробегает все пространство X, а λ — множество Λ всех $(\lambda_1, \ldots, \lambda_m)$ с неотрицательными λ_i $(i = 1, \ldots, m)$.

8.2.3. Двойственные задачи оптимизации. В случае стандартной формы задачи максимизации функции $f(x)$ так называемая *прямая функция* $L_*(x)$ получается минимизацией (по λ) функции Лагранжа $L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i p_i(x)$:

$$L_*(x) = \min_{\lambda \in \Lambda} L(x, \lambda). \tag{8.5}$$

Функция, *двойственная* прямой функции, получается максимизацией (по x) функции Лагранжа $L(x, \lambda)$:

$$L^*(\lambda) = \max_{x \in X} L(x, \lambda). \tag{8.6}$$

Здесь, в отличие от предыдущего пункта, X и Λ могут в общем случае представлять собой произвольные множества. При этом задача максимизации прямой функции $L_*(x)$ называется *прямой задачей оптимизации*, а задача минимизации двойственной ей функции — *двойственной задачей оптимизации*. Переменные $x_1, \ldots, x_n$ носят наименование *прямых*, а переменные $\lambda_1, \ldots, \lambda_m$ — *двойственных переменных*.

Нетрудно показать, что решение прямой задачи оптимизации эквивалентно задаче максимизации функции $f(x)$ при ограниче-

ниях $p_i(x) \geqslant 0$ $(i = 1, \ldots, m)$, $x \in X$. При X, совпадающем со всем пространством, приходим к первой стандартной форме оптимизационной задачи.

Для второй стандартной формы оптимизационной задачи (минимизации функции $f(x)$) все сказанное сохраняет силу при условии замены всюду процессов максимизации процессами минимизации и наоборот.

8.3. Численные методы гладкой оптимизации

Метод оптимизации, использующей условия Куна — Таккера, на практике применяется относительно редко ввиду большой сложности анализа возникающей системы соотношений. К тому же получение самих соотношений основывается на возможности аналитического представления градиентов функций $f(x)$ и $p_i(x)$ $(i = 1, \ldots, m)$, что далеко не всегда практически осуществимо. Поэтому на практике гораздо чаще применяются *численные методы* оптимизации, для которых достаточно уметь находить численные значения градиента в любой заданной точке.

Для простоты изложения вначале мы будем предполагать, что задача оптимизации не содержит ограничений, а затем покажем, каким образом к этому случаю сводятся задачи общего вида с ограничениями.

8.3.1. Общая идея градиентного спуска (подъема). Пусть непрерывно дифференцируемая целевая функция $f(x)$ задана во всех точках пространства X. Для произвольной точки x, в которой градиент $\nabla f(x)$ отличен от нуля, он задает два направления (два вектора единичной длины) $g = g(x)$ и $-g = -g(x)$, называемые соответственно *направлениями градиента* и *антиградиента*. Сдвигаясь от точки x на очень малый шаг ε в любом направлении d, где d — единичный вектор, в силу условия дифференцируемости функции $f(x)$, получим неравенства

$$f(x + \varepsilon g) \geqslant f(x + \varepsilon d) \geqslant f(x - \varepsilon g). \qquad (8.7)$$

Это означает, что движение (на очень малый шаг) в направлении градиента функции обеспечивает наибольший рост, а в направлении антиградиента — наибольшее уменьшение этой функции. Указанные направления называют соответственно направлениями *наискорейшего подъема* и *наискорейшего спуска* функции $f(x)$ в заданной точке x.

Из сказанного вытекает общая идея градиентного подъема или градиентного спуска:

Отправляясь от заданной точки x^0, строим последовательность точек $(x^0, x^1, x^2, \ldots)$ так, что перемещение от каждой точки x^{i-1} к следующей точке x^i, производится в направлении наискорейшего

подъема (при поиске максимума) или наискорейшего спуска (при поиске минимума), строящихся в исходной точке x^{i-1} *).

Трудность практического применения подобной процедуры состоит прежде всего в выборе величины шага ε_i при переходе от точки x^{i-1} к точке x^i. Дело в том, что неравенства (8.7) имеют место лишь в малой окрестности точки x, т. е. для малой величины шага ε_i. Легко видеть (см. рис. 8.4), что при большой величине шага можно, даже двигаясь в направлении наискорейшего подъема, получить не увеличение, а уменьшение значения целевой функции $f(x)$. То же самое имеет, разумеется, место и для процедуры спуска.

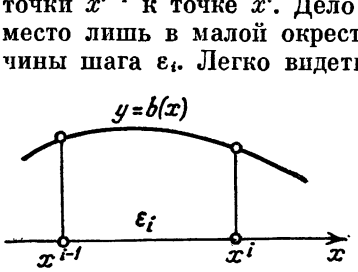

Рис. 8.4

Из сказанного следует, что величину шага ε_i в процедурах наискорейшего подъема и спуска следует выбирать достаточно малой. Однако при движении малыми шагами для приближения к экстремальной точке может потребоваться очень большое число шагов. Кроме того, от процедур подъема и спуска обычно требуется не просто приближение к экстремальной точке $x=a$, а сходимость к ней. Это означает, что по мере роста i расстояние $|x^i-a|$ от точки x^i до точки экстремума $x=a$ должно стремиться к нулю. Это возможно лишь в том случае, если по мере приближения к точке экстремума величина шага ε_i будет также стремиться к нулю.

В следующих двух пунктах мы рассмотрим две основные процедуры, обеспечивающие при определенных условиях сходимость к точке экстремума.

8.3.2. Пропорционально-градиентный метод. Наиболее простым способом обеспечения требуемого уменьшения шага при приближении к точке экстремума для дифференцируемой функции $f(x)$ является выбор длины шага ε_i пропорциональным длине вектора градиента в точке x^{i-1}:

$$\varepsilon_i=\alpha|\nabla f(x^{i-1})|,\ \alpha>0,\ \alpha=\text{const}. \tag{8.8}$$

Такой выбор шага означает, что переход от точки x^{i-1} к точке x^i будет выполняться по формуле

$$x^i=x^{i-1}+\alpha\nabla f(x^{i-1}) \tag{8.9}$$

для процедуры подъема (нахождения максимума функции $f(x)$) и по формуле

$$x^i=x^{i-1}-\alpha\nabla f(x^{i-1}) \tag{8.10}$$

*) Использование верхних индексов для обозначения различных точек пространства X вызвано тем, что нижние индексы употребляются для обозначения координат точки $x=(x_1, x_2, \ldots, x_n)$.

для процедуры спуска (нахождения минимума функции $f(x)$).

Заметим, что для произвольной дифференцируемой функции при произвольном выборе начальной точки x^0 описанная процедура не обязательно приводит к экстремальной точке. Такой точкой может оказаться любая стационарная точка, в окрестности которой величина шага ε_i также стремится к нулю. Однако, если начальное приближение x^0 выбрано достаточно близко к точке экстремума $x=a$, то, как правило, такая сходимость может быть обеспечена при подходящем выборе константы α. Понятие «достаточно близко» может быть уточнено, например, таким образом, что точка x^0 попадает внутрь сферы с центром в экстремальной точке $x=a$, являющейся единственной стационарной точкой в этой сфере.

Обычно же описанная процедура применяется к весьма часто встречающимся на практике задачам выпуклой оптимизации, в которых при условии отсутствия ограничений точка экстремума (притом абсолютного) единственна во всем пространстве. Об этих задачах (для случая гладких функций), где автоматически обеспечивается сходимость описываемых градиентных методов, говорится ниже.

8.3.3. Полношаговый градиентный метод. В этом методе каждый шаг градиентного подъема или спуска делается на максимально возможную длину, обеспечивающую требуемое направление изменения значения функции (т. е. ее увеличение или уменьшение). Иными словами, на полупрямой, исходящей из очередной точки x^{i-1} в направлении градиента (при подъеме) или антиградиента (при спуске), ищется соответственно точка абсолютного максимума или абсолютного минимума, которая и выбирается в качестве следующей точки x^i.

В случае произвольной дифференцируемой функции $f(x)$ решение подобной задачи, называемой *одномерной оптимизационной задачей*, само по себе достаточно трудно. Ее решение упрощается при наложении на функцию $f(x)$ некоторых дополнительных требований, например, выпуклости, о чем будет подробнее рассказано ниже. Сейчас же мы ограничимся наглядным пояснением работы полношагового градиентного метода.

Пусть, например, речь идет о минимизации функции $z=x^2+4y^2$. Чтобы не строить пространственный чертеж, изобразим процесс минимизации на плоскости, задав функцию z ее линиями уровня, т. е. линиями, задаваемыми уравнениями $x^2+4y^2=c$ при различных значениях константы c (рис. 8.5). Из начальной точки x^0 движение по направлению антиградиента d^1 происходит до точки x^1, в которой полупрямая, исходящая из точки x^0 в направлении d^1, касается некоторой линии уровня. Далее тот же процесс повторяется для точки x^1 и т. д. В рассматриваемом случае оче-

видно, что получаемый ряд точек x^0, x^1, x^2, ... сходится к началу координат, представляющему собой точку абсолютного минимума заданной функции $z = x^2 + 4y^2$.

8.3.4. Метод сопряженных градиентов. Полношаговый подъем (или спуск) может, очевидно, осуществляться не обязательно по направлению градиента (или антиградиента), а по любому направлению, в котором целевая функция $f(x)$ меняется в нужную сторону (увеличивается или уменьшается). Вместе с тем очевидно, что, будучи наилучшими при малой величине шага, направления градиента и антиградиента вовсе не обязаны быть таковыми при больших шагах. Так из рис. 8.5 видно, что если в точке x^0 «довернуть» направление антиградиента немного вправо, то, сделав полный шаг по исправленному таким образом направлению, мы получим лучшее приближение к экстремуму, чем точка x^1.

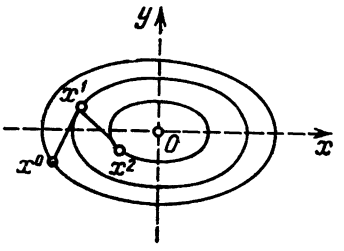

Рис. 8.5

В случае, когда функция $f(x)$ является квадратным полиномом (как это имеет место в примере предыдущего пункта), разработана точная процедура таких «доворотов», получившая наименование *метода сопряженных градиентов*, при которой (при полных шагах) точки экстремума достигаются из любой начальной точки x^0 за n шагов (где n — размерность пространства). Направление d^{i+1} шага от точки x^i к точке x^{i+1} задается в этом методе следующей рекуррентной формулой:

$$d^{i+1} = \pm \nabla f(x^i) + \frac{|\nabla f(x^i)|^2}{|\nabla f(x^{i-1})|^2} d^i, \qquad i = 1, 2, \ldots, \tag{8.11}$$

где $|\nabla f(x)|$ означает длину вектора $\nabla f(x)$. Знак плюс выбирается в том случае, когда решается задача максимизации, а знак минус — в случае решения задачи минимизации. Тот же самый выбор знака осуществляется и для направления d^1 первого шага ($i = 0$). Оно совпадает с направлением $\pm\nabla f(x^0)$.

Вычисляя по приведенным правилам направления d^1, d^2, ... и производя по каждому из них полный шаг, после не более чем n шагов придем к искомой экстремальной точке задачи квадратичной оптимизации.

Чтобы иллюстрировать сказанное, рассмотрим задачу минимизации функции $z = x^2/2 + y^2$. Ее антиградиент $-\nabla z$ есть вектор $-\nabla z = (-x, -2y)$. Пусть дана начальная точка $A_0 = (1, 1/2)$; тогда $d^1 = -\nabla z_0 = (-1, -1)$. Полупрямая, по которой делается первый шаг минимизации, задается уравнениями $x = 1 - t$, $y = 1/2 - t$ ($t \geqslant 0$). По направлению этой полупрямой $z = (1-t)^2/2 + (1/2 - t)^2$. Минимум этой функции достигается при $t = 2/3$,

так что после первого шага мы попадаем в точку A_1 с координатами $x = 1 - 2/3 = 1/3$, $y = 1/2 - 2/3 = -1/6$. В этой точке $-\nabla z_1 = (-1/3, 1/3)$ и отношение $l = \frac{|-\nabla z_1|^2}{|-\nabla z_0|^2} = \frac{1}{9}$, так что $d^2 = -\nabla z_1 + l_1(-\nabla z_0) = \left(-\frac{1}{3}, +\frac{1}{3}\right) + \frac{1}{9}(-1, -1) = \left(-\frac{4}{9}, \frac{2}{9}\right) = \frac{2}{9}(-2, 1)$. Двигаясь по полупрямой $x = \frac{1}{3} - 2t$, $y = -\frac{1}{6} + t$ $(t \geqslant 0)$, мы достигнем минимума функции $z = \frac{1}{2}\left(\frac{1}{3} - 2t\right)^2 + \left(-\frac{1}{6} + t\right)^2$ при $t = 1/6$. В результате придем в точку A_2 с координатами (0, 0), в которой, как это совершенно очевидно, и достигается минимум исходной функции.

В случае произвольной (не квадратичной) целевой функции $f(x)$ метод сопряженных градиентов в большинстве случаев также приводит к лучшим результатам, чем простой полношаговый градиентный метод. Такое положение обуславливается тем, что квадратичное приближение лучше аппроксимирует целевую функцию $f(x)$, чем линейное приближение, фактически используемое в простых градиентных методах. Разумеется, в общем случае не может быть гарантировано получение точного решения за конечное число шагов, как это имеет место в случае квадратичной целевой функции.

8.3.5. Методы сведения общей задачи оптимизации к задаче без ограничений. Описанные выше численные методы гладкой оптимизации в чистом виде применяются обычно для задач без ограничений. Задачи общего вида (с ограничениями) удается сводить к этому случаю за счет изменения целевой функции. Пусть, например, требуется найти максимум функции $f(x)$ при ограничениях $p_i(x) \geqslant 0$ $(i = 1, \ldots, m)$. В так называемом *методе штрафных функций* строится функция $P(x)$, называемая *штрафной функцией*, которая внутри допустимой области M принимает нулевое значение, а вне ее — отрицательна. Чаще всего в качестве такой функции выбирают функцию вида

$$P(x) = -\sum_{i=1}^{m} |\min[p_i(x), 0]|^l, \quad l \geqslant 1.$$

Выбрав штрафную функцию $P(x)$, строят последовательность положительных чисел $r_1 > r_2 > \ldots > r_k > \ldots$, сходящуюся к нулю. Заменяя целевую функцию $f(x)$ функцией

$$F_k(x) = f(x) + \frac{1}{r_k} P(x), \qquad k = 1, 2, \ldots,$$

решают задачу без ограничений для этой функции (начиная с

$k=1$). Если решение x^k принадлежит допустимой области (т. е. если все $g_i(x^k)\geqslant 0$), то x^k является, очевидно, решением исходной задачи с ограничениями. В противном случае заменяют k на $k+1$ и решают задачу для новой целевой функции $F_k(x)$.

При непрерывности функций $f(x)$ и $g_i(x)$ и ограниченности исходной допустимой области M обычно нетрудно выбрать штрафную функцию $P(x)$ так, чтобы все экстремумы x^k существовали, а последовательность $\{x^k\}$ была ограниченной. В таком случае можно показать, что эта последовательность при $k\to\infty$ сходится к точке экстремума (в данном случае максимума) функции $f(x)$ в области M, т. е. к решению задачи оптимизации в ее исходной постановке.

В так называемом *методе барьеров* при решении задачи $\max f(x)$ при $p_i(x)\geqslant 0$ $(i=1, \ldots, m)$ целевая функция $f(x)$ заменяется функцией

$$F_k(x)=f(x)+r_kB(x), \quad r_k>0,$$

где барьерная функция $B(x)$ характеризуется свойством стремиться к $-\infty$ при приближении x к границе допустимой области M изнутри. Иными словами, $B(x)\to-\infty$ при $p_i(x)\to 0$ хотя бы для одного $i=1, \ldots, m$. Такую функцию можно задать, например, формулой

$$B(x)=-\sum_{i=1}^{m}\frac{1}{p_i(x)},$$

если область M задана неравенствами $p_i(x)\geqslant 0$ $(i=1, \ldots, m)$.

Как и в методе штрафных функций, выбирается последовательность $\{r_k\}$ положительных чисел $r_1>r_2>\ldots>r_k>\ldots$, сходящаяся к нулю. Решаются задачи без ограничений для целевых функций $F_k(x)=f(x)+r_kB(x)$ при $k=1, 2, \ldots$ Если при этом используется тот или иной численный метод, в котором начальная точка выбирается внутри M, то наличие барьера обеспечит, что экстремальная точка x^k (при ее существовании) также будет лежать внутри области M. Наличие же экстремума обеспечивается условиями ограниченности области M и непрерывности функций $f(x)$ и $B(x)$.

Решая задачу (без ограничений) при $k=1, 2, \ldots$, мы находим последовательность экстремальных точек $\{x^k\}$, сходящуюся к экстремальной точке исходной задачи. Например, решая задачу $\max x$ при $1-x\geqslant 0$, заменяем ее последовательностью задач $\max\left(x-\frac{r_k}{1-x}\right)$. Для таких задач точка максимума (при движении изнутри) будет задаваться формулой $x^k=1-\sqrt{r_k}$. При $r_k\to 0$ $x^k\to 1$. Точка $x=1$ есть, очевидно, решение исходной задачи. При переходе от задач максимизации к задачам минимизации

достаточно переменить знак барьерной функции. Все остальное будет делаться точно так же, как и в случае задачи максимизации.

Задачу с ограничениями можно решать и непосредственно методами градиентного спуска (подъема). При этом, однако, на каждом шаге нужно выбирать очередную точку при соблюдении условия ее принадлежности допустимой области. Кроме того, при попадании точки на границу области очередной шаг, как правило, уже не может быть сделан в направлении градиента (антиградиента) целевой функции, поскольку в этом направлении может не существовать других допустимых точек (направление ведет вовне допустимой области). Поэтому «чистые» градиентные методы заменяются движением в любом *возможном* (*допустимом*) *направлении*, т. е. в направлении, которое, во-первых, ведет внутрь допустимой области, а во-вторых, обеспечивает нужное направление изменения целевой функции.

Такое видоизменение градиентного метода приводит, однако, к дополнительным трудностям в связи с возможностью так называемого з а к л и н и в а н и я. Сущность заклинивания легко уяснить на простейшем примере максимизации функции $z = y$ в области $1 - x^2 - y^2 \geqslant 0$, т. е. в единичном круге с центром в начале координат (рис. 8.6). Ясно, что точкой максимума является точка C. Если же двигаться к ней из точки $A = A_0$ так, что каждая следующая точка делит пополам дугу $A_k B$, то последовательность точек $\{A_k\}$ будет сходиться к B, которая точкой максимума не является.

Рис. 8.6

Для устранения заклинивания применяются специальные методы, обеспечивающие нужную степень «доворота» возможных направлений внутрь допустимой области.

8.4. Выпуклая оптимизация

Задача оптимизации может быть значительно упрощена при наложении на целевую функцию $f(x)$ и допустимую область M некоторых ограничений. Одним из таких ограничений является так называемое *условие выпуклости*.

Множество M евклидова пространства называется *выпуклым*, если для любых двух точек x^1 и x^2 этого множества все точки соединяющего их отрезка $[x^1, x^2]$ также принадлежат множеству M. Из этого определения автоматически следует, что пересечение любого числа выпуклых множеств также выпукло. Важность этого свойства обуславливается тем обстоятельством, что, как уже отмечалось выше, допустимая область задается обычно системой ра-

венств и неравенств ($p_i(x)=0$, $p_j(x)\geqslant 0$, $p_k(x)\leqslant 0$). Тогда для того, чтобы такая система определяла выпуклое множество, достаточно, очевидно, чтобы область, задаваемая каждым отдельным равенством или неравенством, была выпуклой. Это автоматически имеет место для так называемых *линейных ограничений*, т. е. таких ограничений, у которых функции $p_i(x)$ имеют вид $p_i(x)=a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n+b_i$, где $x_1, \ldots, x_n$ — координаты точки x, и все коэффициенты a_{ij} и b_i постоянны.

Функция $f(x)$ называется *выпуклой* или, более точно, *выпуклой вниз*, если она определена на выпуклом множестве M и для любых двух точек x^1 и x^2 этого множества значение $f(x)$ функции в любой точке x отрезка $[x^1, x^2]$ не превышает значения в той же точке, определенной на данном отрезке линейной функции со значениями $f(x_1)$ и $f(x_2)$ в его концевых точках. Иными словами, если употреблять понятия «над» и «под» применительно к направлению вдоль оси z, отрезок, соединяющий точки $A=(z_1, x^1)$ и $B=(z_2, x^2)$ в пространстве с координатами $(z, x_1, \ldots, x_n)$, лежит над гиперповерхностью $z=f(x)$. Для случая $n=1$ это иллюстрируется рис. 8.7 и в общем случае выражается неравенством $\alpha f(x^1)+(1-\alpha)f(x^2)\geqslant f(\alpha x^1+(1-\alpha)x^2)$ при $0\leqslant\alpha\leqslant 1$. Легко видеть, что при $0\leqslant\alpha\leqslant 1$ выражение $\alpha x^1+(1-\alpha)x^2$ задает различные точки отрезка $[x^1, x^2]$. Из вида приведенного неравенства следует, что любая линейная функция выпукла.

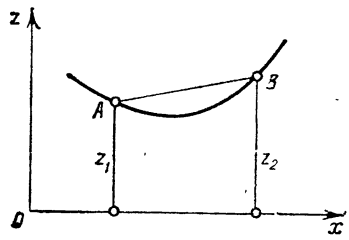

Рис. 8.7

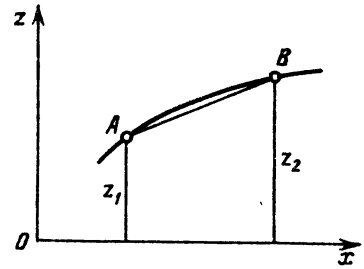

Рис. 8.8

Если в приведенном определении заменить знак неравенства $\geqslant$ на $\leqslant$, то получим определение вогнутой функции, или, что одно и то же, функции, *выпуклой вверх*. В двумерном случае возможный график такой функции иллюстрирует рис. 8.8. Заметим, что согласно приведенным определениям любая линейная функция может считаться как выпуклой, так и вогнутой.

Из определений выпуклости и вогнутости функций легко вывести, что такие функции обязательно непрерывны (хотя и не обязательно дифференцируемы). Далее, для выпуклой функции $f(x)$ область, состоящая из всех точек x, в которых $f(x)\leqslant a$, где a — некоторая константа, представляет собой выпуклое (и притом

замкнутое) множество. То же самое имеет место и в случае вогнутой функции $g(x)$ для неравенства $g(x) \geqslant b$, где b — константа. Заметим, что в случае $a < \min f(x)$ и $b > \max g(x)$ оба эти множества могут оказаться пустыми, что формально не мешает им продолжать оставаться выпуклыми и замкнутыми.

Условимся называть полученные области *областями уровня*, а их границы — *гиперповерхностями* (при $n = 2$ — *линиями*, при $n = 3$ *поверхностями*) *уровня* рассматриваемых функций $f(x) = f(x_1, \ldots, x_n)$ и $g(x) = g(x_1, \ldots, x_n)$.

Для выпуклой функции $f(x)$ в любой точке x^0, лежащей в области определения M функции, существует вектор $r(x^0)$, для которого при любом x из M выполняется неравенство

$$f(x) - f(x^0) \geqslant (r(x^0), x - x^0).$$

В правой части этого неравенства стоит скалярное произведение векторов $r(x^0)$ и $x - x^0$, т. е. произведение длин этих векторов на косинус угла между ними. Вектор $r(x^0)$, удовлетворяющий этому свойству, называется *обобщенным градиентом* (*субградиентом*) функции $f(x)$ в точке x^0. Обычный градиент (в случае его существования и отличия от нуля) является также обобщенным градиентом (причем единственным обобщенным градиентом) в рассматриваемой точке. Если функция $f(x)$ не дифференцируема в некоторой точке x^0, в ней существует, вообще говоря, некоторое множество обобщенных градиентов, называемое *субградиентным множеством* функции $f(x)$ в точке x^0.

В большинстве случаев при оптимизации приходится иметь дело с функциями, имеющими градиент почти всюду. Это означает, что множество M' точек, в которых оптимизируемая функция не дифференцируема, имеет объем (меру), равный нулю. В случае функции одной переменной множество M' состоит обычно из конечного числа точек, в случае функции двух переменных — из конечного числа линий (и, может быть, отдельных точек) и т. д. При соблюдении подобного условия любая точка x области определения M функции $f(x)$ может быть представлена как предел последовательности точек $p = \{x^1, x^2, \ldots, x^k, \ldots\}$ этой области, в каждой из которых функция $f(x)$ дифференцируема и, следовательно, имеет градиент $r_k = r(x^k)$. В большинстве встречающихся на практике случаев последовательность p может быть выбрана таким образом, что последовательность градиентов $\{r_1, r_2, \ldots, r_k, \ldots\}$ сходится к некоторому вектору $r = r(x)$, называемому *почти-градиентом* функции $f(x)$ в точке x.

Принадлежа к классу обобщенных градиентов, почти-градиенты не исчерпывают, однако, всего этого класса. Они составляют как бы границы этого класса, что в двумерном случае наглядно иллюстрирует рис. 8.9. На этом рисунке заштрихованная площадь представляет собой некоторую область уровня $f(x) \geqslant a$

выпуклой функции $f(x)$ двух переменных $x=(x_1, x_2)$. Линия AOD представляет собой множество M' точек, в которых функция $f(x)$ не дифференцируема и, следовательно, не имеет (обычного) градиента. В точке A имеются два почти-градиента этой функции, изображенные на рисунке векторами AB и AC. Первый из них представляет нормаль к линии AKD, а второй — к линии ALD. Они получаются в качестве пределов градиентов $r(x^k)$ функции $f(x)$, когда последовательности точек $\{x^k\}$, сходящиеся к точке x, выбираются соответственно на линиях AKD и ALD. Множество обобщенных градиентов функции $f(x)$ в точке A занимает весь сектор ABC.

8.4.1. Простой субградиентный метод выпуклой оптимизации. Как легко видеть из рис. 8.9, при движении в направлении, противоположном обобщенному градиенту, значение выпуклой целевой функции $f(x)$ не обязательно убывает. Иными словами, это направление не обязательно ведет внутрь области уровня. Подобное положение имеет место, например, для почти-градиентов AB и AC. Вместе с тем нетрудно показать, что при небольшом перемещении в направлении, обратном любому обобщенному градиенту, например, при перемещении из точки A в точку E (см. рис. 8.9), расстояние до точки минимума O обязательно уменьшается (OE меньше, чем OA). Это обстоятельство обусловлено тем, что любая точка минимума O выпуклой функции $f(x)$ лежит в пределах любой непустой области уровня. В силу же выпуклости этой области угол OAE будет непременно острым.

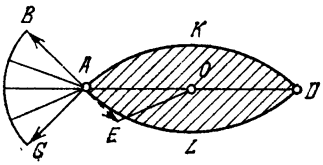

Рис. 8.9

Из приведенных рассмотрений легко приходим к одной из простых идей метода субградиентной оптимизации. Для реализации субградиентной оптимизации в случае задачи без ограничений процесс может начинаться с любой точки x^0. Сдвиг от очередной точки x^i к следующей точке x^{i+1} осуществляется на расстояние l_i в направлении любого обобщенного градиента функции $f(x)$ в точке x^i в случае задачи максимизации вогнутой функции и в обратном направлении — в случае задачи минимизации выпуклой функции. Если шаги l_i сдвигов выбираются таким образом, что $l \to 0$ при $i \to \infty$, а ряд $l_0 + l_1 + l_2 + \ldots$ расходится, то последовательность $\{x^0, x^1, \ldots, x^m, \ldots\}$ сходится к множеству экстремума заданной функции $f(x)$ при условии, что множество ограничено.

Заметим, что в задачах выпуклой оптимизации без ограничений любая экстремальная точка является точкой абсолютного экстремума. Что же касается задачи с ограничениями, то, как легко видеть, рассмотренные выше методы ее сведения к задаче

без ограничений годятся не только для дифференцируемых, но и для произвольных непрерывных функций. Важно лишь уметь вычислять субградиенты для измененных целевых функций.

Для вычисления субградиентов применяются различные способы. Одним из них является сведение задачи вычисления субградиента функции, заданной сложным выражением, к вычислению субградиентов отдельных компонент этого выражения. Для подобного сведения особенно часто применяются следующие свойства выпуклых функций.

Свойство 1. Если функции $f_1(x)$, ..., $f_m(x)$ выпуклы, то выпуклой будет и любая их линейная комбинация $f(x) = \sum_{i=1}^{m} a_i f_i(x)$ с неотрицательными коэффициентами a_i $(a_i \geqslant 0,\ i = 1, \ldots, m)$, а субградиент $S(x)$ этой комбинации равен соответствующей линейной комбинации $\sum_{i=1}^{m} a_i S_i(x)$ субградиентов $S_i(x)$ функций $f_i(x)$.

Свойство 2. Если функции $f_1(x)$, ..., $f_m(x)$ выпуклы, то выпуклой будет и функция $f(x) = \max\limits_i f_i(x)$, и все субградиентные множества $S_i(x^0)$ функций $f_i(x)$, для которых $f_i(x^0) = f(x)$ входят в субградиентное множество $S(x^0)$ функции $f(x)$ в любой заданной точке $x = x^0$.

Свойство 2 имеет особо важное значение, поскольку на практике большинство случаев возникновения недифференцируемости происходит в результате применения операций $\max\limits_i$ (для задач минимизации) и $\min\limits_i$ (для задач максимизации) к дифференцируемым (часто даже к линейным) функциям $f_i(x)$. Пусть, например, в одномерном случае $f(x) = \max\limits_i f_i(x)$, где $f_1(x) = x$, $f_2(x) = 2 - x$. График функции $f(x)$ приведен на рис. 8.10. В точке $x = 1$ функция $f(x)$ недифференцируема. Так как $f_1(1) = f_2(1)$, то по свойству 2 субградиенты $S_1(x)$ и $S_2(x)$ функций $f_1(x)$ и $f_2(x)$ в точке $x = 1$ входят в субградиент $S(1)$ функции $f(x)$ в этой точке. Но субградиенты функций $f_1(x)$ и $f_2(x)$ в силу дифференцируемости этих функций равны их градиентам: $S_1(x) = \frac{d}{dx} f_1(x) = 1$, $S_2(x) = \frac{d}{dx} f_2(x) = -1$. По свойству 2 функции $S_1(1)$ и $S_2(1)$ входят в субградиент $S(1)$.

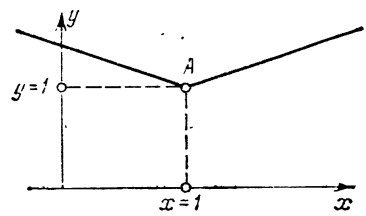

Рис. 8.10

Для выпуклых функций справедливо следующее свойство.

Свойство 3. Субградиент $S(x)$ выпуклой функции $f(x)$ в любой точке представляет собой выпуклое множество. Иными словами, если векторы $r_1(x)$ и $r_2(x)$ принадлежат $S(x)$, то векторы $\alpha r_1(x)+(1-\alpha)r_2(x)$ при $0 \leqslant \alpha \leqslant 1$ также принадлежат $S(x)$.

Используя это свойство в рассматриваемом примере, приходим к тому, что субградиент $S(1)$ функции $f(x)=\max(x,\ 2-x)$ в точке $x=1$ составляют числа (одномерные векторы) $\alpha+(1-\alpha)(-1)=2\alpha-1$ при $0 \leqslant \alpha \leqslant 1$.

Заметим, что все приведенные свойства (с заменой $\max$ на $\min$ в свойстве 2) имеют место не только для выпуклых, но и для вогнутых функций.

8.4.2. Методы растяжения пространства. Рассмотренный в предыдущем пункте простой субградиентный метод может оказаться весьма медленно сходящимся, если области уровня целевой функции сильно вытянуты в одном направлении, или, как часто в этом случае говорят, имеют вид узких и длинных оврагов. В двумерном случае пример такого оврага показан на рис. 8.11. Одно из возможных направлений обобщенного антиградиента в точке A минимизируемой выпуклой функции $f(x)$ показано вектором AE, нормальным к верхней линии уровня. В случае наличия оврага угол OAE близок к прямому, так что расстояние OE (где O — точка минимума) меньше расстояния OA лишь на весьма малую величину. При движении в подобных плохих направлениях приближение к точке минимума O будет совершаться весьма медленно.

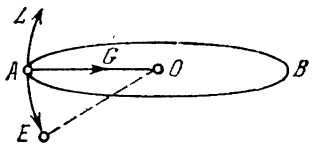

Рис. 8.11

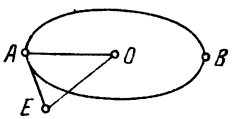

Рис. 8.12

Заметим, что овражная ситуация характерна для многих проблем негладкой оптимизации. Для улучшения сходимости метода в подобных случаях имеются методы *растяжения пространства*. Если в ситуации, изображенной на рис. 8.11, пространства растянуть в направлении «плохого» обобщенного антиградиента (или, что то же самое, обобщенного градиента AE), то придем к ситуации, изображенной на рис. 8.12. Овраг при этом растянется в ширину, а угол OAE в результате уменьшится, что обеспечит увеличение быстроты сходимости субградиентного спуска.

Недостаток описанного способа растяжения пространства проявляется в случае, когда выбранное направление обобщенного

антиградиента будет хорошим, например, совпадающим с вектором AG на рис. 8.11. Тогда растяжение лишь удлинит овраг и еще более ухудшит сходимость.

Гораздо лучшие результаты получаются при выполнении растяжения пространства в направлении разности обобщенных почти-градиентов в двух близких друг к другу точках или даже в одной точке при условии заведомой недифференцируемости целевой функции $f(x)$ в этой точке. На рис. 8.11 применение этого приема в точке A приведет к растяжению пространства в направлении вектора EL, т. е. в подходящем для растяжения направлении.

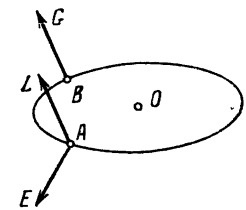

Рис. 8.13

Быстрое изменение направления градиента при смещении в близкую точку на практике свидетельствует, как правило, о том, что мы перескакиваем через дно оврага с одного его склона на другой. В этом случае разность градиентов покажет направление поперек оврага, что и обеспечит правильное направление растяжения пространства. Ситуация, описывающая подобный скачок градиента, изображена на рис. 8.13. Разность градиентов AE и BG задается вектором LE (векторы AL и BG равны между собой).

В алгоритмах, использующих растяжение пространства в направлении разности двух последовательных градиентов, шаговый множитель выбирается путем приближенного поиска минимума по направлению, при этом в случае неоднозначности вычисления обобщенного градиента в очередной точке выбирается тот, который образует с направлением движения острый или прямой угол.

Растяжение пространства сводится к замене переменных $x_1, \ldots, x_n$ в целевой функции $f(x)=f(x_1, \ldots, x_n)$ некоторыми их линейными комбинациями. Особенно просто выполняется растяжение в направлении одной из координатных осей. Переменная x_i, соответствующая этой оси, заменяется при этом на $k^{-1}x_i'$, где k — коэффициент растяжения, а остальные переменные остаются без изменения.

8.4.3. Методы с уточнением выбираемого направления. В последние годы большое распространение получили методы выпуклой оптимизации, использующие информацию о субградиентах функции в точках, близких к заданной, для уточнения направления спуска (подъема) из заданной точки. В большинстве случаев такое уточнение позволяет выбрать направление спуска (подъема), ведущее внутрь области уровня, на границе которой лежит рассматриваемая точка. Благодаря этому открывается возможность использования метода полношагового спуска (подъема), ускоряющего, вообще говоря, процесс приближения к экстремуму.

Заметим, однако, что подобные уточнения усложняют каждый шаг процесса оптимизации. Может случиться, что эти усложнения не будут скомпенсированы уменьшением числа шагов движения к экстремуму. Поэтому на практике желательны компромиссы, приводящие к достаточно хорошим направлениям за разумное время. Целью подобных компромиссов является обеспечение наибольшей скорости работы всего алгоритма оптимизации в целом.

8.4.4. Метод эллипсоидов. В случае задачи выпуклой оптимизации с ограничениями, как правило, удается так определить допустимую область M_0, что она оказывается ограниченной. Заключив область M_0 в сферу S_0, проведем какую-либо гиперплоскость P_0 через центр этой сферы.

В силу выпуклости (вогнутости) целевой функции $f(x)$ любой обобщенный градиент (антиградиент) в произвольной точке гиперплоскости P_0 внутри сферы S_0 будет указывать в ту полусферу, где находится максимум (минимум) функции $f(x)$ в случае ее вогнутости (выпуклости). Выбирая нужную (содержащую экстремум) полусферу M_0, описываем вокруг нее эллипсоид L_0 минимального объема.

Эллипсоид L_0 оказывается сплюснутым в направлении градиента $\nabla f(x_0)$, а его центр лежит на луче, проходящем через x_0 в направлении антиградиента. Произведя операцию растяжения пространства в направлении градиента, превращаем эллипсоид L_1 в сферу S_1.

Для сферы S_1 повторяем описанный процесс, получая эллипсоид L_1 и новую сферу S_2. Продолжая подобным образом, получим (с учетом изменения масштабов на каждом шаге) последовательность сфер $\{S_0, S_1, \ldots, S_i, \ldots\}$, уменьшающегося объема, который в пределе стремится к нулю. Так как искомая точка экстремума функции $f(x)$ находится внутри каждой из сфер S_i, то последовательность $\{S_0, S_1, \ldots\}$, стягивается к этой точке. Фактически (без учета изменения масштабов) точка экстремума аппроксимируется последовательностью эллипсоидов уменьшающегося объема.

При наличии множества точек экстремума указанный процесс приводит к аппроксимации одной из них. Разрезание множества экстремума гиперплоскостью может привести к случаю наличия в некоторых ее точках нулевого градиента. Это значит, что получена точка экстремума.

Метод эллипсоидов сходится со скоростью геометрической прогрессии, если скорость сходимости измерять по отклонению наилучшего достигнутого на данном шаге значения $f(x)$ от оптимального, при этом знаменатель геометрической прогрессии зависит только от размерности пространства n асимптотически:

$$q_n \approx 1 - \frac{1}{2n^2}.$$

При больших n оп близок к 1, что обуславливает практически медленную сходимость метода. Отметим, что метод эллипсоидов является частным случаем метода обобщенного градиентного спуска с растяжением пространства в направлении градиента. Благодаря законченности теории сходимости метода эллипсоидов, он был использован для построения и обоснования полиномиального алгоритма решения задачи линейного программирования с целыми коэффициентами.

8.5. Негладкая оптимизация

Описанные в § 8.4 методы оптимизации выпуклых (вогнутых) функций не предполагают гладкости этих функций. Отказ от свойства выпуклости (вогнутости) нарушает, вообще говоря, сходимость указанных методов. Однако, если начальное приближение находится достаточно близко к какому-либо экстремуму (локальному или абсолютному) произвольной непрерывной негладкой функции $f(x)$, методы § 8.4 (за исключением метода эллипсоидов) на практике обычно приводят к его нахождению с любой требуемой степенью точности. В этом (несколько условном) смысле можно говорить о применимости описанных методов к общим задачам локальной негладкой оптимизации. «Патологические» примеры, которые опровергают абсолютную их применимость, на практике встречаются достаточно редко, поэтому их можно игнорировать.

8.5.1. Метод координатного спуска (подъема). При оптимизации недифференцируемых функций в принципе можно вообще отказаться от любых обобщений понятия градиента, используя лишь шаги вдоль осей координат. С этой целью исследуют значения целевой функции $f(x_1, \ldots, x_n)$ на очередном шаге оптимизации с приращением $\pm\delta_i$ одной из координат и переходят от точки $(x_1, \ldots, x_n)$ к точке $(x_1, \ldots, x_{i-1}, x_i \pm \delta_i, x_{i+1}, \ldots, x_n)$, выбирая знак приращения таким образом, чтобы значение функции $f(x_1, \ldots, x_n)$ изменялось в нужном направлении (увеличивалось при максимизации и уменьшалось при минимизации). Величина δ_i выбирается обычно как в полношаговых методах, обеспечивая максимальное возможное изменение функции в выбранном направлении.

Сдвиги совершаются поочередно по всем осям координат, причем после сдвига по последней оси x_n снова возвращаются к первой оси x_1. Повторяя указанный процесс достаточно большое число раз, как правило, можно найти сколь угодно хорошее приближение некоторого экстремума заданной негладкой функции, предполагая, разумеется, что такой экстремум (обычно локальный) существует. Преимуществом метода является крайняя простота каждого шага, поскольку вопрос о направлении сдвига решается без какого-либо намека на дифференцирование или его обобщение.

8.5.2. Стохастическая оптимизация. Иногда оказывается полезным заменить сдвиги вдоль координатных осей сдвигами (с подходящим знаком) вдоль случайно выбираемого на каждом шаге направления. При достаточно общих предположениях в результате таких сдвигов с вероятностью единица за достаточно большое число шагов экстремальная точка может быть аппроксимирована с любой требуемой степенью точности. Подкупающим достоинством метода является (как и в предыдущем случае) простота каждого отдельного шага оптимизации. Однако число шагов по сравнению с методами, обеспечивающими поиск «хороших» направлений движения, у подобных простых методов, как правило, значительно больше. Поэтому на практике они могут оказаться (и во многих случаях действительно оказываются) менее эффективными, чем более сложные методы, использующие понятие субградиента и различные дополнительные приемы ускорения сходимости.

8.6. Линейное программирование

Одной из наиболее часто встречающихся задач выпуклой оптимизации является так называемая *задача линейного программирования*, в которой как целевая функция, так и все ограничения линейны. Для решения этой задачи могут быть, разумеется, применены общие приемы выпуклой оптимизации, описанные в § 8.4. Однако на практике для этого случая употребляют иные, более простые приемы, с одним из которых (так называемым симплекс-методом) мы познакомимся ниже.

Заметим, что встречающиеся на практике постановки задач линейного программирования предусматривают еще одно дополнительное ограничение, а именно условие неотрицательности значений всех переменных. Далее, как уже отмечалось выше, ограничения типа неравенств $p_i(x) \geqslant 0$ или $p_i(x) \leqslant 0$ введением дополнительных переменных с неотрицательными значениями превращаются в равенства $p_i(x) - z_i = 0$ или $p_i(x) + z_i = 0$. Кроме того, простым изменением знака целевой функции задача нахождения максимума сводится к задаче нахождения минимума, так что, не нарушая общности, в линейном программировании можно ограничиться лишь решением задачи минимизации.

Суммируя все сказанное, приходим к стандартной постановке задачи линейного программирования:

Найти минимум линейной функции $d + c_1x_1 + c_2x_2 + \ldots + c_nx_n$ при ограничениях $b_i + a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = 0$ $(i = 1, \ldots, m)$, $x_j \geqslant 0$ $(j = 1, \ldots, n)$.

Иными словами, речь идет о нахождении неотрицательного решения $x = (x_1, \ldots, x_n)$ заданной системы линейных алгебраических

уравнений $b_i + \sum_{j=1}^{n} a_{ij}x_j = 0$ $(i = 1, \ldots, m)$, обращающего в минимум значение заданной линейной функции $d + \sum_{j=1}^{n} c_j x_j$.

Задача линейного программирования может не иметь решения по следующим причинам: 1) система S уравнений $b_i + \sum_{j=1}^{n} a_{ij}x_j = 0$ $(i = 1, \ldots, m)$ несовместна, т. е. вообще не имеет решений; 2) система S не имеет ни одного неотрицательного решения; 3) на множестве M неотрицательных решений системы S целевая функция $f = d + \sum_{j=1}^{n} c_j x_j$ может принимать сколь угодно большие по абсолютной величине отрицательные значения, т. е. $\min\limits_{M} f = -\infty$.

Последняя причина исключает конечные решения, которые единственно и могут иметь практический интерес.

В случае, когда система S имеет единственное неотрицательное решение, оно, очевидно, и будет представлять собой решение задачи линейного программирования. На практике, однако, с таким случаем сталкиваться почти не приходится. Речь идет, таким образом, о случае, когда система S имеет множество M неотрицательных решений. Нетрудно показать, что это множество (если оно не состоит из одной точки) будет непременно бесконечным. В этом случае число независимых уравнений в системе S должно быть меньше числа n переменных, которое мы будем называть размерностью задачи линейного программирования. Это означает, что если в исходной системе S отброшены все зависимые уравнения, то $m \leqslant n$, причем при $m = n$ система S в силу независимости уравнений имеет единственное решение, так что нетривиальная постановка задачи линейного программирования предполагает строгое неравенство $m < n$. Заметим, что даже в таком случае неотрицательное решение может оказаться единственным. Нетрудно доказать, однако, что такое решение должно иметь $n - m$ нулевых координат.

8.6.1. Симплекс-метод. Один из наиболее распространенных методов решения задачи линейного программирования в стандартной постановке состоит в последовательном применении к линейным функциям так называемых *симплексных преобразований*, представляющих собой решение одного из ограничивающих уравнений $b_i + \sum_{j=1}^{n} a_{ij}x_j = 0$ относительно какого-либо неизвестного x_j и подстановки найденного таким образом значения x_j во все остальные ограничивающие уравнения и в целевую функцию

$f = d + \sum_{j=1}^{n} c_j x_j$. При этом предполагается дополнительно, что все ограничивающие уравнения были записаны в так называемой правильной форме и что правильность формы сохраняется и после выполнения преобразования.

Правильная форма записи ограничений предполагает наличие двух типов линейных уравнений: 0-уравнений

$$0 = B - (A_1 x_1 + A_2 x_2 + \ldots + A_n x_n), \quad \text{где} \quad B \geqslant 0,$$

и x-уравнений

$$x_j = D - (c_1 x_1 + \ldots + c_{j-1} x_{j-1} + c_{j+1} x_{j+1} + \ldots + c_n x_n), \text{ где } D \geqslant 0,$$

которые при фиксированном j будем называть x_j-уравнениями. Ясно, что, меняя в случае необходимости (при $b_i < 0$) знаки у всех членов уравнения $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0$, его всегда можно привести к виду 0-уравнения. К виду же x_j-уравнения можно привести лишь такое уравнение $b_i + \sum_{j=1}^{n} a_{ij} x_j = 0$, у которого знаки коэффициентов b_i и a_{ij} противоположны (точнее, для которого соблюдается условие $b_i a_{ij} \leqslant 0$). При приведении к x_j-виду 0-уравнения, записанного в правильной форме $0 = B - \sum_{j=1}^{n} A_j x_j$, необходимым условием такого приведения является соблюдение условия $BA_j \geqslant 0$. При этом из числа 0-уравнений, которые удовлетворяют данному условию, для симплекс-преобразования выбирается одно из тех 0-уравнений, у которых отношение B/A_j имеет наименьшее значение. Выполнение этого условия обеспечивает сохранение неотрицательности свободных членов 0-уравнений, в которые производится подстановка найденного выражения для x_j.

Проиллюстрируем выполнение симплексного преобразования на примере задачи линейного программирования: найти минимум функции $f = 6 - x - 3y$ в области S, заданной системой неравенств $x + y \leqslant 3$; $y \leqslant 2$, $x \geqslant 0$, $y \geqslant 0$. Прежде всего, вводя новые неотрицательные переменные u и z, обратим нетривиальные ограничивающие неравенства в равенства

$$x + y + u = 3, \; y + z = 2$$

и перепишем их в правильной форме (в виде 0-уравнений)

$$0 = 2 - (y + z), \quad 0 = 3 - (x + y + u).$$

Выбираем какую-либо переменную, для которой хотя бы одно из последних уравнений удовлетворяет первому условию ($BA_j \geqslant 0$). Такими в рассматриваемом случае будут все переменные.

Фиксируем одну из них, например, y. Первому условию ($BA_j \geqslant 0$) для переменной y удовлетворяют оба уравнения. По второму условию $\left(\min \frac{B}{A_j}\right)$ для симплексного y-преобразования выбирается первое уравнение, у которого $B/A_j = 2$, против $B/A_j = 3$ для второго уравнения. Разрешая первое уравнение относительно y и подставляя найденное выражение для y во второе уравнение, выполняем тем самым симплексное y-преобразование. В результате получаем систему уравнений:

$$y = 2 - (z), \quad 0 = 1 - (x - z + u).$$

Первая цель симплекс-метода состоит в том, чтобы путем последовательных симплексных преобразований исключить все 0-уравнения. В нашем случае для этой цели может подойти как x-преобразование, так и u-преобразование. Выполняя первое из них, приведем систему ограничивающих уравнений к виду

$$y = 2 - z, \quad x = 1 - (-z + u).$$

При достижении первой цели тривиальным образом решается задача нахождения одного из возможных неотрицательных решений рассматриваемой системы уравнений. С этой целью достаточно, как легко видеть, приписать нулевые значения всем переменным в правых частях x-уравнений и вычислить тем самым значения переменных, стоящих в их левых частях. Эти последние значения будут равны свободным членам правых частей x-уравнений, которые в силу правильности x-уравнений неотрицательны. В нашем случае указанный прием дает неотрицательное решение системы:

$$x = 1, \quad y = 3, \quad z = 0, \quad u = 0.$$

Если первая цель симплекс-метода оказывается недостижимой, то это свидетельствует об отсутствии у системы ограничивающих уравнений неотрицательных решений, а следовательно, и о неразрешимости поставленной задачи линейного программирования. Это имеет, например, место при единственном ограничении $0 = 1 - (-x - y - u)$, к которому симплекс-преобразования оказываются неприменимыми. При наличии у системы неотрицательных решений все 0-уравнения могут быть исключены с помощью симплексных преобразований, что и было сделано в рассматриваемом нами случае.

После достижения первой цели все ограничивающие уравнения оказываются записанными в виде, разрешенном относительно некоторого множества переменных x_j. Множество этих переменных называют *базисом*, к которому приведена система. На следующем шаге выражения для переменных базиса подставляются в целевую функцию f. Может оказаться, что после такой подстановки в

выражении для f все коэффициенты при оставшихся (небазисных неизвестных окажутся неотрицательными: $f = p + \sum_j q_j x_j$, где $q_j \geqslant 0$. Поскольку $x_j \geqslant 0$, то минимальное значение функции f (равное свободному члену p) получается при $x_j = 0$ для всех небазисных x_j. Так как все базисные переменные при этом также получают неотрицательные значения, то задача линейного программирования оказывается решенной.

Это как раз и имеет место в рассматриваемом нами случае. После выполнения подстановок $y = 2 - z$, $x = 1 - (-z + u)$ в целевую функцию $f = 6 - x - 3y$ она приобретает вид $f = -1 + 2z + u$ с положительными коэффициентами при небазисных переменных. Поэтому найденное решение $x = 1$, $y = 2$, $z = 0$, $u = 0$ системы ограничивающих уравнений решает задачу линейного программирования: минимум функции $f = 6 - x - 3y$ достигается (при заданных ограничениях) в точке $x = 1$, $y = 2$, а его значение равно -1.

При другой целевой функции, например функции $g = 4 - x + y$, подстановка найденных значений x и y приводит к результату

$$y = 5 - 2z + u.$$

Поскольку коэффициент при z отрицателен, приведенные выше рассуждения не имеют места. Для полного решения задачи линейного программирования необходимо продолжить симплексные преобразования с целью (которую назовем второй целью) устранить все отрицательные коэффициенты в выражении для целевой функции. Для этого в базис вводятся (вместо некоторых переменных, введенных ранее) те переменные, у которых коэффициенты в (преобразованной) целевой функции отрицательны. В данном случае такой переменной является z. Вводя ее в базис вместо переменной y (для чего y-уравнение решаем относительно z), получим

$$z = 2 - y, \quad x = 3 - (y + u).$$

Подставляя найденные выражения в целевую функцию g, приведем ее к виду

$$g = 1 + y + u.$$

Поскольку вторая цель достигнута (все коэффициенты при неизвестных в целевой функции неотрицательны), то, полагая $y = u = 0$, вычислим $z = 2$, $x = 3$, $g = 1$. Таким образом, при заданных ограничениях функция принимает минимальное значение, равное 1, в точке с координатами $x = 3$, $y = 0$.

После достижения второй цели приведенная целевая функция может оказаться не зависящей от некоторых небазисных переменных. Ясно, что таким переменным можно придавать произ-

вольные (разумеется, в соответствии с принятым согласованием, неотрицательные) значения, не меняя значения целевой функции. В этом случае задача линейного программирования имеет множество решений, задаваемых выражениями, в которых указанные переменные выступают как параметры.

Заменяя в рассматриваемом нами примере целевую функцию на функцию $h = 4 - x - y$ и приводя ограничения к уже полученному выше виду $y = 2 - z$, $x = 1 - (-z + u)$, подставим эти значения в целевую функцию. После подстановки она приведется к виду $h = 1 + u$. Минимум этой функции (равный 1) достигается при $u = 0$ и произвольном z. Поэтому решениями задачи будут все точки с координатами $x = 1 + z$, $y = 2 - z$ с условиями $z \geqslant 0$, $x = 1 + z \geqslant 0$, $y = 2 - z \geqslant 0$. Отсюда следует, что $0 \leqslant z \leqslant 2$. Итак, решение задачи дают все точки $(1 + z, 2 - z)$ при z, меняющемся в пределах $0 \leqslant z \leqslant 2$.

Заметим, что вторая цель даже при условии достижимости первой цели может оказаться недостижимой. Это происходит всякий раз, когда рассматриваемая целевая функция f не имеет минимума в заданной области $M \left(\min\limits_{M} f = -\infty\right)$. Например, при $f = 1 - x$ и ограничениях $x + y \geqslant 1$, $x \geqslant 0$, $y \geqslant 0$ после сведения неравенства к равенству $x + y - z = 1$ 0-уравнение будет иметь вид $0 = 1 - (x + y - z)$. Это уравнение может быть симплексно разрешено как относительно x, так и относительно y: $x = 1 - (y - z)$, $y = 1 - (x - z)$. В обоих случаях целевая функция не приводится к требуемому виду: в первом случае $f = y - 2$, во втором $f = 1 - x$.

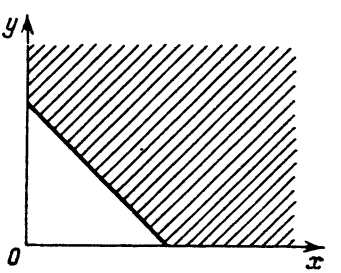

Рис. 8.14

Как видно из рис. 8.14, допустимая область (на рисунке заштрихована) бесконечна вправо и вверх, а целевая функция неограниченно убывает по мере роста x.

Имеется еще одна причина, которая может помешать решению задачи линейного программирования симплексным методом даже в том случае, когда такая задача имеет решение. Это так называемое *зацикливание*. Оно заключается в том, что после выполнения некоторого числа симплексных преобразований система ограничивающих уравнений возвращается к исходному виду, не исчерпав всех базисов и в том числе базиса, в котором достигается вторая цель (требуемый вид целевой функции).

Для устранения зацикливания в симплекс-метод вводятся специальные дополнения, которые рассматриваются в более подробных курсах. С такими дополнениями симплексный метод приво-

дит к решению задачи линейного программирования во всех случаях, когда такое решение существует.

Симплексный метод требует довольно сложных вычислений. При N неизвестных и M уравнениях число арифметических операций, необходимых для выполнения одного симплексного преобразования, выражается приближенно формулой $2MN$. Для достижения первой цели потребуется M таких преобразований, т. е. $2M^2N$ арифметических операций. Для достижения же второй цели (полного решения задачи линейного программирования) нужны будут, как правило, многократные изменения первоначального базиса, что еще более усложняет задачу. В исходной постановке при n неизвестных и k ограничивающих неравенствах (без дополнительных равенств) $N = n + k$, $M = k$, так что на одно симплексное преобразование тратится $2k(n + k)$ арифметических операций.

Из симплекс-метода непосредственно следует, что для стандартной формы задачи линейного программирования с M линейно независимыми ограничениями типа равенств всегда имеется решение, содержащее не более M ненулевых значений переменных.

8.6.2. Транспортная задача. В некоторых частных случаях решение задачи линейного программирования может быть значительно упрощено. Наиболее важным из таких случаев является так называемая *транспортная задача*, называемая также иногда *задачей о назначениях*. Как станет ясно из дальнейшего, переменные в этой задаче удобно снабжать парой индексов i, j. Если первый индекс пробегает m значений, а второй n значений, то общее число переменных x_{ij} (размерность задачи) равно, очевидно, mn. Сущность задачи состоит в минимизации целевой функции

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

при $m + n$ ограничениях типа равенств:

$$\sum_{j=1}^{n} x_{ij} = a_i, \quad \sum_{i=1}^{m} x_{ij} = b_j.$$

Обычно предполагается дополнительно соблюдение равенства

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j,$$

иначе задача не будет иметь решений. Общее число независимых ограничений — равенств уменьшается на единицу, равняясь тем самым $m + n - 1$.

Как следует из результатов предыдущего пункта, при этом условии имеется решение сформулированной задачи минимизации, содержащее не более $m + n - 1$ ненулевых значений переменных x_{ij}. Мы будем называть их *назначениями*.

Название транспортной рассматриваемая задача получила потому, что к ней сводится оптимизация плана перевозок грузов из m пунктов отправления с запасами $a_1, \ldots, a_m$ в n пунктов назначения с потребностями $b_1, \ldots, b_n$. Роль коэффициентов c_{ij} в целевой функции играют удельные стоимости, т. е. стоимости перевозки одной единицы груза из пункта i в пункт j. Задача состоит в минимизации общей стоимости перевозки грузов при условии, что грузы оказываются полностью вывезенными из всех пунктов отправления и потребности всех пунктов назначения оказываются полностью удовлетворенными. Имеются и другие интерпретации рассматриваемой (частной) задачи линейного программирования.

При решении транспортной задачи пользуются двумя $m \times n$-матрицами: *матрицей планов* $P = |p_{ij}|$ и *матрицей стоимостей* $C = |c_{ij}|$, подвергая их специальным преобразованиям. Для формирования матрицы P на начальном шаге делается ровно $m + n - 1$ назначений (некоторые из них могут быть нулевыми) так, чтобы, во-первых, удовлетворялись условия $\sum_{j=1}^{n} x_{ij} = a_i$ $(i = 1, \ldots, m)$, $\sum_{i=1}^{m} x_{ij} = b_j$ $(j = 1, \ldots, n)$, а во-вторых, чтобы выбранные (для назначений) элементы матрицы не образовывали ни одного *цикла*. Под циклом здесь понимается последовательность элементов матрицы $l_1 \cdot l_2, \ldots, l_{k+1} \cdot l_1$ $(k \geqslant 1)$, начинающаяся и кончающаяся одним и тем же элементом (любые ее два соседних элемента расположены либо в одном столбце, либо в одной строке). Пример такого цикла $(l_1 l_2 l_3 l_4 l_5 l_6 l_1)$ дает матрица $\begin{Vmatrix} l_1 & & l_2 \\ & l_4 & l_3 \\ l_6 & l_5 & \end{Vmatrix}$, в которой явно выделены лишь элементы, образующие цикл. При этом элементы l_1, l_3, l_5 составляют так называемый *нечетный полуцикл* цикла $(l_1 l_2 l_3 l_4 l_5 l_6 l_1)$ (с фиксированным началом l_1), а элементы l_2, l_4, l_6 — *четный полуцикл*.

Удовлетворяющий этим условиям начальный выбор может быть сделан последовательным (по строкам, а внутри строки — по столбцам) при помощи максимальных назначений, лимитируемых лишь имеющимися запасами и потребностями. Например, для $a_1 = 10$, $a_2 = 15$, $a_3 = 20$, $b_1 = 15$, $b_2 = 15$, $b_3 = 10$, $b_4 = 5$ начальные назначения задаются следующей таблицей:

	15	15	10	5
10	10			
15	5	10		
20		5	10	5

Строки этой таблицы озаглавлены запасами a_i в пунктах отправления, а столбцы — потребностями b_j в пунктах назначения. Первое назначение в первой строке исчерпывает весь запас в первом пункте отправления, второе назначение исчерпывает остающиеся потребности первого пункта назначения и т. д. Общее число назначений (равное 6) удовлетворяет условию $m+n-1=3+4-1=6$, циклы отсутствуют, так что начальную матрицу планов

$$P_0=\begin{Vmatrix}10 & 0 & 0 & 0\\ 5 & 10 & 0 & 0\\ 0 & 5 & 10 & 5\end{Vmatrix}$$

можно считать построенной.

Пусть матрица удельных стоимостей имеет вид

$$C=\begin{Vmatrix}2 & 3 & 4 & 4\\ 3 & 4 & 3 & 3\\ 4 & 3 & 2 & 4\end{Vmatrix}.$$

Жирным шрифтом мы выделим в ней элементы, соответствующие сделанным назначениям. Преобразования матрицы C состоят в прибавлении ко всем элементам одной и той же строки или одного и того же столбца некоторой константы (положительной или отрицательной). Целью таких преобразований является обращение в нуль всех выделенных элементов матрицы *).

Прибавляя к строкам матрицы C числа -3, -4, -3, а к столбцам — числа $+1$, 0, $+1$, -1, превратим ее в матрицу

$$C_1=\begin{Vmatrix}\mathbf{0} & \mathbf{0} & 2 & 0\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -2\\ 2 & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{Vmatrix}.$$

Если бы все элементы матрицы C_1 были неотрицательными, то, как можно показать, первоначальный выбор назначений давал бы оптимальное решение. Поскольку в нашем случае дело обстоит не так, применяется процедура исправления начальных назначений. Для исправления присоединяем к уже выделенным элементам матрицы C_1 ее наибольший по абсолютной величине отрицательный элемент (в данном случае элемент $c_{24}=2$). Если начальный выбор содержал точно $m+n-1$ элементов, то, как нетрудно доказать, при таком присоединении обязательно образуется цикл из выделенных элементов, в данном случае цикл ($c_{24}=-2$, $c_{34}=0$, $c_{32}=0$, $c_{22}=0$, c_{24}).

Рассмотрим теперь соответствующий цикл $p_0=(p_{24}, p_{34}, p_{32}, p_{22}, p_{24})$ в матрице планов P и превратим его в цикл $(p_{24}+p, p_{34}-p, p_{32}+p, p_{22}-p, p_{24}+p)$, где p — наименьшая величина в четном

*) В случае, когда выделенные элементы матрицы не составляют цикла, достижение этой цели оказывается всегда возможным.

полуцикле (p_{34}, p_{22}) цикла p_0 (величина p как бы сдвигается из каждого элемента четного полуцикла к следующему за ним элементу нечетного полуцикла). В нашем случае $p = p_{34} = 5$. Производя указанное преобразование цикла и исключая из числа выделенных элементов вновь возникший нулевой элемент $p_{34} - p$, придем к новой матрице планов P_1:

$$P_1 = \begin{Vmatrix} \mathbf{10} & 0 & 0 & 0 \\ 5 & 5 & 0 & \mathbf{5} \\ 0 & \mathbf{10} & \mathbf{10} & 0 \end{Vmatrix}.$$

Здесь, как и раньше, полужирным даны выделенные элементы. Выделяем те же элементы в матрице C_1:

$$C_1' = \begin{Vmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -2 \\ 2 & 0 & 0 & 0 \end{Vmatrix}.$$

Прибавляя к последнему столбцу число +2, приведем матрицу к виду

$$C_2 = \begin{Vmatrix} 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 \end{Vmatrix}$$

с нулевыми выделенными элементами, не содержащую отрицательных элементов. Это означает, что план P_1 оптимален.

Суммарная стоимость перевозок S_1 по этому плану равна $S_1 = 10 \cdot 2 + 5 \cdot 3 + 5 \cdot 4 + 5 \cdot 3 + 10 \cdot 3 + 10 \cdot 2 = 120$, в то время как суммарная стоимость перевозок S_0 по начальному плану P_0 была равна $S_0 = 10 \cdot 2 + 5 \cdot 3 + 10 \cdot 4 + 5 \cdot 3 + 10 \cdot 2 + 5 \cdot 4 = 130$.

При другой матрице стоимостей, например матрице

$$\begin{Vmatrix} 2 & 3 & 4 & 4 \\ 3 & 4 & 2 & 3 \\ 1 & 3 & 2 & 4 \end{Vmatrix},$$

для перехода к оптимальному плану

$$P = \begin{Vmatrix} 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 5 \\ 15 & 5 & 0 & 0 \end{Vmatrix}$$

потребовалось бы несколько шагов исправлений начального плана. При той же стоимости $S = 130$ начального плана оптимальный план дал бы значение стоимости, равное 95, т. е. значительно меньше по сравнению с начальным вариантом.

Как и всякая задача линейного программирования, транспортная задача может иногда иметь не один, а множество оптимальных планов. На такую ситуацию указывает наличие дополнительных (помимо выделенных) нулей в заключительной (преобразо-

ванной) матрице удельных стоимостей. Присоединяя эти нули точно так же, как мы поступали с отрицательными элементами, можно двигать вдоль возникающих циклов те или иные количества грузов, не нарушая при этом оптимальности плана.

Например, в заключительной матрице C_2 рассмотренного выше примера можно образовать цикл из нулевых элементов (c_{21}, c_{22}, c_{12}, c_{11}, c_{21}) и передвинуть в нем какую-нибудь величину груза, скажем c_{22}. В результате получим новый план

$$P_1' = \begin{Vmatrix} 7 & 3 & 0 & 0 \\ 8 & 2 & 0 & 5 \\ 0 & 10 & 10 & 0 \end{Vmatrix}$$

со значением общей стоимости перевозок $S_1 = 7 \cdot 2 + 3 \cdot 3 + 8 \cdot 3 + 2 \cdot 3 + 5 \cdot 3 + 10 \cdot 3 + 10 \cdot 2 = 120$. Таким образом, этот план, как и план P_1, является оптимальным.

С помощью подобных замен в пределах множества оптимальных планов можно добиться того, чтобы план, не теряя свойства оптимальности, приобрел некоторые дополнительные полезные свойства. Разумеется, для этого необходимо, чтобы транспортная задача обладала не одним, а многими решениями, что, конечно, на практике встречается далеко не всегда.

8.7. Целочисленное линейное программирование

В ряде случаев на практике приходится встречаться с такими задачами линейного программирования, в которых допустимы лишь целочисленные решения. Иными словами, все координаты экстремальной точки должны быть обязательно целыми числами. В случае транспортной задачи целочисленность решения при целых a_i ($i = 1, \ldots, m$) и b_j ($j = 1, \ldots, n$) обеспечивается автоматически изложенным в предыдущем пункте методом решения этой задачи.

В общем случае получение целочисленного решения требует применения специальных методов, включающих в себя как составную часть методы линейного программирования. Наиболее часто для этих целей применяются методы *отсечений* и метод *ветвей и границ*. Метод ветвей и границ, который можно рассматривать как разновидность метода последовательного анализа вариантов со специфическими правилами их развития и отбора, является универсальным методом, с помощью которого можно решать и существенно более общие задачи (как дискретные, так и непрерывные). Схема метода описывается в § 8.8.

Методы отсечений, образующие целую группу приемов, базируются на следующих идеях. Вначале решается обычная задача линейного программирования A_0 (с временно отброшенным требованием целочисленности решения). Если у точки $a^0 = (a_1, \ldots$

$\ldots, a_n)$, полученной в результате решения задачи A_0, все координаты — целые числа, то эта точка дает, очевидно, решение не только задачи A_0, но и исходной задачи целочисленного линейного программирования, которую мы условимся обозначать через A.

Если же хотя бы одна из координат точки a^0, например a_j, не является целым числом, то к исходной задаче целочисленного линейного программирования добавляется новое ограничение (линейное), построенное с использованием информации, имеющейся в таблице симплекс-метода, соответствующей шагу, на котором построено решение a^0, и обладающее следующими свойствами:

1) это ограничение является линейным неравенством и отсекает точку a^0, т. е. a^0 не удовлетворяет условиям задачи линейного программирования, которая получается из исходной путем добавления этого ограничения;

2) оно не отсекает ни одной целочисленной точки из множества допустимых решений задачи A_0;

3) из всех возможных способов построения такого ограничения выбирается в определенном смысле наилучший, т. е. такой, который удовлетворяет требованиям 1, 2 и отсекает от множества решений задачи A_0 «не слишком малую» часть.

В результате получается новая, отличающаяся от A_0 лишь добавленным отсечением, задача A_1, к которой вновь применима описанная процедура. Таким образом, методы отсечения строят и решают последовательность линейных задач A_0, A_1, ..., каждая из которых отличается от предыдущей лишь одним новым ограничением. Свойства 1—3, которыми обладают отсечения, гарантируют, что эта последовательность не оборвется раньше, чем будет построено целочисленное решение некоторой задачи (свойство 1); в процессе решения не будет отброшено оптимальное решение исходной целочисленной задачи (свойство 2); при некоторых минимальных и естественных требованиях к задаче процесс закончится после конечного числа итераций (свойство 3). После окончания процесса будет либо построено решение a^k задачи A_k с целочисленными компонентами, являющееся решением задачи A, либо будет показано, что задача A_k не имеет допустимых решений. В последнем случае и исходная задача неразрешима.

Иногда приходится иметь дело со смешанными задачами, в которых лишь часть переменных должны быть целочисленны. Методы отсечения разработаны и для таких задач. Процесс в этом случае заканчивается, как только построено решение линейной задачи, удовлетворяющее требованиям целочисленности лишь по этим переменным. Методы отсечения применимы, разумеется, и к задачам *булева линейного программирования*, в которых каждая переменная может принимать лишь одно из двух возможных значений: 0 или 1. В этом случае процесс решения задачи ничем не

отличается от общего случая, но при решении ряда классов таких задач методы отсечения оказываются достаточно эффективными.

В общем случае методы отсечения обладают двумя существенными недостатками. Число итераций в них может расти по экспоненте от размерности задачи. Кроме того, коэффициенты симплекс-таблицы с ростом числа итераций имеют тенденцию к увеличению их значений, что приводит обычно к вычислительным неприятностям, связанным с выходом значений этих коэффициентов за разрядную сетку ЭВМ, либо к накоплению погрешности вычислений, за счет чего может быть получено недопустимое решение задачи.

8.8. Общая задача дискретной оптимизации

В общем случае задача дискретной оптимизации не предполагает линейности как целевой функции $f(x)$, так и ограничений $p_i(x) \geqslant 0$ $(i = 1, \ldots, m)$. Не требуется также непременно целочисленность координат точек, среди которых находится решение. Важно лишь, чтобы они образовывали дискретное множество M. В большинстве случаев общее число точек допустимого множества M является конечным, так что в принципе задача оптимизации может быть решена простым переборо м, т. е. вычислением значения целевой функции во всех точках допустимой области и выбором среди них требуемых экстремальных (минимальных или максимальных) значений. Однако обычно на практике общее число N точек оказывается столь большим, что простой перебор физически невозможен даже при использовании самых мощных ЭВМ.

Основная задача в дискретной оптимизации сводится поэтому к разработке методов, направленных на максимально возможное сужение перебора.

В так называемых *прямых методах* дискретной оптимизации обычно используются те или иные аналоги рассматривавшихся выше (в непрерывном случае) градиентных методов. К ним относятся методы локальной оптимизации, в частности *метод вектора спада*. В общих чертах вычислительную схему этого метода для решения задачи минимизации действительной функции $f(x)$, определенной на некотором подпространстве M дискретного метрического пространства D, можно описать следующим образом.

Вначале выбираем некоторое начальное приближение $x^0 \in M$ и радиус r. В пространстве D рассматриваем окрестность $O(x^0, r)$ радиуса r с центром в точке x^0. Исследуя координаты вектора спада, характеризующего изменение значений целевой функции в точках множества $O(x^0, r) \cap M$ по сравнению с $f(x^0)$, определяем, является ли x^0 точкой локального минимума функции f. Если нет, то с помощью вектора спада в множестве $O(x^0, r) \cap M$ выбираем

точку x^1, для которой значение целевой функции меньше, чем $f(x^0)$. На следующей итерации снова повторяем описанную процедуру, исходя уже из x^1 как центра новой окрестности, и т. д. Процесс вычислений считаем оконченным, если получено некоторое локально оптимальное решение задачи. Схема метода допускает прерывание вычислений на любой итерации с выдачей в качестве результата последнего полученного допустимого решения задачи.

Отметим, что размерность вектора спада $\Delta=(\Delta_1, \ldots, \Delta_s)$ функции $f(x)$ относительно окрестности $O(x^k, r)$, где $k=0, 1, \ldots$, зависит как от метрики, выбранной в пространстве D, и радиуса r, так и от вида целевой функции. В большинстве случаев она значительно меньше числа точек множества $O(x^k, r)\cap M$. Вычисление всех s координат вектора спада, так или иначе связанное с перебором соответствующих точек из $O(x^k, r)\cap M$, производится лишь в том случае, когда x^k является точкой локального минимума функции $f(x)$. На любой промежуточной итерации метода вычисление координат вектора спада прекращается, как только определено некоторое направление уменьшения значения функции $f(x)$ по сравнению с $f(x^k)$.

Схема метода вектора спада является достаточно гибкой и позволяет варьировать такими параметрами, как метрика, величина радиуса, способ перебора точек в окрестностях при построении вектора спада. В рамках этой схемы разработан ряд алгоритмов решения различных классов задач дискретного программирования (в частности, задач полностью и частично целочисленного программирования, задач с булевыми переменными, задач комбинаторного типа). Основываясь на схеме метода вектора спада, разработан также метод направляющих окрестностей для решения целочисленных задач выпуклого программирования, который является аналогом метода возможных направлений, предназначенного для решения непрерывных задач.

Анализ результатов машинного эксперимента по решению задач дискретной оптимизации различных классов показывает, что метод вектора спада позволяет в большинстве случаев получать решения задач, близкие к оптимальному, за практически приемлемое время и при использовании сравнительно небольших объемов памяти ЭВМ. При этом для каждого из возможных алгоритмов метода (при фиксированных перечисленных выше параметрах) точность получаемого решения зависит от выбранного начального приближения и времени, выделенного для решения задачи.

В *релаксационных методах* прибегают к приему ослабления (релаксации) ограничений и замены целевой функции $f(x)$ ее минорантой $f'(x)$ (в задаче минимизации), т. е. функцией, удовлетворяющей условию $f'(x)\leqslant f(x)$ для всех x. Из допустимой

области M', полученной расширением исходной области M ($M' \supset M$) за счет ослабления ограничений, M' и $f'(x)$ выбираются так, чтобы, во-первых, релаксированная задача допускала сравнительно простые способы решения и, во-вторых, чтобы она как можно меньше отличалась от исходной по целевой функции и множеству допустимых решений:

$$\min_{x \in M'} f'(x) \leqslant \min_{x \in M'} f(x) \leqslant \min_{x \in M} f(x).$$

Аналогичные неравенства верны и для задачи максимизации:

$$\max_{x \in M'} f''(x) \geqslant \max_{x \in M'} f(x) \geqslant \max_{x \in M} f(x),$$

но здесь $f''(x)$ — мажоранта $f(x)$, т. е. $f''(x) \geqslant f(x)$ для всех $x \in M'$.

Простейшим примером релаксации является отбрасывание условий целочисленности в задаче целочисленного линейного программирования. В этом случае M — множество целочисленных точек из выпуклого многогранного множества M', а $f'(x) = f''(x) = f(x)$.

Методы релаксации используются обычно в качестве способа построения оценок в методе ветвей и границ.

8.8.1. Метод ветвей и границ (МВГ). Метод был разработан вначале для решения задач целочисленного линейного программирования (в этом случае он называется еще методом Лэнд и Дойг), а затем был распространен на более общие дискретные и многокритериальные задачи. Он позволяет строить точное или приближенное решение с заданной относительной погрешностью по целевой функции. Суть метода ветвей и границ заключается в том, что вместо исходной задачи оптимизации $A_0 = \{\min f_0(x), x \in M_0\}$ строится и решается последовательность релаксированных задач $\bar{A}_i = \{\min \bar{f}_i(x), x \in \bar{M}_i\}$, $i = 0, 1, 2, \ldots$ Вначале решается задача $\bar{A}_0$, являющаяся релаксацией A_0. Если ее решение x_0 является допустимым для M_0 и $\bar{f}_0(x_0) = f_0(x_0)$, то x_0 — оптимальное решение A_0, и процесс заканчивается. В противном случае $\bar{f}_0(x_0)$ является оценкой снизу для целевой функции на оптимальном решении задачи A_0, и алгоритм начинает процесс ветвления исходной задачи A_0 на несколько задач (непосредственных потомков).

Ветвление осуществляется таким образом, чтобы множество допустимых решений задач-потомков совпадало с множеством M_0, а решения релаксированных задач приближались к решениям задач-потомков. Например, в задаче целочисленного линейного программирования A_0 ветвление обычно осуществляется путем разбиения всех целочисленных точек на два множества M_1 и M_2 путем добавления к условиям задачи A_0 ограничения $x_j \leqslant a_j$ —

для задачи A_1 и $x_j \geqslant a_j + 1$ — для задачи A_2. Здесь a_j — целая часть значения переменной x_j в оптимальном решении x_0 задачи A_0; индекс j соответствует любой переменной, принимающей нецелое значение в точке x_0. Для каждой полученной при ветвлении задачи A_i строится и решается ее релаксация $\bar{A}_i$; кроме того, если задача A_i непротиворечива, то она добавляется к списку активных задач, из которого после ветвления удаляется задача, породившая их. В начале работы список активных задач включает в себя лишь исходную задачу A_0.

На рис. 8.15 показана геометрическая интерпретация шага процесса ветвления для задачи A_0 с двумя переменными. Множество решений задачи $\bar{A}_0$ ограничено многоугольником $OABCDE$. В результате ветвления по переменной x_1 из этого многоугольника удаляется внутренность заштрихованной полосы, содержащая и точку C — оптимальное решение задачи $\bar{A}_0$. Здесь $x_{\text{опт}}$ — решение задачи A_0, круглыми точками отмечено множество M_0, прямая $cx = cx_0$ — уравнение целевой функции, соответствующее оптимальному решению задачи $\bar{A}_0$. Уточнение оценок здесь достигается за счет «лучшей» релаксации задач A_1 и A_2 по сравнению с A_0.

Следующим этапом алгоритма является процедура выбора из списка активных задач. Обычно выбирается задача A_i с наименьшим значением $f_i(x_i)$, где x_i — оптимальное решение задачи $\bar{A}_i$. Из эвристических соображений именно задача $\bar{A}_i$, вероятнее всего, содержит оптимальное решение исходной задачи. Если $\bar{f}_i(x_i) = f_0(x_i)$ и $x_i \in M_0$, то понятно, что x_i — оптимальное решение A_0. В этом случае процесс решения заканчивается. В противном случае осуществляется процесс ветвления задачи A_i.

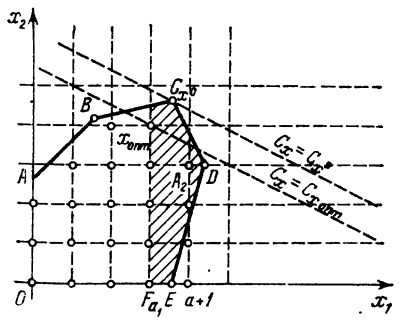

Рис. 8.15

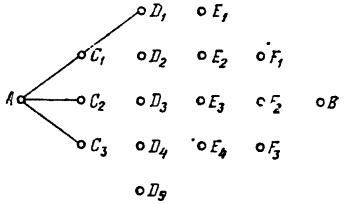

Рис. 8.16

В результате процесса строится *дерево задач* (рис. 8.16).

Некоторые вершины этого дерева породили своих потомков (A_0, A_1, A_3, A_5, A_7, A_8), и к моменту построения они уже исключены из списка активных задач. Другие вершины соответствуют недопустимым задачам и по этой причине также исключены из

этого списка. По оставшимся в списке вершинам возможен еще процесс ветвления.

Зачастую МВГ содержит еще одну процедуру, которая пытается построить допустимое решение $\tilde{x}_i$ для рассматриваемой задачи A_i и, в случае успеха, вычисляет значение $f_0(\tilde{x}_i)$. Если $\tilde{x}_i$ найдено, то величина $f_0(\tilde{x}_i)$ сравнивается с наилучшим значением целевой функции $f_0(x_i)$ и $f_0(x_R)$, полученным при анализе допустимых решений A_0 на просмотренных ранее вершинах дерева задач. В случае, если $f_0(\tilde{x}_i) < f_0(x_R)$, запоминается новое значение $x_R = \tilde{x}_i$. Величина $f_0(x_R)$ называется *рекордом*, а x_R — *рекордным решением* на текущем шаге. На начальном шаге $f_0(x_R)$ полагается равным достаточно большому числу, а x_R не определено. Значение рекорда позволяет сократить перебор в процессе решения, поскольку, если $\bar{f}_k(x_k) \geqslant f_0(x_R)$, то задача A_k не содержит решений, лучших, чем x_R, и ее можно исключить из списка активных задач. При использовании процедуры построения рекорда несколько изменяется правило окончания вычислений: алгоритм заканчивает работу, если список активных задач пуст. При этом рекордное решение, если оно определено, является оптимальным. В противном случае исходная задача не имеет допустимых решений.

Таким образом, МВГ есть процесс порождения ветвей (разбиение исходной и промежуточных задач на новые подзадачи), построения границ целевой функции на этих ветвях (путем решения релаксированных задач) и последующего анализа решения задач путем сравнения их с рекордным решением.

МВГ в общем случае также, как и все другие методы дискретной оптимизации, дает экспоненциальный объем вычислений от размерности задачи. Однако многочисленные экспериментальные исследования эффективности различных методов показывают, что при точном решении практических задач МВГ пока остается наиболее предпочтительным. Заметим, что в отличие от простой релаксации в методе ветвей и границ в общем случае меняются не только допустимые области, но и значения целевых функций.

8.9. Теория игр

К методам дискретной оптимизации и линейного программирования часто приходится обращаться при анализе и выборе решений в конфликтных ситуациях, т. е. при наличии сторон, преследующих различные (чаще всего — прямо противоположные) цели. Подобные ситуации принято называть *играми*. В зависимости от числа участников игры делятся на *парные* (с двумя участниками) и *множественные* (имеющие не менее трех участников). Результат, или исход игры, даже в том случае, когда он не имеет прямой количественной оценки, обычно харак-

теризуется некоторым числом, например, выигрыш $+1$, проигрыш -1, ничья 0. Наиболее полно разработана теория парных игр с нулевой суммой, т. е. таких игр, при которых одна сторона выигрывает то, что проигрывает другая. Развитие игры происходит в результате последовательного выполнения тех или иных *ходов*. Ход называется личным, если он предпринимается одним из игроков в результате сознательного анализа ситуаций. В противоположность личным, случайные ходы возникают не в результате сознательного решения, а в результате того или иного случайного процесса (например, сдача карт).

Стратегией игрока называется совокупность правил, определяющих поведение игрока от начала игры до ее завершения.

Задание стратегий (A, B) двух игроков в парной игре полностью определяет ее исход, т. е. выигрыш одного и проигрыш другого*). Игра называется *конечной*, если у каждого игрока имеется лишь конечное число стратегий. Результаты конечной парной игры с нулевой суммой можно задавать матрицей, строки и столбцы которой соответствуют различным стратегиям, а ее элементы — соответствующие выигрыши одной стороны (равные проигрышам другой). Эта матрица называется *платежной матрицей* или *матрицей игры*. При этом удобно проигрыш первой стороны рассматривать как ее отрицательный выигрыш, а выигрыш второй — как ее отрицательный проигрыш. Если первая сторона имеет m стратегий, а вторая n, то говорят, что мы имеем дело с игрой $m \times n$.

Рассмотрим игру $m \times n$ с матрицей

	B_1	B_2	...	B_n
A_1	a_{11}	a_{12}	...	a_{1n}
A_2	a_{21}	a_{22}	...	a_{2n}
...	...	...	...	...
A_m	a_{m1}	a_{m2}	...	a_{mn}

Если первый игрок применяет стратегию A_i, то второй будет стремиться к тому, чтобы выбором соответствующей стратегии B_j свести выигрыш первого игрока к минимуму. Величина этого минимума, которую мы обозначим через α_i, равна, очевидно, $\min\limits_j a_{ij}$.

С точки зрения первого игрока (при любых ответах противника) целесообразно стремиться найти такую стратегию, при которой α_i обращается в максимум. Этот максимум, который мы обозначим α, называется *нижней ценой* игры. Поскольку значение α

*) Если в игре имеются случайные ходы, то, строго говоря, определяются не выигрыш и проигрыш, а их математические ожидания.

вычисляется по формуле

$$\alpha = \max_i \min_j a_{ij},$$

то его называют также *максимином*. Ему соответствует *максиминная стратегия* (их может быть и несколько), придерживаясь которой первый игрок при любых стратегиях противника обеспечит себе выигрыш, не меньший α (в зависимости от знака α это может быть и проигрыш, который в этом случае окажется минимальным).

Аналогичным образом определяется минимальный проигрыш (который может быть в действительности и выигрышем) для второго игрока:

$$\beta = \min_j \max_i a_{ij}.$$

Величина β называется *верхней ценой* игры или *минимаксом*. Ей соответствуют *минимаксные стратегии* второго игрока.

Имеет место неравенство $\alpha \leqslant \beta$. При $\alpha = \beta$ решение получается в чистых стратегиях. Для их нахождения достаточно выделить в платежной матрице максиминные (т.е. равные α) элементы *). Любая пара строк и столбцов, на пересечении которых находится такой элемент, соответствует паре чистых оптимальных стратегий. При $\alpha < \beta$ первый игрок может существенно увеличить свой средний выигрыш по сравнению с α, если он будет пользоваться не чистой (одной-единственной) стратегией, а так называемой *смешанной стратегией*. Смешанная стратегия C состоит в том, что при повторении игры происходит случайный выбор стратегии из некоторого множества смешиваемых стратегий и для каждой смешиваемой стратегии указывается вероятность ее выбора.

Доказано, что для любой конечной парной игры с нулевой суммой существует пара оптимальных стратегий (вообще говоря, смешанных). Свойство оптимальности означает, что любое отступление одного из игроков от оптимальной стратегии (при условии, что второй игрок продолжает придерживаться своей оптимальной стратегии) при многократном повторении игры может только уменьшить его средний выигрыш (увеличить средний проигрыш).

Величина выигрыша (быть может, отрицательного) первого игрока при пользовании обеими сторонами парой оптимальных стратегий называется *ценой игры* и обозначается γ. Цена игры заключена между нижней и верхней ценами: $\alpha \leqslant \gamma \leqslant \beta$. Стратегии, которые смешиваются для получения оптимальной стратегии, будем называть *полезными*.

*) Поскольку $\alpha = \beta$, то максиминные элементы будут одновременно и минимаксными.

Решить игру — значит найти пару оптимальных стратегий и цену игры. Решение игры обладает одним важным свойством: если один из игроков использует свою оптимальную стратегию, а другой смешивает свои полезные стратегии в любых пропорциях (не обязательно оптимальных), то средний выигрыш продолжает оставаться равным цене игры.

При этом, правда, как и при любых отступлениях от оптимальной стратегии, соответствующее изменение стратегии противником может привести к увеличению его среднего выигрыша. Доказано, что у игры $m \times n$ существует такое оптимальное решение, что число полезных стратегий с каждой стороны не превосходит минимального из чисел m и n.

Решение всякой парной конечной игры с нулевой суммой может быть получено методами линейного программирования. Однако перед тем, как использовать эти методы, на практике обычно стараются провести предварительное упрощение задачи, исключая заведомо бесполезные стратегии.

Пусть, например, имеется игра 3×3 с матрицей

	B_1	B_2	B_3
A_1	1	0	2
A_2	0	2	1
A_3	−1	2	−2

Все элементы третьей строки матрицы не больше, чем соответствующие элементы второй строки. Поэтому, применяя вторую стратегию, первый игрок получит при любых стратегиях противника не меньший выигрыш, чем при применении третьей стратегии.

Значит, третья стратегия заведомо не может быть полезной, и ее следует исключить из рассмотрения. В оставшейся матрице 2×3 все элементы третьего столбца больше, чем соответствующие элементы первого столбца. Следовательно, если второй игрок применяет третью стратегию, то выигрыш его противника (т. е. его собственный проигрыш) больше, чем при применении первой стратегии. Поэтому при поиске решения стратегия B_3 может быть заведомо исключена из рассмотрения.

Таким образом, в результате исключения заведомо бесполезных стратегий игра 3×3 свелась к игре 2×2 с матрицей

	B_1	B_2
A_1	1	0
A_2	0	2

Следующим шагом в поиске решения находим нижнюю и верхнюю цену игры:

$$\alpha = \max_i \min_j a_{ij} = \max_i (0, 0) = 0,$$

$$\beta = \min_j \max_i a_{ij} = \min_j (1, 2) = 1.$$

Поскольку максимин (нулевой элемент) встречается как в первой, так и во второй строке, то обе стратегии первого игрока являются максиминными. Минимаксной же стратегией второго игрока является лишь стратегия B_1.

В нашем случае $\alpha \neq \beta$ и, следовательно, решение нужно искать среди смешанных стратегий.

Рассмотрим другой пример. Пусть игра 2×3 задана матрицей

	B_1	B_2	B_3
A_1	0	3	−1
A_2	3	2	1

Находим нижнюю и верхнюю цену игры:

$$\alpha = \max_i \min_j a_{ij} = \max_i (-1, 1) = 1,$$

$$\beta = \min_j \max_i a_{ij} = \min_j (3, 3, 1) = 1.$$

В этом случае $\alpha = \beta = 1$. Поскольку $\alpha \leqslant \gamma \leqslant \beta$, то цена игры также равна 1. Решение же игры задается парой чистых стратегий (A_2, B_3), ибо максимин (он же минимакс) расположен во второй строке и третьем столбце.

Как уже отмечалось, в области чистых стратегий решение может быть получено непосредственно. Если же решение нужно искать в области смешанных стратегий (при $\alpha \neq \beta$), то в общем случае $m \times n$-матрицы $\|a_{ij}\|$ применяется следующий прием. Считая все m стратегий первого игрока полезными, определим вероятности их применения в смешанной оптимальной стратегии (если какая-то стратегия в действительности бесполезна, то соответствующая вероятность обратится в нуль). Обозначим эти вероятности через $p_1, p_2, \ldots, p_m$, а цену игры (пока неизвестную) — через γ.

Поскольку при оптимальной стратегии средний выигрыш первого игрока не меньше γ при любой стратегии противника, то запишем n неравенств

$$\begin{aligned} p_1 a_{11} + p_2 a_{21} + \ldots + p_m a_{m1} &\geqslant \gamma, \\ p_1 a_{12} + p_2 a_{22} + \ldots + p_m a_{m2} &\geqslant \gamma, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ p_1 a_{1n} + p_2 a_{2n} + \ldots + p_m a_{mn} &\geqslant \gamma. \end{aligned} \tag{8.12}$$

Введем новые неизвестные

$$x_1 = p_1/\gamma,\ x_2 = p_2/\gamma,\ \ldots,\ x_m = p_m/\gamma.$$

Чтобы исключить возможность деления на нуль, можно всегда добиться, чтобы $\gamma > 0$. Для этой цели достаточно ко всем элементам матрицы $\|a_{ij}\|$ прибавить одно и то же положительное число и все ее элементы сделать положительными. Легко понять, что эта операция увеличивает цену игры на C, но не меняет искомых оптимальных стратегий.

Поскольку $p_1 + p_2 + \ldots + p_m = 1$, то $x_1 + x_2 + \ldots + x_m = 1/\gamma$. Разделив левые и правые части неравенств (8.12) на γ, получим новую систему неравенств:

$$\begin{aligned} &a_{11}x_1 + a_{21}x_2 + \ldots + a_{m1}x_m \geqslant 1,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ &a_{12}x_1 + a_{22}x_2 + \ldots + a_{m2}x_m \geqslant 1,\\ &a_{1n}x_1 + a_{2n}x_2 + \ldots + a_{mn}x_m \geqslant 1. \end{aligned} \tag{8.13}$$

При сделанных предположениях все x_i неотрицательны. Поскольку цель оптимальной стратегии — максимизация выигрыша, то при ее достижении линейная функция $f = x_1 + x_2 + \ldots + x_m = 1/\gamma$ должна обратиться в минимум. Итак, оптимальная стратегия первого игрока (т. е. набор вероятностей $p_i = \gamma x_i$) находится в результате минимизации функции $f = x_1 + x_2 + \ldots + x_m$ при неотрицательных x_i, удовлетворяющих системе неравенств (8.13). Это есть обычная задача линейного программирования, которая может быть решена методами, описанными в § 8.6. В результате ее решения находится не только оптимальная стратегия первого игрока, но и цена игры γ.

Зная цену игры, оптимальную стратегию второго игрока обычно можно находить уже без решения задачи линейного программирования*). Для этого выбирается $n-1$ полезных стратегий первого игрока (имея возможность менять местами игроков, можно считать, что $m \geqslant n$) и для каждой из них записывается средний выигрыш, который при этом должен быть обязательно равен цене игры γ. Например, если для первого игрока полезна стратегия A_i, то ей соответствует уравнение $a_{i1}q_1 + a_{i2}q_2 + \ldots + a_{in}q_n = \gamma$. Кроме того, имеется еще одно уравнение $q_1 + q_2 + \ldots + q_n = 1$. Всего имеет n уравнений для n величин $q_1, q_2, \ldots, q_n$, из которых

*) Чтобы найти оптимальную стратегию второго игрока методом линейного програмирования, достаточно поменять игроков местами и применить тот же прием, что и выше.

(при использовании их линейной независимости) можно найти значение этих величин.

Для игр 2×2 оптимальная стратегия обоих игроков может быть получена без решения задачи линейного программирования. При $\alpha=\beta$ это очевидно, поскольку решением являются чистые стратегии. При $\alpha\neq\beta$ обе стратегии первого игрока и обе стратегии второго обязательно будут полезными (иначе существовало бы решение в чистых стратегиях). Поэтому неравенства (8.12) превращаются в равенства

$$\begin{aligned} a_{11}p_1+a_{21}p_2&=\gamma,\\ a_{11}p_1+a_{22}p_2&=\gamma. \end{aligned} \tag{8.14}$$

Поскольку $p_1+p_2=1$, то имеется достаточное число уравнений для определения значений всех трех неизвестных p_1, p_2, γ.

Применим этот прием для решения первого из рассматривавшихся выше примеров с матрицей $\left\|\begin{matrix}1 & 0\\ 0 & 2\end{matrix}\right\|$. Уравнение (8.14) запишем в виде $p_1=\gamma$, $2p_2=\gamma$. Присоединяя к ним уравнение $p_1+p_2=1$, получаем решение: $p_2=1/3$; $p_1=2/3$; $\gamma=2/3$. Следовательно, при случайном чередовании первой и второй стратегии с относительными частотами 2/3 и 1/3 соответственно первому игроку обеспечен средний выигрыш в размере 2/3. Заметим, что этот вывод идет вразрез с интуицией, которая подсказывает более часто использовать вторую стратегию, поскольку она имеет многообещающую строку выигрышей: (0, 2) по сравнению с (1, 0) в первой стратегии.

Для нахождения оптимальной стратегии второго игрока достаточно (поскольку уже известна цена игры) составить уравнение для его среднего проигрыша при любой из двух полезных стратегий первого игрока, например, стратегии A_1. Величина среднего проигрыша (равная цене игры) представится при этом в виде $1\cdot q_1+0\cdot q_2$. Приравнивая ее цене игры $\gamma=2/3$, получаем $q_1=2/3$ и, следовательно, $q_2=1/3$. Итак, второй игрок должен смешивать стратегии B_1 и B_2 с частотами 2/3 и 1/3 соответственно.

Описанный метод простого решения игры может быть обобщен также на игры $2\times n$.

При больших платежных матрицах используются приближенные методы решения. Для получения приближения употребляется следующий итерационный метод. Игрок A выбирает любую из своих стратегий, например A_{i_1}. Игрок B выбирает в качестве ответа ту стратегию B_{j_1}, которая наиболее невыгодна для A при стратегии A_{i_1}. Игрок A отвечает на это лучшей (против стратегии B_{j_1}) стратегией A_{i_2}. Игрок B ищет стратегию B_{j_2}, наилучшую против смеси в равных пропорциях стратегий A_{i_1} и A_{i_2}. Стратегия A_{i_3} должна быть наилучшей против равновероятной

смеси стратегий B_{j1} и B_{j2}, стратегия B_{j3} — наилучшей против равновероятной смеси стратегий A_{i1}, A_{i2}, A_{i3} и т. д.

Если продолжать этот процесс достаточно долго, то частоты, с которыми встречаются различные стратегии, стремятся к вероятностям их применения в паре оптимальных стратегий, а средний выигрыш — к цене игры.

Заметим, что при $\alpha \approx \beta$ достаточно хорошими приближениями к решению будут чистые (минимаксные или максиминные) стратегии.

В настоящее время развитие теории игр вышло далеко за пределы рассмотренного нами простейшего случая парных игр с нулевой суммой. Однако ограниченный объем книги не позволяет дать представление о более сложных разделах современной теории игр, таких, как множественные коалиционные игры, дифференциальные игры и др.

8.10. Динамическое программирование

На практике очень часто приходится встречаться со случаями, когда целью оптимизации является установление наилучшей последовательности тех или иных работ (производственных операций, этапов строительства различного рода сооружений и т. п.). С подобной целью имеют дело при решении задач так называемого *динамического программирования.*

Одной из первых задач такого рода, привлекших внимание математиков, была так называемая *задача о коммивояжере* (бродячем торговце). Суть ее состоит в следующем: имеется $n+1$ городов A_0, A_1, ..., A_n $(n \geqslant 1)$ с заданными между ними расстояниями d_{ij} $(i, j = 0, 1, \ldots, n)$; требуется, отправляясь из A_0, выбрать такой маршрут передвижения A_0, A_{i1}, A_{i2}, ..., A_{in}, A_0, при котором коммивояжер, побывав в каждом городе по одному разу, вернулся бы в исходный пункт A_0, проделав минимально возможный суммарный путь.

Эта задача может быть в принципе решена методом простого перебора всех возможных маршрутов. Легко видеть, однако, что общее число таких маршрутов равно $n! = 1 \cdot 2 \cdot 3 \ldots (n-1) \cdot n$, а эта величина уже при n, измеряемом несколькими десятками, столь огромна, что практически исключает возможность прямых вычислений даже при использовании ЭВМ. Проблема, следовательно, состоит в том, чтобы резко сократить перебор, отбрасывая заранее заведомо непригодные множества вариантов. Методы решения этой проблемы как раз и составляют суть динамического программирования.

Рассмотрим основные принципы динамического программирования на примере задачи о коммивояжере. Пусть имеется всего

пять пунктов: A_0, A_1, A_2, A_3, A_4 со следующей таблицей 8.1. расстояний *):

Для определения кратчайшего пути коммивояжера будем развивать варианты его передвижения последовательно, пункт за пунктом. Начинаем с вариантов, состоящих из трех участков: $A_0A_{i1}A_{i2}A_{i3}$, группируя их по последнему пункту A_{i3}. Некоторые из полученных вариантов можно отбросить, не развивая их далее. Так, например, сравнивая варианты $A_0A_1A_2A_3$ и $A_0A_2A_1A_3$, приходим к тому, что для первого варианта суммарный путь равен $300+500+250=1050$, а для второго $250+500+350=1100$. Поскольку оба варианта относятся к одному и тому же множеству пунктов и оканчиваются в одном и том же пункте A_3, то в дальнейшем оба варианта могут развиваться одинаково. Ясно, что проигрыш 50 км, полученный для второго варианта, делает его бесперспективным в сравнении с первым при любом его дальнейшем развитии. Следовательно, второй вариант может быть отброшен. Тем самым отбрасываются и все получающиеся из него варианты дальнейшего движения (число которых при больших n может быть огромным).

Таблица 8.1

	A_0	A_1	A_2	A_3	A_4
A_0	0	300	250	200	400
A_1	300	0	500	350	600
A_2	250	500	0	250	200
A_3	200	350	250	0	250
A_4	400	600	200	250	0

Сведем результаты рассмотрения вариантов в табл. 8.2. Далее развиваются и сравниваются только перспективные варианты (в нашем случае их число равно 12). Развитие каждого из них

Таблица 8.2

Варианты	Расстояние, км	Перспективно или нет	Варианты	Расстояние, км	Перспективно или нет
$A_0A_2A_3A_1$	850	да	$A_0A_1A_2A_3$	1050	да
$A_0A_3A_2A_1$	950	нет	$A_0A_2A_1A_3$	1100	нет
$A_0A_2A_4A_1$	1050	да	$A_0A_1A_4A_3$	1150	да
$A_0A_4A_2A_1$	1100	нет	$A_0A_4A_1A_3$	1350	нет
$A_0A_3A_4A_1$	1050	нет	$A_0A_2A_4A_3$	700	да
$A_0A_4A_3A_1$	1000	да	$A_0A_4A_2A_3$	850	нет
$A_0A_1A_3A_2$	900	да	$A_0A_1A_2A_4$	1000	да
$A_0A_3A_1A_2$	1050	нет	$A_0A_2A_1A_4$	1350	нет
$A_0A_1A_4A_2$	1100	да	$A_0A_1A_3A_4$	900	да
$A_0A_4A_1A_2$	1500	нет	$A_0A_3A_1A_4$	1150	нет
$A_0A_3A_4A_2$	650	да	$A_0A_2A_3A_4$	750	нет
$A_0A_4A_3A_2$	900	нет	$A_0A_3A_2A_4$	650	да

*) Расстояния определяются для конкретных путей сообщения не обязательно по прямой.

однозначно (поскольку для каждого из них остается лишь один непосещенный пункт).

Результаты сравнения вариантов приведены в табл. 8.3. Из таблицы выбираем четыре перспективных варианта. Дальнейшее их развитие заключается в возвращении в исходный пункт A_0. Для этих четырех окончательных маршрутов вычисляем суммарные расстояния и записываем в табл. 8.4. Из этой таблицы видно, что существуют два оптимальных маршрута следования коммивояжера: $A_0A_1A_3A_4A_2A_0$ и $A_0A_2A_4A_3A_1A_0$, имеющие минимальную из всех возможных маршрутов длину, равную 1350 км.

Таблица 8.3

Варианты	Расстояние, км	Перспективно или нет	Варианты	Расстояние, км	Перспективно или нет
$A_0A_2A_3A_1A_4$	1450	нее	$A_0A_1A_2A_3A_4$	1300	нет
$A_0A_2A_4A_1A_3$	1400	нет	$A_0A_1A_4A_3A_2$	1400	нет
$A_0A_4A_3A_1A_2$	1500	нет	$A_0A_2A_1A_3A_1$	1050	да
$A_0A_1A_3A_2A_4$	1100	да	$A_0A_1A_2A_4A_3$	1250	да
$A_0A_1A_4A_2A_3$	1350	нет	$A_0A_1A_3A_4A_2$	1100	да
$A_0A_3A_4A_2A_1$	1150	нет	$A_0A_3A_2A_1A_1$	1250	нет

Таблица 8.4

Варианты	Расстояние, км	Варианты	Расстояние, км
$A_0A_1A_2A_3A_4A_0$	1500	$A_0A_1A_2A_4A_3A_0$	1450
$A_0A_2A_4A_3A_1A_0$	1350	$A_0A_1A_3A_4A_2A_0$	1350

Рассмотренный пример демонстрирует основной прием динамического программирования — нахождение *правил доминирования*, которые позволяют производить сравнение вариантов развития последовательностей и заблаговременное отсеивание бесперспективных вариантов. В ряде случаев в задачах динамического программирования удается получить столь сильные правила доминирования, что они определяют элементы оптимальной последовательности однозначно один за другим. В этом случае правила доминирования называют обычно *разрешающими правилами*.

Разрешающие правила (в случае их существования) обычно выводятся с помощью *принципа оптимальности Беллмана*. Суть принципа оптимальности состоит в следующем. Пусть критерий $\mathcal{F}$ (задаваемый формулой или алгоритмом), дающий числовую оценку качества варианта (последовательности) $A_n = a_{i_1}a_{i_2}\ldots a_{i_n}$, можно применять не только ко всей последовательности, но и к

любому ее начальному отрезку $A_k = a_{i1}a_{i2}\ldots a_{ik}$. Последовательность A_n, которой соответствует экстремальное значение критерия $\mathcal{F}$, называется *оптимальной*. Если любой начальный отрезок оптимальной последовательности также оптимален (в классе всех последовательностей, составленных из тех же элементов и, быть может, еще имеющий те же начало и конец, что и данный отрезок), то говорят, что для соответствующей задачи справедлив *принцип оптимальности*.

Принцип оптимальности, как нетрудно понять, всегда выполняется для аддитивных критериев $\mathcal{F}$, характеризующихся тем свойством, что для любой последовательности A, составленной из двух подпоследовательностей B и C $(A = BC)$,

$$\mathcal{F}(A) = \mathcal{F}(BC) = \mathcal{F}(B) + \mathcal{F}(C).$$

Рассмотрим один из многочисленных примеров задач динамического программирования, для которых выполняется принцип оптимальности. Предположим, что нам необходимо обработать последовательно n различных деталей $d_1, d_2, \ldots, d_n$. Время, требуемое для обработки детали d_i, обозначим через t_i, а стоимость хранения этой детали в единицу времени — через ρ_i $(i = 1, 2, \ldots, n)$. Предполагая, что обработку можно производить в любом порядке, и пренебрегая временем переналадки оборудования, нужно найти такой порядок обработки, при котором общая стоимость $\mathcal{F}$ хранения деталей (до начала их обработки) минимальна.

Справедливость принципа оптимальности в данном случае очевидна: если бы какой-то начальный отрезок оптимальной последовательности $d_{i_1}d_{i_2}\ldots d_{i_k}$ был не оптимальным, то перестановкой его элементов, приводящей его к оптимальному виду, можно было бы, очевидно, уменьшить значение критерия $\mathcal{F}$ для всей последовательности.

Пусть теперь $D = d_{i_1}d_{i_2}\ldots d_{i_k}$ — любой начальный отрезок оптимальной последовательности $(k \geqslant 2)$, а D' — последовательность, полученная из D перестановкой двух его последних элементов. Тогда по принципу оптимальности

$$\mathcal{F}(D) \leqslant \mathcal{F}(D'). \tag{8.15}$$

Обозначим через T_j сумму $t_{i_1} + t_{i_2} + \ldots + t_{i_j}$. Тогда

$$\mathcal{F}(D) = \rho_{i_2}T_1 + \rho_{i_3}T_2 + \ldots + \rho_{i_{k-1}}T_{k-2} + \rho_{i_k}\left(T_{k-2} + t_{i_{k-1}}\right),$$

$$\mathcal{F}(D') = \rho_{i_2}T_1 + \rho_{i_3}T_2 + \ldots + \rho_{i_k}T_{k-2} + \rho_{i_{k-1}}\left(T_{k-2} + t_{i_k}\right).$$

Подставляя эти значения в неравенство (8.15) и вычеркивая одинаковые члены в левой и правой частях, приходим к искомому разрешающему правилу

$$\rho_{i_k}t_{i_{k-1}} \leqslant \rho_{i_{k-1}}t_{i_k}.$$

или, что то же самое,

$$t_{i_{k-1}}/\rho_{i_{k-1}} \leqslant t_{i_k}/\rho_{i_k}.$$

Из этого правила следует, что оптимальная последовательность предполагает обработку деталей в порядке роста (точнее, неубывания) отношения времени обработки t_i к стоимости ρ_i ее хранения в единицу времени: первой должна обрабатываться деталь с минимальной величиной отношения t_i/ρ_i, следующей обрабатывается деталь с минимальным отношением t_j/ρ_j среди оставшихся деталей и т. п.

Любопытно, что при интуитивном подходе к решению данной задачи обычно предлагается обработка в порядке убывания величины удельной стоимости ρ_i хранения деталей, что, как видно из приведенного рассмотрения, не приводит, вообще говоря, к оптимальной последовательности.

Рассмотрим еще один пример задачи динамического программирования, возникающей при проектировании дорог, трубопроводов, линий электропередач и других протяженных объектов.

Пусть, например, нам требуется проложить кабель из пункта A в пункт B (см. рис. 8.16) так, чтобы он прошел через один из пунктов C_i, один из пунктов D_j и т. д. Между любой парой соседних пунктов (A и C_i, C_i и D_i и т. д.) возможен лишь один вариант прокладки (например, по прямой). Требуется определить вариант прокладки, имеющий наименьшую стоимость.

Как и в предыдущем случае, легко понять, что в данной задаче имеет место принцип оптимальности. Для первого отрезка AC_i вычисляем стоимости всех вариантов: AC_1, AC_2, AC_3. Каждый из полученных отрезков продолжаем во все D_j и для всех D_j фиксируем варианты траекторий AC_iD_j, имеющие минимальную стоимость (если искать не все, а лишь одну оптимальную траекторию, то таких вариантов будет пять). Далее, продолжаем каждую из этих траекторий во все E_k и оставляем для каждого E_k вариант с наименьшей стоимостью (таких вариантов будет четыре). Повторяем ту же процедуру с точками F_r и т. д., пока не достигнем точки B. В нашем случае это приведет к анализу $3+3\cdot 5+5\cdot 4+4\cdot 3+3\cdot 1=53$ траекторий (из которых только три — полные), тогда как общее число всех возможных полных траекторий равно, очевидно, $3\cdot 5\cdot 4\cdot 3=180$.

Если число точек в каждом сечении (C_i, D_j и т. д.) одинаково и равно m, а число сечений (не считая A и B) есть n, то при применении описанного приема анализируется $m+(n-1)m^2+m$ траекторий, а их общее число равно, как нетрудно видеть, m^n. При больших m и n перебор сокращается значительно и, что не менее важно, при развитии траекторий на каждом шаге необходимо хранить ограниченное число траекторий (не более $2m$), что

резко ограничивает требования к объему памяти при решении задачи на ЭВМ.

Необходимость иметь большие m и n вытекает из того, что задачи описанного класса возникают обычно в результате дискретной аппроксимации непрерывных задач: в исходной задаче допустимы любые криволинейные траектории, соединяющие точки A и B; мы же аппроксимируем эти кривые ломаными, проходящими через вершины достаточно густой (для обеспечения хорошей аппроксимации) сетки.

Число подлежащих рассмотрению траекторий может быть еще более уменьшено, если к правилам доминирования, вытекающим из принципа оптимальности, предъявляются дополнительные ограничения, вытекающие из конкретного смысла задачи (например, отсутствие резких изломов траектории, ее обязательное прохождение через заданные промежуточные пункты и т. п.).

Количество элементарных арифметических операций, которые необходимо выполнить при решении задач динамического программирования, определяется помимо количества просматриваемых вариантов также сложностью вычисления критерия $\mathcal{F}$ для каждого из оцениваемых вариантов. Эта сложность может сильно варьироваться.

8.11. Вариационные задачи

Обычные задачи оптимизации ставят своей целью нахождение значений конечного числа вещественных или целочисленных переменных, оптимизирующих ту или иную функцию этих переменных. В вариационных задачах место переменных занимают функции, а критерий оптимизации — некоторая вещественная функция, зависящая от этих функций и называемая *функционалом*. С точки зрения современной абстрактной математики, оперирующей переменными любой природы, между этими двумя классами задач нет никаких принципиальных отличий. Однако практические методы нахождения экстремумов обычных функций и функционалов сильно разнятся между собой.

В простейшем случае имеют дело с функцией $y(x)$ одного переменного и с функционалом вида $I = \int_{a_1}^{a_2} F(x, y, y')\,dx$, где a_1 и a_2 — константы, а $F(x, y, z)$ — заданная функция трех вещественных переменных. При этом предполагается, что как сама функция $y = y(x)$, так и ее производная $y' = \frac{d}{dx}\,y(x)$ непрерывны на отрезке $a_1 \leqslant x \leqslant a_2$.

В классическом вариационном исчислении доказывается, что экстремум (минимум или максимум) функционала I достигается

на функциях $y(x)$, удовлетворяющих дифференциальному уравнению, называемому *уравнением Эйлера*:

$$F_y + F_{xy'} - F_{yy'} \cdot y' - F_{y'y'} y'' = 0.$$

Здесь через F_y и F_{uv} ($u = x$, y или y', $v = y'$) обозначены соответственно первая и вторые частные производные функции $F(x, y, y')$ по переменной y и по переменным u, v.

Для иллюстрации использования уравнения Эйлера рассмотрим простой пример функционала $I = \int_{a_1}^{a_2} \sqrt{1 + (y')^2}\, dx$, задающего длину кривой $y = y(x)$ на отрезке $[a_1, a_2]$. В уравнении Эйлера при этом все члены, кроме последнего, обращаются в нуль. Из оставшегося соотношения $-F_{y'y'} y'' = 0$ выводим прежде всего тривиальное решение $y'' = 0$, т. е. уравнение прямой линии, которая, как известно, дает минимум функционалу I. Поскольку столь же очевидно, что других минимумов, а также и максимумов этого функционала не существует, можно не рассматривать другого решения полученного уравнения: $F_{y'y'} = 0$. Более подробный анализ показывает, что это решение (сводящееся к решению уравнения $y'y'' = \text{const}$) не дает, вообще говоря, экстремального значения функционалу I. И это в порядке вещей, поскольку уравнение Эйлера дает лишь необходимое, но не обязательно достаточное условие экстремума заданного функционала.

Заметим, что общее решение дифференциального уравнения $y'' = 0$, имеющее вид $y = Ax + B$, содержит две произвольные постоянные A и B. Их значение может быть найдено из краевых условий, определяющих начальную и конечную точки кривой (в данном случае прямой), дающей решение задачи.

Возможны постановки вариационных задач и с более сильными краевыми условиями, включающими в себя, например, наряду со значениями искомой функции $b_1 = f(a_1)$ и $b_2 = f(a_2)$ на концах также значения ее производной $a_1 = f'(a_1)$, $c_2 = f'(a_2)$ в этих точках. Задачи на условный экстремум могут связывать искомые функции теми или иными дополнительными условиями. Вариационные задачи естественным образом обобщаются на функции многих переменных. Мы не будем, однако, вдаваться в эти постановки.

Отметим лишь, что при приближенном решении вариационных задач их можно в принципе сводить к задачам оптимизации обычных функций. Например, в рассмотренной выше простейшей постановке задачу нахождения неизвестной функции $y = f(x)$, оптимизирующей значение функционала $I = \int_{a_1}^{a_2} F(x_1, y_1, y')\, dx_1$ мож-

но заменить нахождением значений $y_1, \ldots, y_n$ этой функции в некоторых точках $x_1, \ldots, x_n$, достаточно плотно заполняющих отрезок $[a_1, a_2]$. При вычислении I следует заменить интеграл его приближенным значением (конечной суммой), выраженным через значения подынтегральной функции в точках $x_1, \ldots, x_n$, а значения производной y' в этих точках — теми или иными их приближениями, например $f'(x_i) \approx (f(x_{i+1}) - f(x_i))/(x_{i+1} - x_i)$.

С одним из возможных приближенных методов при решении вариационных задач с ограничениями — методом последовательного анализа вариантов — мы уже познакомились в предыдущем пункте. В следующей главе рассматривается принадлежащий Л. С. Понтрягину аналитический метод решения вариационных задач с ограничениями, встречающихся при оптимизации управления в динамических системах.

8.12. Системная оптимизация

Рассмотренные до сих пор постановки оптимизационных задач были *однокритериальными*, т. е. использовали одну-единственную целевую функцию (функционал). В практике проектирования больших систем и управления такими системами, как правило, используется много критериев. В ряде случаев их удается тем или иным способом свести к одному критерию и тем самым вернуться к уже исследованному случаю однокритериальной оптимизации. Простейший способ такого сведения заключается в так называемом *взвешивании* критериев. Если $f_1(x), \ldots, f_n(x)$ — целевые функции, выражающие значения используемых критериев, то для каждой из них, сообразуясь с относительной важностью критериев, выбирается положительный весовой коэффициент λ_i. Операция взвешивания критериев (целевых функций) $f_1(x), \ldots, f_n(x)$ состоит в замене их единственным критерием (целевой функцией) $f(x) = \lambda_1 f_1(x) + \ldots + \lambda_n f_n(x)$.

Однако для многих задач, связанных с большими системами, подобное сведение оказывается практически невозможным, так что в процессе оптимизации приходится иметь дело с *векторной* (*многокритериальной*) целевой функцией. При этом допустимая область M может меняться в процессе оптимизации. Более того, в ее целенаправленном изменении как раз и заключается основная содержательная сущность процесса оптимизации для подобного класса задач.

Поскольку законы возможных изменений допустимой области M задаются обычно системой моделей, то описываемый подход к оптимизационным задачам естественно называть *системным*. Заметим, что при системном подходе изменения ограничений, задающих допустимую область в пространстве тех или иных параметров, происходят, как правило, в результате последователь-

ности решений, выбираемых из дискретного множества возможных решений, причем само это множество в начале процесса оптимизации обычно бывает не полностью заданным и пополняется в процессе диалога с людьми (плановиками или конструкторами), владеющими приемами выработки новых решений, не до конца формализованными.

Приведем одну из характерных формализованных постановок задачи системной оптимизации. Для того чтобы лучше уяснить идею, проиллюстрировав ее графически, рассмотрим двукритериальный случай. Предположим также, что выбором значений этих критериев однозначно определяется соответствующее решение. Иными словами, искомое решение ищется непосредственно в пространстве K критериев оптимизации, которые мы обозначим x_1 и x_2 (рис. 8.17).

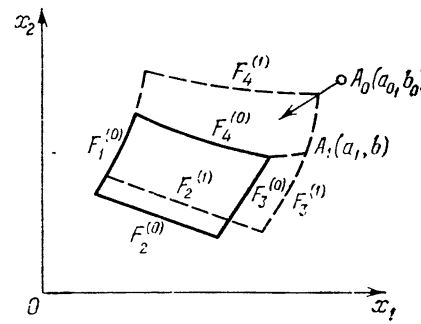

Рис. 8.17

Процесс решения начинается с того, что в заданном пространстве K выбирается некоторая точка A_0 с координатами a_0, b_0— желательное решение задачи. Далее, строятся начальные ограничения $F_1^{(0)}(x_1, x_2) \geqslant 0, \ldots, F_n^{(0)}(x_1, x_2) \geqslant 0$, задающие начальную допустимую область P_0. Прямой проверкой устанавливается, принадлежит ли точка A_0 области P_0. В первом случае в принципе может быть применена обычная (классическая) процедура оптимизации либо по одному из критериев x_1, x_2, либо по той или иной их комбинации.

Однако при системном подходе применяется обычно совершенно другой прием, а именно: в соответствии с моделью M высшего уровня, управляющей выбором критериев, точка A_0 выводится из пределов допустимой области P_0, как это и показано на рис. 8.17.

После этого выделяются те ограничения, которые не выполняются в точке A_0 (в рассматриваемом случае ими будут $F_3^{(0)}$ и $F_4^{(0)}$). Обращаясь к моделям M_3 и M_4, формирующим эти ограничения, в диалоговом режиме опробываются те или иные решения, изменяющие соответствующие ограничения в нужном направлении (если такое изменение оказывается возможным). Нужным при этом считается то направление, которое уменьшает абсолютную величину отрицательных невязок $F_i^{(0)}(a_0, b_0)$ (в рассматриваемом случае $F_3^{(0)}(a_0, b_0)$ и $F_4^{(0)}(a_0, b_0)$).

Следует иметь в виду, что во многих случаях ограничения F_i оказываются взаимосвязанными, так что изменение одного из них

приводит к изменению определенной части других ограничений. Управление выбором решений для изменения ограничений определяется при этом минимизацией некоторой функции штрафа $g_0(a_0, b_0)$. В качестве такой функции выбирается обычно максимальная абсолютная величина отрицательных невязок $\lambda_i F_i^{(0)}(a_0, b_0)$ (где λ_i — некоторые положительные весовые коэффициенты). Если таких невязок нет, то по определению $g_0(a_0, b_0) = 0$.

В результате управления появляется ряд решений $R_1, \ldots, R_m$, приводящих к уменьшению значения функции штрафа, которое после m-го решения обозначим $g_m(a_0, b_0)$. Каждое из принятых решений, изменяя ограничения, приводит к соответствующему изменению допустимой области. На рис. 8.17 показаны два таких изменения: первое изменяет ограничения $F_3^{(0)}, F_2^{(0)}$, заменяя их соответственно ограничениями $F_3^{(1)}$ и $F_2^{(1)}$; второе затрагивает лишь одно ограничение $F_4^{(0)}$, заменяя его ограничением $F_4^{(1)}$. Получающаяся допустимая область P_2 ограничена линиями $F_1^{(0)}$, $F_2^{(1)}, F_3^{(1)}, F_4^{(1)}$, а соответствующее значение функции штрафа равно $g_2(a_0, b_0)$. Заметим, что заблаговременный выбор конечной допустимой области невозможен ввиду того, что последовательность областей $P_0, P_1, \ldots$ может не быть упорядочена по вложению. Кроме того, огромная трудоемкость формирования новых ограничений не позволяет выполнить эту работу заблаговременно, поскольку при этом потребовалось бы сделать много лишней работы по изменению несущественных ограничений.

Если (как на рис. 8.17) $g_2(a_0, b_0) \neq 0$, а решений, приводящих к дальнейшему уменьшению значения функции штрафа, нет, то происходит возвращение к высшей модели M, управляющей выбором желательного решения $A(a, b)$. Путем ряда последовательных решений $D_1, D_2, \ldots, D_k$ на изменение начального решения задачи $A_0(a_0, b_0)$ оно заменяется на $A_1(a_1, b_1), \ldots, A_k(a_k, b_k)$, пока очередная точка $A_k(a_k, b_k)$ не окажется в допустимой области (на рисунке $k = 1$). Решения на изменения выбираются из допустимого множества решений с целью минимизации функции штрафа. Этот процесс близок к классическому процессу оптимизации, за исключением того обстоятельства, что шаги выбираются не произвольно, а в соответствии с допустимыми (моделью M) решениями.

Наконец, после попадания точки A_k в заключительную допустимую область P_m может быть применена дополнительная процедура оптимизации по каким-либо комбинациям критериев x_1 и x_2 в пределах этой допустимой области. Такая процедура отличается от классической лишь тем, что выбор шагов оптимизации не произволен, а управляется моделью M высшего уровня. Если дальнейшему улучшению избранного критерия мешают некото-

рые ограничения, поддающиеся дальнейшим изменениям в нужную сторону, то процесс оптимизации может быть продолжен за счет включения в него последовательных изменений.

Заметим, что однозначное определение решения задачи выбором значений всех критериев оптимизации встречается не столь редко, как это может показаться на первый взгляд. Оно имеет, например, место для планово-экономических задач, где критерием (векторным) является чистый выпуск продукции разных видов, а решением задачи — полный выпуск. В случае, когда такая однозначность отсутствует, пространство, в котором ищется решение, помимо координат, соответствующих критериям оптимизации, может иметь и другие координаты. Описанный выше процесс оптимизации при этом усложняется за счет того, что точки $A_i(a_i, b_i)$ заменяются гиперплоскостями. Усложняется и определение функции штрафа: в качестве нее может быть взято, например, расстояние от выбранной гиперплоскости до очередной допустимой области в пространстве с заданными сжатиями (растяжениями) вдоль осей, соответствующих критериям оптимизации.

В самом общем случае вместо гиперплоскостей могут фигурировать точечные множества произвольного вида. Возможны постановки, при которых на этих множествах значения критериев определены неоднозначно, а для различения более или менее предпочтительных решений на множествах задаются (моделью высшего уровня M) соответствующие весовые функции. Однако важной чертой системной оптимизации, сохраняющейся при всех подходах, помимо многокритериальности и возможности изменения допустимой области является взаимодействие моделей различных уровней. В случае планово-экономических задач решения в этих моделях проводятся управляющими различных уровней, а в случае проектно-конструкторских задач — проектантами.

Глава IX

АВТОМАТИЗАЦИЯ ИЗМЕРЕНИЙ И УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

9.1. Технология автоматизации измерений

Задача автоматизации измерений включает в себя ряд этапов: съем показаний с датчика (измерительного прибора), превращение их в дискретную (цифровую) форму, ввод в спецпроцессор или универсальную ЭВМ и, наконец, обработка результатов измерений, подразделяющаяся на первичную и вторичную обработку. Исторически автоматизация измерений началась с последнего этапа, т. е. с применения ЭВМ для вторичной (особо сложной) обработки результатов измерений. Примером может служить восстановление формы рудного тела по результатам геофизических измерений.

Первичная обработка измерений на ЭВМ при ручном вводе данных оказывается весьма малоэффективной, поскольку потери времени на ввод при относительно малом числе операций по их обработке обычно не компенсируются экономией, получаемой за счет атоматизации обработки. Поэтому центральной проблемой комплексной автоматизации процесса измерений является автоматизация ввода исходных данных. Решение этой задачи производится тремя основными способами. Во-первых, многие измерительные приборы (например, электрокардиографы) еще до начала эпохи автоматизации измерений снабжались средствами фиксации измеряемых величин в виде одной или нескольких кривых на тот или иной носитель (чаще всего — бумажный).

Поскольку такой способ фиксации информации существует и будет существовать в течение определенного периода в процессе перехода к безбумажной технологии, были приняты меры для автоматизации ввода этой информации в ЭВМ. Эта цель достигается с помощью специальных устройств графического ввода, которые в автоматическом или иногда в полуавтоматическом режиме измеряют координаты точек на кривых, изображенных на бумажных носителях (или на фотопленке), и осуществляют их ввод в ЭВМ. Полуавтоматический режим заключается в ручном сканировании нужной кривой (указании нужных точек) и авто-

матическом выполнении собственно измерений. Необходимость в таком режиме возникает в том случае, когда на носителе фиксируется большое число кривых, из которых для измерения нужно выделить лишь одну или несколько кривых, интересующих исследователя. С подобным положением приходится встречаться, например, при измерениях треков частиц, фиксируемых в результате экспериментов на ускорителях элементарных частиц (в пузырьковых камерах). Другой случай — ввод сложных контуров (особенно в условиях невысокой контрастности изображений), с чем достаточно часто приходится встречаться в биологических исследованиях, изучении аэрофотоснимков и т. п.

Второй способ автоматизации ввода — создание отдельных измерительных устройств и измерительных комплексов, регистрирующих измеряемые величины в цифровом виде на машинно-ориентированных носителях (чаще всего — на магнитных лентах). После установки таких носителей на вводно-выводные устройства общего назначения или устройства внешней памяти универсальных ЭВМ может осуществляться непосредственный ввод в ЭВМ записанной на них информации. В настоящее время построены и широко используются на практике измерительные комплексы для испытаний сложных объектов (самолетов, кораблей и др.), способные фиксировать на машинных носителях информацию от многих сотен датчиков. В состав таких комплексов вводятся *устройства связи с объектом* (УСО), обеспечивающие перевод измеряемых параметров в цифровую форму и передачу этих данных на запись в той или иной (жестко определенной или оперативно регулируемой) последовательности.

Третий способ заключается в непосредственном подключении (через УСО) измерительных приборов (датчиков) к специализированным или универсальным ЭВМ. При этом зачастую используются комбинируемые тем или иным способом модули Камака. В состав таких модулей помимо аналого-цифровых преобразователей могут включаться счетчики и регистры для фиксации цифровых сигналов, коммутаторы, устройства управления отображением информации на дисплеях, устройства для первичной обработки данных и др.

В настоящее время аналоговые измерительные приборы все в большей степени вытесняются цифровыми приборами, включающими в свой состав необходимые аналого-цифровые преобразователи. Развитие микропроцессорной техники привело к появлению измерительных приборов со встроенными устройствами первичной обработки (а иногда и вторичной). В случае необходимости более сложной обработки (в частности, согласованной обработки сигналов, поступающих от различных измерительных устройств) подобные приборы допускают относительно простое подсоединение к универсальным или специализированным ЭВМ.

9.2. Первичная обработка результатов измерений

Задачи первичной обработки результатов измерений определяются целью получения возможно более точного значения некоторой постоянной величины. В более сложных случаях целью может служить установление функциональной зависимости измеряемой величины от времени, двух измеряемых величин друг от друга и т. п.

9.2.1. Простейшая задача усреднения. Пусть в результате некоторого количества n измерений постоянной величины a получены значения $a_1, a_2, \ldots, a_n$. Ошибка измерения $\rho_i = a_i - a$ представляет собой случайную величину, закон распределения которой, как правило, симметричен относительно нулевой точки. Иными словами, положительные и отрицательные значения ошибки каждого единичного измерения оказываются равновероятными. В этом случае при предположении о равной точности всех произведенных измерений наилучшей оценкой $\bar{a}$ для измеряемой величины a оказывается среднее арифметическое результатов отдельных измерений:

$$\bar{a} = \frac{1}{n}(a_1 + a_2 + \ldots + a_n). \tag{9.1}$$

Оценку для точности приближения получаемого таким образом среднего значения $\bar{a}$ к истинному значению a измеряемой величины обычно дает вычисление так называемой *дисперсии* σ^2 измерений, полученных по формуле

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^2. \tag{9.2}$$

Квадратный корень из дисперсии дает так называемую *среднюю квадратичную ошибку* (отклонение) σ для данного ряда измерений:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^2}. \tag{9.3}$$

Заметим, что в случае, когда вместо оценки $\bar{a}$ истинного значения a измеряемой величины берется само это значение, при вычислении дисперсии и среднего квадратичного отклонения вместо множителя $1/(n-1)$ нужно брать множитель $1/n$. Причина, обуславливающая выбор большего по величине множителя $(1/(n-1) > 1/n)$, состоит в том, что среднее квадратичное отклонение ряда измерений от истинного значения измеряемой величины a будет, вообще говоря, больше, чем истинное (с множителем $1/n$) среднее квадратичное отклонение этого ряда от его среднего

значения $\bar{a}$. В математической статистике показывается, что использование множителя $1/(n-1)$ в формуле (9.2) дает наилучшую (так называемую *несмещенную*) оценку истинного значения дисперсии

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(a_i - a)^2.$$

9.2.2. Выделение сигнала на уровне помех. На практике нередко приходится встречаться со случаем, когда требуется выделить периодически повторяющийся сигнал x на фоне сильных помех. Подобная ситуация имеет, например, место в сверхдальней радиолокации (особенно при локации других планет). Поскольку моменты прихода сигнала априори неизвестны, измерения производятся не только в эти моменты, но и в промежуточные моменты времени (равноотстоящие друг от друга).

Таким образом, возникает ряд измерений $x_1, x_2, \ldots, x_n$, причем известен период k посылки, а значит, и прихода интересующего нас полезного сигнала. Для выделения этого сигнала вычислим k средних значений, предполагая, что общее число измерений n кратно k, т. е. $n = mk$. Обозначим через $\bar{x}_r$ среднее значение, вычисляемое по формуле

$$\bar{x}_r = \frac{1}{m}\sum_{i=0}^{m-1} x_{ki+r}, \qquad r = 1, \ldots, k.$$

Если полезный сигнал приходит в момент $r = r_0$ (а значит, и во все моменты времени $ki + r_0$ для $i = 0, 1, \ldots, m-1$), то среднее значение $\bar{x}_{r_0}$ при $n \to \infty$ будет стремиться (стохастически) к $\bar{x}$, а все средние $\bar{x}_r$ при $r \neq r_0$ — к нулю. Таким образом, при достаточно большом числе измерений можно со сколь угодно высокой степенью достоверности найти моменты времени $r_0 + ki$ ($i = 0, \ldots, m-1$) прихода полезного сигнала, а также вычислить его величину.

9.2.3. Экспериментальные распределения вероятностей. При измерениях случайной величины x в ряде случаев оказывается недостаточным определение лишь ее среднего значения и дисперсии. Наибольшую информацию дает *распределение вероятностей* случайной величины, которое может быть представлено в двух видах: дифференциальном и интегральном. *Интегральное распределение* вероятностей случайной величины x представляет собой функцию $P(s)$, при любом s равную вероятности выполнения неравенства $x < s$. Если $P(s)$ — дифференцируемая функция, то ее производная $p(s) = \frac{d}{ds}P(s)$ представляет собой *плотность вероятности* случайной величины x. Иными словами, $p(s)$ представляет

собой предел отношения вероятности того, что случайная величина x принимает значение в интервале $s \leqslant x < s + \Delta s$, к длине этого интервала Δs при $\Delta s \to 0$.

При экспериментальном определении законов распределения случайных величин вероятность того или иного события заменяется *частотой* его повторяемости в эксперименте, т. е. отношением числа экспериментов, в которых данное событие имеет место, к общему числу проделанных экспериментов. Для получения *дифференциального закона распределения* (плотности вероятности $p(s)$) интервал значений, которые принимала в эксперименте случайная величина x, разбивается на равные интервалы Δ_i ($i = 1, \ldots, k$). Если обозначить через n_i число экспериментов (измерений), при которых случайная величина x принимала значение в интервале Δ_i, а через n — общее число измерений, то экспериментальное значение функции плотности вероятности $p(s)$ на интервале Δ_i принимается постоянным и равным $\frac{1}{\Delta}\frac{n_i}{n}$, где через Δ обозначена длина интервала. При графическом изображении дифференциального закона распределения плотности вероятности часто вместо относительных долей n_i/n используют сами числа n_i (что равносильно просто изменению масштаба вдоль оси ординат). Обычно опускают также и деление на (постоянную) длину интервала Δ.

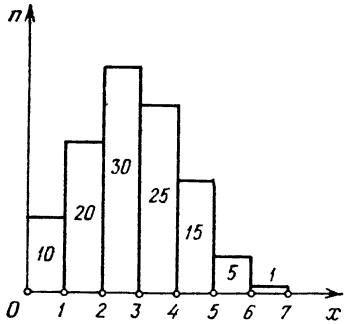

Рис. 9.1

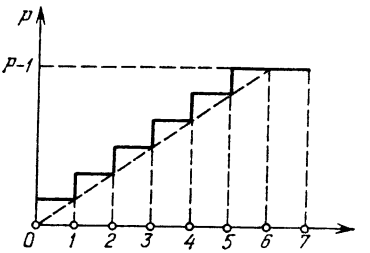

Рис. 9.2

Полученная ступенчатая линия носит наименование *гистограммы*. На рис. 9.1 приведен пример гистограммы для случайной величины x. Из нее явствует, что в эксперименте значения величины x, большие или равные нулю, но меньшие единицы, встречаются 10 раз, значения между 1 и 2 — 20 раз и т. д. Если случайная величина x может принимать лишь целочисленные значения, приведенная гистограмма означает, очевидно, что в эксперименте значения $x = 0$ встречались 10 раз, значения $x = 1$ — 20 раз и т. д.

При построении графика экспериментального интегрального закона распределения случайной величины в соответствии с определением этого закона также получается ступенчатая линия. Однако для многих приложений ее удобно заменять приближением, получаемым интегрированием дифференциального закона распределения. В этом случае интегральное распределение представляется в виде непрерывной ломаной. На рис. 9.2 сплошной линией изображено точное, а штриховой — приближенное экспериментальное интегральное распределение для целочисленной случайной величины с заданной на рис. 9.1 гистограммой.

Заметим, что если величина x не является целочисленной, то точное интегральное экспериментальное распределение разбивается, вообще говоря, внутри изображенных на рис. 9.2 целочисленных интервалов на более мелкие ступеньки. В этом случае оно будет, очевидно, еще ближе к приближению, изображенному штриховой линией.

9.2.4. Мода и медиана. Помимо обычного среднего значения случайной величины x, о котором шла речь в п. 9.2.1, иногда оказывается полезным вычисление еще двух значений, также в каком-то смысле играющих роль средних значений,— моды и медианы.

Мода представляет собой наиболее вероятное значение случайной величины x, соответствующее абсолютному максимуму плотности ее вероятности. В экспериментальной гистограмме рис. 9.2 для случая целочисленной величины x модой, очевидно, будет $x=2$. Если рассматривать ступенчатую линию гистограммы как приближение некоторой непрерывной кривой, то, будучи вычислено для этой кривой, значение моды будет расположено между 2 и 3.

Медианой называется значение $x=s$ случайной величины, при котором интегральная вероятность $P(s)$ равна 1/2. При экспериментальном определении закона распределения для медианы m получается, как правило, не какое-то определенное значение, а лишь оценки снизу и сверху: $s_1 < m < s_2$, где $P(s_1) < 1/2$, $P(s_2) > 1/2$, где s_1 и s_2 — два значения величины x, между которыми нет промежуточных (измеренных)*). Используя линейное приближение для интегрального закона распределения на отрезке $[s_1, s_2]$, получим приближение для медианы:

$$m = s_1 + \Delta_1 = s_2 - \Delta_2, \text{ где } \Delta_1 = \frac{0{,}5 - P(s_1)}{P(s_2) - P(s_1)}(s_2 - s_1),$$

$$\Delta_2 = \frac{P(s_2) - 0{,}5}{P(s_2) - P(s_1)}(s_2 - s_1). \quad (9.4)$$

*) Разумеется, может случиться, что для какого-то s_i имеет место точное равенство $P(s_i) = 1/2$. Тогда $x = s_i$ и будет, очевидно, значением медианы.

При использовании медианы вместо среднего квадратичного отклонения в качестве меры разброса распределения используются так называемые *квартили*. *Нижний квартиль* q_1 представляет собой значение случайной величины с интегральным законом распределения $P(s)$, при котором $P(q_1)=1/4$. Аналогично, *верхний квартиль* q_2 определяется равенством $P(q_2)=3/4$. При экспериментальном определении квартилей используется тот же прием, что и при определении медианы. В формулах (9.4) при этом величина 0,5 заменяется соответственно на 0,25 и 0,75.

9.3. Интерполяция, экстраполяция и аппроксимация

Одной из более сложных задач обработки результатов измерений является задача нахождения функциональной зависимости между переменными по конечному числу измерений. Мы рассмотрим такую задачу для функции одного переменного $y=f(x)$, значения которой измерены лишь в конечном числе точек: $y_i=f(x_i)$ $(i=1, \ldots, n)$. В задаче *интерполяции* в широком смысле слова обычно требуется найти уравнение кривой $y=f(x)$, проходящей через все заданные точки $A_i(x_i, y_i)$ $(i=1, \ldots, n)$.

В задаче *аппроксимации* искомая кривая должна пройти в том или ином смысле б л и з к о к заданным точкам. Обе эти задачи не имеют, разумеется, однозначного решения. Для достижения однозначности требуется уточнить вид интерполирующей (аппроксимирующей) функции. В случае, когда вид функции априори неизвестен, обычно прибегают к полиномиальной интерполяции (аппроксимации), используя при этом полиномы возможно более низкой степени.

Точное решение (интерполяции) для n точек $A_i(x_i, y_i)$ $(i=1, \ldots, n)$ всегда может быть получено в виде полинома $y=L(x)$ $(n-1)$-й степени, который однозначно определяется этими точками*). Одной из простейших форм представления интерполяционного полинома является интерполяционный полином Лагранжа

$$L(x)=y_1\frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)}+$$
$$+y_2\frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)}+\ldots$$
$$\ldots+y_n\frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})}. \quad (9.5)$$

При $x=x_1$ первый член суммы обратится в y_1, тогда как остальные

*) Не исключен случай вырождения, когда степень полинома $L(x)$ оказывается меньше $n-1$.

ные члены будут равны нулю, при $x = x_2$ второй член будет равен y_2, а остальные равны нулю и т. д.

Пример. Пусть при $x = 0, 1, 2$ переменная y принимает соответственно значения 0, 1, 3. Тогда

$$L(x) = 0 + 1 \cdot \frac{(x-0)(x-2)}{(1-0)(1-2)} + 3 \cdot \frac{(x-0)(x-1)}{(2-0)(2-1)} =$$

$$= -x(x-2) + \frac{3}{2} x(x-1) = \frac{1}{2} x(x+1).$$

Непосредственной проверкой убеждаемся, что кривая $y = \frac{1}{2} x(x+1)$ действительно проходит точки (0, 0), (1, 1) и (2, 3).

Зная интерполирующую функцию $y = L(x)$, можно вычислять ее значения при промежуточных значениях аргумента x, не измеренные непосредственно в эксперименте. В этом, собственно говоря, и состоит задача интерполяции в узком смысле слова. Например, при $L(x) = x(x+1)/2$ легко находим $L(1/2) = 3/8$, $L(3/2) = 15/8$. Как правило, для подавляющего большинства измеряемых в эксперименте функциональных зависимостей полиномиальная интерполяция дает достаточно хорошие результаты.

С меньшей степенью уверенности можно распространить найденную зависимость за пределы интервала выполненных измерений, решая тем самым так называемую задачу *экстраполяции*. Например, при $y = L(x) = x(x+1)/2$ для $x = 3$ имеем $y = 6$, для $x = 4$ имеем $y = 10$ и т. д. Впрочем, достаточно близко к интервалу выполненных измерений экстраполяция может дать столь же хорошие результаты, как и интерполяция. Например, $L(2, 2) = 3{,}52$ будет с большой степенью уверенности хорошей оценкой для результата измерения рассмотренной выше функции в точке $x = 2{,}2$.

Чтобы избежать появления полиномов высоких степеней, часто при полиномиальной интерполяции (экстраполяции) используют лишь две или три соседние точки. Эти простейшие виды интерполяции (экстраполяции) носят соответственно наименования *линейной* и *квадратичной интерполяции* (*экстраполяции*). В этом случае искомая (интерполирующая) функция будет иметь разные выражения на различных участках интервала измерений.

Полиномиальные представления являются достаточно удобными, однако ничто не мешает использовать для целей интерполяции и особенно для целей экстраполяции функции другого вида. Так, если известно, что искомая функция $y = f(x)$ является периодической с периодом T, то для ее представления лучше всего воспользоваться отрезком *ряда Фурье*:

$$y = f(x) = a_0 + a_1 \cos px + b_1 \sin px + \ldots + a_n \cos px + b_n \sin px, \tag{9.6}$$

где $p = 2\pi/T$, а коэффициенты a_i и b_i могут быть определены из системы $2n+1$ линейных алгебраических уравнений, получающихся в результате подстановок координат заданных точек в левую и правую части уравнения (9.6).

В других случаях могут возникать системы нелинейных и даже трансцендентных уравнений. Например, для функции $f(x) = ae^{bx}$ и точек $A_1(1, 2)$, $A_2(2, 5)$ возникает система двух трансцендентных уравнений:

$$2 = ae^{b}, \quad 5 = ae^{2b}.$$

Разделив левые и правые части этих уравнений друг на друга, получим $e^b = 2{,}5$, $b = \ln 2{,}5$; из первого уравнения находим $a = 2e^{-b} = 0{,}8$. В более сложных случаях приходится прибегать к различным численным методам решения систем уравнений. С помощью современных ЭВМ такие задачи решаются относительно просто.

В задачах аппроксимации также имеют дело с тем или иным классом функций $y = f(x, a_1, \ldots, a_m)$, зависящих от параметров $a_1, a_2, \ldots, a_m$, например, полиномов заданной степени с неизвестными коэффициентами. Используя заданные точки $A_i(x_i, y_i)$, составляют разности $d_i = f - y_i$ $(i = 1, \ldots, n)$ и выбором значений параметров $a_1, \ldots, a_m$ минимизируют ту или иную целевую функцию от этих разностей. В качестве такой функции чаще всего выбирается сумма квадратов разностей или максимальное значение абсолютной величины таких разностей. В первом случае говорят об аппроксимации по *методу наименьших квадратов*, во втором — о *чебышевской* (*минимаксной*) *аппроксимации*. Во всех случаях для фактического нахождения аппроксимирующей функции требуется решить некоторую оптимизационную задачу. В методе наименьших квадратов искомое решение находится обычно из системы уравнений:

$$\sum_{j=1}^{m} d_j \frac{\partial d_j}{\partial a_j} = 0 \qquad (9.7)$$

при подстановке в d_i координат заданных точек $A_i(x_i, y_i)$ $(i = 1, \ldots, n)$.

Рассмотрим пример линейной аппроксимации $y = ax + b$ зависимости, заданной точками $A_1(0, 2)$, $A_2(1, 3)$, $A_3(2, 5)$, по методу наименьших квадратов. Иными словами, параметры a и b должны быть найдены из условия минимума выражения $z = (b-2)^2 + (a+b-3)^2 + (2a+b-5)^2$. Легко найдем, что $\partial z/\partial a = 10a + 6b - 26$, $\partial z/\partial b = 6a + 6b - 20$. Из системы уравнений $\partial z/\partial a = 0$, $\partial z/\partial b = 0$ найдем $a = 3/2$, $b = 11/6$, так что искомая линейная аппроксимация выражается функцией $y = 3x/2 + 11/6$. Заметим, что

график этой функции не проходит ни через одну из заданных точек.

Задачи интерполяции и аппроксимации, рассмотренные нами для функций одного переменного, легко распространяются и на функции многих переменных.

9.3.1. Интерполяция и экстраполяция при наличии помех. Центральным вопросом для получения хорошей интерполяции и особенно экстраполяции является правильный выбор экстраполирующей функции $y = f(x, a_1, \ldots, a_m)$. Решение этого вопроса, достаточно сложное само по себе, еще более усложняется при наложении на основную закономерность изменения экстраполируемой функции различного рода случайных помех. Предположим сначала, что помеха мала по абсолютной величине и порядок ее величины приблизительно известен. Кроме того, будем считать, что помеха действует с равной вероятностью как в сторону увеличения изучаемой величины $y(x)$, так и в сторону ее уменьшения, так что среднее значение помехи равно нулю.

Приведем теперь в описанных условиях один достаточный признак для полиномиальной экстраполяции. Он основан на использовании *разделенных разностей* для любой функции $y = F(x)$, заданной на конечном множестве точек $x_1, x_2, \ldots, x_n$. Обозначим через $y_1, y_2, \ldots, y_n$ значения функции $F(x)$ в соответствующих точках. Тогда

первые разделенные разности вычисляются по формуле

$$\Delta y_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad i = 1, 2, \ldots, n-1; \tag{9.8}$$

вторые разделенные разности:

$$\Delta^2 y_i = \frac{\Delta y_{i+1} - \Delta y_i}{x_{i+2} - x_i}, \quad i = 1, 2, \ldots, n-2; \tag{9.9}$$

третьи разделенные разности:

$$\Delta^3 y_i = \frac{\Delta^2 y_{i+1} - \Delta^2 y_i}{x_{i+3} - x_i}, \quad i = 1, 2, \ldots, n-3; \tag{9.10}$$

разделенные разности порядка $k \leqslant n-1$:

$$\Delta^k y_i = \frac{\Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i}{x_{i+k} - x_i}, \quad i = 1, 2, \ldots, n-k. \tag{9.11}$$

Если $F(x)$ — полином k-й степени, то его $(k+1)$-е разделенные разности обращаются в нуль. При точных методах экстраполяции (проведении экстраполирующей кривой через заданные точки) обращение в нуль разделенных разностей порядка p означает возможность экстраполяции полиномом степени $p-1$. При

малой помехе с нулевым средним значением условие обращения p-х разностей в нуль должно быть заменено совокупностью двух условий: 1) абсолютные величины разделенных разностей p-го порядка малы (сравнимы с величиной помехи); 2) среднее значение этих разностей имеет еще меньший порядок величины. При этих условиях для аппроксимации экстраполирующей функции следует выбрать полином $(p-1)$-й степени.

Пусть, например, функция задана множеством точек (1, 1); (2, 3); (3, 9); (4, 15); (5, 26); (6, 37); (7, 48); (8, 65). Нетрудно видеть, что значения функции $y(x)$ с точностью до ± 1 совпадают с x_i^2. Иными словами, на квадратичную зависимость наложена помеха, принимающая значение ± 1. Первые разделенные разности для заданных значений функции равны соответственно 2, 6, 6, 11, 11, 11, 17. Для вторых разностей получим 2, 0, 5/2, 0, 0, 3, а для третьих $\pm 2/3$, $5/6$, $-5/6$, 0, 1. Среднее значение для вторых разностей равно $(2+5/2+3)/6=1{,}25$, для третьих $(-4+5-5+6)/(6\cdot 5)=1/15$. Порядок величин третьих разностей сравним с величиной помехи, а их среднее на порядок меньше. Таким образом, приходим к выводу о необходимости применить квадратичную экстраполяцию. Этот вывод вполне согласуется с тем, что заданные значения функции $y(x)$ могут быть вычислены по формуле $y=x^2+\varepsilon(x)$, где $\varepsilon(x)$ — случайная помеха.

Описанный признак может быть распространен и на некоторые неполиномиальные зависимости. С этой целью используется подходящая замена переменных, сводящая исходную зависимость к полиномиальной. Например, зависимость

$$y=a_0+a_1e^x+a_2e^{2x}+\ldots+a_ne^{nx}$$

подстановкой $t=\ln x$ сводится к полиномиальной зависимости

$$x=a_0+a_1t+a_2t^2+\ldots+a_nt^n.$$

Подстановка $z=\ln y$, $s=\ln x$ превращает зависимость вида $y=a/x$ в линейную зависимость $z=\ln a-s$ и т. д.

Указанный прием использует так называемые *функциональные шкалы*. Суть их заключается в том, что на шкале от начала координат откладываются отрезки, длина которых равна не значениям самой величины, а есть некоторая заданная функция от этих значений. На рис. 9.3 в качестве примера на оси абсцисс построена *логарифмическая шкала*. Ее использование по оси ординат превращает в прямую линию график любой функции вида $y=ae^{bx}$. Логарифмические шкалы по обеим осям линеаризуют зависимости вида $y=ax^b$.

Иногда вид экстраполирующей функции может быть подсказан самой природой изучаемого процесса. Так, динамику розничного спроса на тот или иной товар естественно характеризовать

произведением $f(t)\varphi(t)\psi(t)$, где $\varphi(t)$, $\psi(t)$ — периодические функции с периодами, равными соответственно неделе и половине месяца (выплата зарплат). Для характеристики сезона может добавиться третий периодический множитель с периодом в один год, характеризующий сезонные изменения спроса. Этот множитель $f(t)$ определяет динамику изменения среднегодового спроса. Он может экстраполироваться с помощью полиномиальных или экспоненциальных приближений. Для периодических множителей, как уже отмечалось выше, можно использовать приближения в виде начальных отрезков соответствующих рядов Фурье.

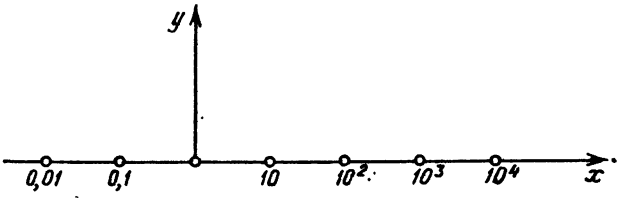

Рис. 9.3

Обычные методы аппроксимации дают при экстраполяции хорошие результаты лишь тогда, когда случайная помеха имеет сравнительно небольшую величину. Если это условие не выполняется, то необходимо предварительно отфильтровать помеху. В ряде случаев такая фильтрация может быть сделана на основании содержательного представления об изучаемом процессе *).

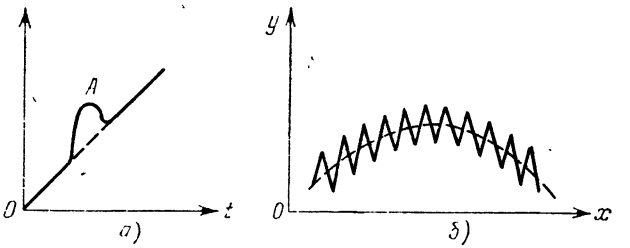

Рис. 9.4

Рассмотрим в качестве примера динамику ремонтно-восстановительных работ жилого фонда некоторого города за ряд лет (рис. 9.4, *а*). Допустим, что выброс A, имеющийся на кривой, явился результатом стихийного бедствия, которое случается в среднем раз в тысячу лет. Размер этой «помехи» настолько велик, что ее учет при экстраполяции сильно исказил бы реальную

*) Использование подобного приема требует диалогового метода решения задачи.

картину. Отфильтровав помеху (т. е. удалив из графика расходы, вызванные стихийным бедствием), можно значительно улучшить прогноз.

Другой случай, когда задача фильтрации помех получает достаточно удовлетворительное решение, имеет место при условии, что основной процесс выражается сравнительно медленно меняющейся функцией, в то время как помеха (не обязательно малая по величине), сохраняя нулевое среднее значение, меняет знак с большой частотой. Характерная картина такого процесса изображена на рис. 9.4, *б*. Одним из методов, позволяющих (хотя бы частично) отфильтровать помеху, является *метод сглаживания* на основе простого усреднения. Пусть $y_0, y_1, \ldots, y_n$ — значения функции. Предположим, что точки $x_0, x_1, \ldots, x_n$, в которых вычислены эти значения, расположены на оси абсцисс достаточно густо, чтобы уловить все «взлеты» и «падения» быстро-периодической составляющей изучаемой функции. Выберем интервал Δ так, чтобы он был существенно больше среднего интервала Δ_0, на котором быстро-периодическая составляющая процесса меняет знак, и в то же время был меньше интервала Δ_1, на котором медленно меняющаяся составляющая процесса изменится существенно с точки зрения принятой точности экстраполяции *).

Двигая Δ вдоль отрезка $[x_1, x_n]$, заменяем для всех попавших в него точек $x_p, x_{p+1}, \ldots, x_{p+k}$ соответствующие значения $y_p, \ldots, y_{p+k}$ их средними значениями $\frac{1}{k+1}(y_p + y_{p+1} + \ldots + y_{p+k})$, относя эти значения к середине соответствующего отрезка. При усреднении значения быстро-периодической составляющей гасят друг друга и медленно меняющаяся часть процесса проявляется более отчетливо.

Когда точки x_i расположены на равных расстояниях друг от друга, все сводится к выбору числа q и замене значения y_q на

$$y_q = \frac{1}{2q+1}(y_0 + y_1 + \ldots + y_{2q}), \tag{9.12}$$

значения y_{q+1} — на

$$y_{q+1} = \frac{1}{2q+1}(y_1 + y_2 + \ldots + y_{2q+1}) \tag{9.13}$$

*) Если величины интервалов Δ_0 и Δ_1 имеют один и тот же порядок, то неизбежно падение точности экстраполяции. Заметим также, что условие малости изменения медленно меняющейся составляющей процесса на интервале Δ можно заменить условием малости изменения ее первой производной.

и т. д. Наилучший выбор интервала Δ (или числа q) определяется предположениями о свойствах быстро-периодической и медленно меняющейся составляющих функции $y(x)$ и требует специального дополнительного исследования.

Иногда при сглаживании «загрубляют» приближение, используя средние значения для не перекрывающих друг друга интервалов. Так, вместо ежедневных значений спроса (365 точек за год) изучают поведение среднего спроса за каждую неделю (52 точки за год).

Второй способ фильтрации быстро-периодических составляющих основан на использовании спектрального анализа и излагается ниже.

9.3.2. Экстраполяция случайных процессов. Отметим одно важное различие, которое имеется при экстраполяции детерминированных процессов (быть может, со случайными помехами) и чисто случайных процессов. Смысл экстраполяции детерминированного процесса состоит в угадывании закономерности, которая имеет место для всего изучаемого участка процесса. В случайных процессах встречаемся с противоположным явлением, поскольку поиск такой закономерности противоречит самой сущности процессов. Правда, свойство непрерывности случайного процесса приводит к тому, что его значения в близких друг к другу точках взаимозависимы (хотя и не в строго функциональном, а лишь в вероятностном смысле). По мере удаления точек друг от друга эта зависимость делается все более слабой. Методы экстраполяции случайных процессов должны учитывать это обстоятельство, отдавая большее предпочтение точкам, к настоящему моменту близким по сравнению с более далекими точками.

При применении метода наименьших квадратов отмеченный факт может быть учтен введением специальных весовых коэффициентов $\lambda_1, \lambda_2, \ldots, \lambda_n$, убывающих по мере удаления в прошлое. При равномерном расположении точек $x_1, \ldots, x_n$ (в которых задан процесс) вдоль оси абсцисс лучше всего воспользоваться экспоненциальным законом убывания коэффициентов λ_i (справа налево). Иными словами, выбрав некоторое положительное число $r > 1$, полагаем

$$\lambda_1 = r^{(n-1)}, \ \lambda_2 = r^{(n-2)}, \ \ldots, \ \lambda_{n-1} = r, \quad \lambda_n = 1.$$

Подобный метод выбора весовых коэффициентов носит наименование *экспоненциального взвешивания*. При экстраполирующей функции взвешивания $f(x, a_1, \ldots, a_m)$ сумма квадратов, которую требуется минимизировать, запишется в виде

$$S_r = \sum_{i=1}^{n} r^{n-i} (f(x_i, a_1, \ldots, a_m) - y_i)^2, \qquad (9.14)$$

Результат экстраполяции не изменится, если вместо $r<1$ взять число $R=1/r>1$ и записать взвешенную сумму квадратов

$$S_R=\sum_{i=1}^{n} R^{i-1}[f(x_i, a_1, \ldots, a_m)-y_i]^2. \qquad (9.15)$$

Выбор величины r (или соответственно R) зависит от степени гладкости процесса. Для ее определения необходимо более глубокое исследование рассматриваемого процесса и, прежде всего, изучение его автокорреляционной функции.

9.4. Корреляция и автокорреляция

На практике часто возникает задача — установить, являются ли две наблюдаемые случайные величины x и y независимыми или связаны какой-либо зависимостью (не обязательно строго детерминированной). С этой целью рассматриваются две последовательности значений наблюдаемых величин $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_n$, так что искомая зависимость (в случае ее существования) должна иметь место между соответствующими членами последовательностей (с одинаковыми индексами). Обозначим через $\bar{x}$ и $\bar{y}$ средние значения этих величин, а через σ_x и σ_y — их средние квадратичные отклонения. Величина

$$r=\frac{1}{n\sigma_x\sigma_y}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}) \qquad (9.16)$$

называется *коэффициентом корреляции* двух рассматриваемых последовательностей значений. Величина r может меняться в пределах от -1 до $+1$.

Значениям коэффициента корреляции, равным ± 1, соответствует наличие детерминированной линейной зависимости $y_i=ax_i+b$ $(a\neq 0)$ (или $x_i=cy_i+d$, где $c=1/a$, $d=-b/a$) между величинами x и y, а значение $r=0$ означает (с определенной долей достоверности) отсутствие связи между ними. При этом в обеих последовательностях $\{x_i\}$ и $\{y_i\}$ не исключаются повторяющиеся значения.

Если обозначить через $\bar{y}_x$ среднее значение величины y, вычисленное для всех значений второй последовательности, отвечающих одному и тому же значению (равному x) величины второй последовательности, то получим уравнение

$$\bar{y}_x-\bar{y}=r\frac{\sigma_y}{\sigma_x}(x-\bar{x}), \qquad (9.17)$$

которое называется *уравнением регрессии y на x*. Аналогично уравнение

$$\bar{x}_y-\bar{x}=r\frac{\sigma_x}{\sigma_y}(y-\bar{y}) \qquad (9.18)$$

называются *уравнением регрессии x на y*. Уравнения регрессии дают наилучшие (по методу наименьших квадратов) линейные аппроксимации для зависимостей условных (при заданном y или x) средних значений величин y и x от величин x и y соответственно.

Разумеется, линейные аппроксимации могут плохо отражать истинную природу указанных зависимостей. Для нахождения нелинейных аппроксимаций (нелинейных регрессий) в случае необходимости могут быть использованы общие методы, изложенные в § 9.3, которые применяются к последовательностям условных средних значений измеряемых величин. Например, для $x=1, 2, 1, 3, 2, 1, 3$, $y=2, 3, 4, 5, 3, 6, 3$ условные средние величины y для $x=1, 2, 3$ равны соответственно $\bar{y}_{x=1}=(2+4+6)/3$, $\bar{y}_{x=2}=(3+3)/2=3$, $\bar{y}_{x=3}=(5+3)/2=4$, так что искомая нелинейная аппроксимация (регрессия) зависимости величины y от величины x представится функцией $y=f(x)$, для которой $f(1)=4$, $f(2)=2$, $f(3)=4$, откуда $y=(3x^2-11x+14)/2$.

Имея в виду значение коэффициента корреляции r для установления связей между измеряемыми величинами, программы для его вычисления обычно включают в состав математического обеспечения измерительных комплексов.

Для последовательностей $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_k$ значений случайных величин x и y может вычисляться коэффициент корреляции r_k со сдвигом k $(0 \leqslant k \leqslant n)$:

$$r_k = \frac{1}{(n-k)\,\sigma_x \sigma_y} \sum_{i=0}^{n-k} (x_{i+k} - \bar{x})(y_i - \bar{y}), \tag{9.19}$$

где средние значения $\bar{x}$ и $\bar{y}$, а также средние квадратичные отклонения σ_x, σ_y вычисляются для отрезков последовательностей: $x_{k+1}, \ldots, x_n, y_1, \ldots, y_{n-k}$. Аналогично при отрицательном $k=-|k|$ $(0<k<n)$ коэффициент сдвинутой корреляции равен

$$r_k = \frac{1}{(n-k)\,\sigma_x \sigma_y} \sum_{i=|k|+1}^{n} (x_{i+k} - \bar{x})(y_i - \bar{y}). \tag{9.20}$$

Будучи функцией целочисленного параметра k, коэффициент r_k может рассматриваться как (дискретная) *функция взаимной корреляции* двух заданных последовательностей. Если даже обычный коэффициент корреляции $r=r_0$ равен нулю, то неравенство нулю коэффициента r_k при некотором $k=0$ может рассматриваться как наличие некоторой стохастической зависимости между развертывающимися во времени дискретными случайными процессами x и y (с запаздыванием или опережением).

Если вторая последовательность совпадает с первой (т. е. $x_i=y_i$ для всех $i=1, \ldots, n$), то выражения (9.19) и (9.20) задают

(дискретную) *автокорреляционную функцию* процесса $x_1, x_2, \ldots$ $\ldots, x_n$. Она характеризует величину зависимости последующих членов от его предыдущих значений.

Будучи построена для непрерывных случайных процессов, теория корреляционных функций использует вместо сумм интегралы. Однако при численном решении возникающих здесь задач интегралы приходится заменять суммами, так что в результате мы возвращаемся к уже рассмотренному дискретному случаю.

9.5. Спектральный анализ

Как известно, любая непрерывная функция $y = f(x)$ вещественного переменного, заданная на любом отрезке длины l, имеющая на этом отрезке конечное число локальных экстремумов и принимающая на его концах равные значения, может быть разложена на этом отрезке в *ряд Фурье*

$$f(x) = \sum_{k=0}^{\infty} \left(a_k \cos \frac{2\pi}{l} kx + b_k \sin \frac{2\pi}{l} kx \right). \tag{9.21}$$

Полагая, $h_k = + \sqrt{a_k^2 + b_k^2}$ и $\psi_k = \operatorname{arctg}(a_k/b_k)$, можно представить этот ряд в виде

$$f(x) = \sum_{k=0}^{\infty} h_k \sin \left(\frac{2\pi}{l} kx + \psi_k \right). \tag{9.22}$$

Каждый член ряда представляет собой *гармонику* (гармоническое колебание) с *амплитудой* h_k и *частотой* k/l (колебаний в секунду, если x — время в секундах), или $2\pi k/l$ (радиан в секунду), и с *фазой* ψ_k (радиан). Легко видеть, что при изменении положения отрезка (области определения функции $f(x)$) на оси абсцисс в разложении (9.22) амплитуды и частоты остаются неизменными — меняются одни лишь фазы. При изменении масштаба по оси абсцисс меняются частоты и фазы, амплитуды остаются неизменными. Зависимость амплитуды h_k от частоты k/l (или $2\pi k/l$) называется *амплитудно-частотной характеристикой*, а зависимость фазы ψ_k от частоты — *фазово-частотной характеристикой* функции $f(x)$. Правая часть формулы (9.22) называется *спектральным разложением* или просто *спектром* функции $f(x)$. Впрочем, зачастую спектр функции отождествляется просто с ее амплитудно-частотной характеристикой. Нахождение спектров функций составляет задачу так называемого *спектрального* (или гармонического) *анализа*.

Разложение в ряд Фурье допускают не только непрерывные функции, но и функции, имеющие (на заданном отрезке) конечное число разрывов первого рода. Каждая такая точка $x = c$ ха-

рактеризуется наличием конечных пределов $f(c-0)$ и $f(c+0)$ при стремлении x к точке разрыва соответственно слева и справа. В подобных точках ряд Фурье сходится к среднему арифметическому обоих пределов. Аналогично, если на концах отрезка c_1 и c_2 функция имеет разные значения, то ряд Фурье на обоих концах сходится к среднему арифметическому этих значений (в общем случае $(f(c_1+0)+f(c_2-0))/2$).

В практическом спектральном анализе предпочитают иметь дело с рядами Фурье в комплексной области. Поскольку $e^{w} = \cos w + i \sin w$, то оказывается возможным ряд (9.21) представить в виде

$$f(x) = \sum_{k=0}^{\infty} r_k e^{i\frac{2\pi}{l}kx}, \tag{9.23}$$

где $r_k = a_k - b_k i$. Правда, для такого представления в случае вещественной функции $f(x)$ требуется соблюдение дополнительного условия сходимости ряда (9.21) к нулю. Однако оказывается, что если функция разложима в обычный ряд Фурье, это условие выполняется. При этом амплитуда и фаза спектрального разложения будут совпадать соответственно с модулем и аргументом комплексного коэффициента r_n (с точностью до знака).

9.5.1. Дискретное преобразование Фурье. Коэффициенты комплексного ряда Фурье (9.23) легко вычисляются интегрированием по заданному отрезку $[c_1, c_2]$:

$$r_k = \frac{1}{l}\int_{c_1}^{c_2} f(x)\, e^{-i\frac{2\pi}{l}kx}\, dx.$$

При дискретном задании функции $f(x)$ ее значения известны лишь в конечном числе точек. Обычно предполагают, что соседние точки находятся на равных расстояниях друг от друга, с шагом, который мы обозначим d, так что общее число точек на отрезке (считая оба конца) будет равно $l/d+1 = N+1$.

Поскольку амплитудно-частотная характеристика функции (ее спектр в узком смысле слова) не зависит от положения отрезка на оси абсцисс, его левый конец обычно совмещается с началом координат, так что координаты x_n точек, в которых задана функция $f(x)$, задаются формулой $x_n = ln/N = nd$ $(n=0, \ldots, N)$.

Стало быть, можно заменить интеграл суммой

$$\frac{1}{l}\sum_{n=0}^{N-1} f(x_n)\, e^{-i\frac{2\pi}{l}kx_n} d.$$

Величина $2\pi/l = \delta$ представляет собой элементарную частоту (в радианах), так что частота w_k k-й гармоники выражается фор-

мулой $w_k = k\delta$. В результате для k-го коэффициента r_k комплексного ряда Фурье получим формулу

$$r_k = \frac{1}{N}\sum_{n=0}^{N-1} f(nd)\, e^{-i\delta dkn}, \quad \delta = \frac{2\pi}{l}, \quad d = \frac{l}{N}. \tag{9.24}$$

Формула (9.24) представляет собой приближенное выражение для коэффициента r_k. Однако оказывается, что сумма N первых членов ряда (9.23) при вычислении коэффициентов r_k по этой формуле точно совпадает со значениями функции $f(x)$ в точках ln/N $(n = 0, \ldots, N-1)$. Перенося множитель $1/n$ из (9.24) в (9.23) и обозначая $r_k = F(k\delta)$, получим две (точные) формулы:

$$F(k\delta) = \sum_{n=0}^{N-1} f(nd)\, e^{-i\delta dkn}, \quad k = 0, 1, \ldots, N-1, \tag{9.25}$$

$$f(nd) = \frac{1}{N}\sum_{k=0}^{N-1} F(k\delta)\, e^{i\delta dkn}, \quad n = 0, 1, \ldots, N-1, \tag{9.26}$$

где d — интервал дискретности аргумента (обычно времени), а δ — интервал дискретности частоты, причем $\delta d = 2\pi/N$. Первая из этих формул называется *прямым*, а вторая — *обратным дискретным преобразованием Фурье* (ДПФ). Модуль $|F(k\delta)|$ дает умноженную на N *амплитудно-частотную характеристику* (*спектр*) преобразуемой функции $f(nd)$.

9.5.2. Быстрое преобразование Фурье. Вычисление по формуле (9.25) для всех частот δ_k $(k = 0, 1, \ldots, N-1)$ требует N^2 умножений и сложений комплексных чисел, что при больших N может сильно затруднить спектральный анализ даже при использовании мощных ЭВМ. Однако, используя различные искусственные приемы для вычисления суммы парных произведений, можно значительно ускорить выполнение ДПФ. Алгоритмы для подобных вычислений принято называть *быстрым преобразованием Фурье* (БПФ).

Обозначая известные (измеренные) значения $f(nd)$ преобразуемой функции через f_n, величину $e^{-i\delta d} = e^{-i2\pi/N}$ — через w, а $F(\delta_k)$ — через F_k, представим формулу (9.25) в виде

$$F_k = \sum_{n=0}^{N-1} f_n w^{kn}, \quad w = e^{-i2\pi/N}. \tag{9.27}$$

Если N делится на 2, то, обозначая $\sum_{l=0}^{N/2-1} f_{2l}(w^2)^{lk}$ через G_k, а $\sum_{l=0}^{N/2-1} f_{2l+1}(w^2)^{lk}$ — через H_k, легко найдем, что

$$F_k = G_k + w^k H_k. \tag{9.28}$$

Так как $k \leqslant N-1$, то при вычислении значений G_k и H_k обычным способом (не считая вычисления w^2), т. е. за $(N/2)^2$ операций, на вычисления F_k по формуле (9.28) будет затрачено не более $1+N+N^2/2$ операций. В действительности, используя при вычислении w^k значение w^2, можно еще более уменьшить эту оценку.

Если число N представляет собой степень двойки ($N=2^m$), то указанный прием редукции, называемый *прореживанием по аргументу* (или по времени, если x в $f(x)$ есть время), применяемый последовательно m раз, позволяет снизить оценку для общего числа операций с N^2 до $N\log_2 N$. При $N=1024=2^{10}$ это составляет уменьшение времени счета более чем в 100 раз. При дальнейшем увеличении N экономия становится еще более значительной.

Наряду с прореживанием по аргументу можно использовать *прореживание по частоте*, основанное на формулах

$$F_{2k}=\sum_{l=0}^{N/2-1}(f_l+f_{l+N/2})\,(w^2)^{lk}, \tag{9.29}$$

$$F_{2k+1}=\sum_{l=0}^{N/2-1}[(f_l-f_{l+N/2})\,w^l]^{lk}. \tag{9.30}$$

Как и в предыдущем случае, для количества операций Фурье $N=2^m$ справедлива оценка $N\log_2 N$.

Заметим, что операции прореживания сводят задачу ДПФ к задаче ДПФ с меньшим числом точек. В рассмотренном выше случае это было уменьшение в два раза. Нетрудно, однако, вывести формулы, сводящие ДПФ с N точками к ДПФ с N/p точками, где p — любое целое число, делящее целое число N. Таким образом, редукция может быть применена не только к ДПФ, у которых N есть степень двойки, но и к ДПФ, у которых N есть любое составное (не простое) число. Заметим еще, что методы быстрого преобразования применимы не только к прямому, но и к обратному преобразованию Фурье.

Поскольку спектральный анализ находит широчайшее применение как при обработке результатов измерений, так и для управления технологическими процессами, оказалось целесообразным для выполнения БПФ строить специализированные процессоры.

9.5.3. Понятие о непрерывном преобразовании Фурье. Если отрезок, на котором задана функция $f(x)$, бесконечен, то ее разложение в ряд Фурье на этом отрезке становится, вообще говоря, невозможным. Спектр такой функции в общем случае не дискретен, а непрерывен. Амплитудно-частотная характеристика

$F(w)$ функции $f(x)$ представима преобразованием Фурье:

$$F(w) = \int_{-\infty}^{\infty} f(x)\, e^{-iwx} dx. \tag{9.31}$$

По этой характеристике исходная функция восстанавливается обратным преобразованием Фурье:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w)\, e^{iwx} dw. \tag{9.32}$$

Если $f(x) = 0$ при $x = 0$, то, очевидно, нижний предел интегрирования в формуле (9.31) можно положить равным нулю. Поскольку в реальных вычислениях верхний предел интегрирования также должен быть заменен конечным числом l (хотя достаточно большим), то становится ясной роль рассмотренного выше дискретного преобразования Фурье как аппроксимации непрерывного случая.

9.5.4. Использование преобразования Фурье для фильтрации шумов. На практике часто встречается задача фильтрации шумов, спектр которых заметно сдвинут относительно спектра основного сигнала. Чаще всего спектр основного сигнала $f(x)$ состоит в основном из низкочастотных гармоник (функция $f(x)$ меняется относительно медленно), а спектр шума $\varphi(x)$, наоборот,— высокочастотных. Наблюдаемый сигнал $g(x) = f(x) + \varphi(x)$ на некотором отрезке $0 \leqslant x \leqslant l$ обычно имеет в таком случае ДПФ с двумя выраженными «горбами» в низкочастотной и высокочастотной областях амплитудно-частотной характеристики, как это показано на рис. 9.5. Отсекая в дискретном преобразовании Фурье $G(w)$ гармоники с частотой $w > a$ (т. е. полагая $G(w) = 0$ при $w > a$) и производя обратное преобразование Фурье для полученной таким образом функции $G(w)$, получим функцию $g(x)$, которая будет достаточно хорошей аппроксимацией полезного сигнала $f(x)$.

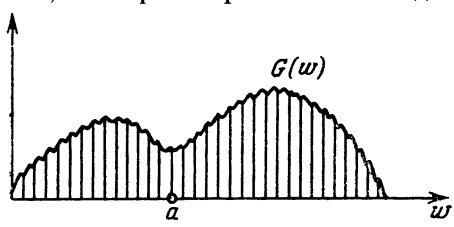

Рис. 9.5

9.6. Системы автоматизированного контроля оборудования

Задача *систем* (*устройств*) *автоматизированного контроля оборудования* (сокращенно САКО или УКО) состоит в том, чтобы автоматизировать измерение и фиксацию значений различного

рода параметров, характеризующих состояние контролируемого оборудования, а также, возможно, и специальную сигнализацию о назревающих аварийных ситуациях или просто о нежелательных отклонениях контролируемых параметров. На ранних стадиях развития автоматики значения каждого контролируемого параметра (как правило, в аналоговой форме) выводились на контрольно-измерительные приборы, установленные обычно на общем пульте (щите). От каждого такого прибора к контролируемой точке протягивались провода (трубки — при использовании гидравлических или пневматических приборов). Приборы обычно снабжались самописцами, записывающими динамику изменения измеряемых параметров в виде кривых (чаще всего на бумажном носителе). Допустимые пределы изменений параметра фиксировались на шкале прибора в виде специальных меток (красных линий) или контактов, включавших предупреждающую или аварийную сигнализацию.

При большом числе измерительных приборов (для сложных систем оно может измеряться сотнями) контрольные щиты становятся громоздкими и дорогими. Кроме того, обилие приборов рассредоточивает внимание оператора, усложняет его работу и приводит зачастую к серьезным ошибкам. Поэтому, хотя описанные системы автоматизированного контроля все еще имеют достаточно широкое распространение, они постепенно вытесняются более совершенными *кибернетическими* системами, основанными на использовании ЭВМ или специализированных дискретных информационно-вычислительных устройств.

Главной внешней отличительной особенностью подобных УКО, которые мы будем называть *цифровыми*, является замена на пульте огромного числа шкал различных измерительных приборов небольшим числом устройств отображения (дисплеев или экранов). На эти устройства могут выводиться значения любого контролируемого параметра, история его изменения, а также, в случае необходимости, значения любых требуемых функций от текущих и прошлых значений параметров, например прогнозные (экстраполируемые) значения параметров, на те или иные моменты времени. Значения параметров и функций, как правило, выводятся в цифровом виде, однако, в случае необходимости, может быть предусмотрена и графическая форма вывода.

Порядок (или, как часто говорят, дисциплина) отображения определяется программой, заложенной в *устройстве управления* системой (универсальной или специализированной ЭВМ). Это может быть, например, вывод в заданные моменты времени (используя встроенные в устройство управления электронные часы, называемые *таймером*). Вне очереди могут выводиться значения параметров, приближающиеся к установленным для них крити-

ческим значениям. Наконец, во всех случаях предусматривается немедленный вывод значения любого параметра или функции по требованию оператора. Для этого оператору достаточно набрать имя требуемого параметра или нажать кнопку на пульте, идентифицирующую этот параметр. В большинстве современных цифровых устройств контроля предусматривается возможность *программной функциональной переориентации* пультовой клавиатуры при изменениях режимов контроля (без изменений в аппаратуре). Можно, например, с помощью изменения программы в устройстве управления изменить параметр, идентифицируемый той или иной кнопкой. Единственное аппаратурное изменение в аппаратуре, производящееся при этом,— замена имени параметра, помещаемого у соответствующей кнопки или непосредственно на ней.

Фиксация значений контролируемых параметров в цифровых ЭВМ УКО осуществляется не на бумаге, а на машинно-ориентированных носителях (чаще всего — на магнитных лентах или гибких магнитных дисках). Благодаря этому исключается процесс подготовки информации для ее последующей обработки. Наличие в цифровых УКО универсального (ЭВМ) или специализированного устройства переработки информации позволяет формировать любые сколь угодно сложные критерии для выделения ситуаций, требующих повышенного внимания оператора.

Например, непрерывно прогнозируя (экстраполируя) значение какого-либо параметра, можно уловить момент, когда до возможного принятия им нежелательного значения остается одна минута или один час. При этом управляющее устройство реализует описанные выше процедуры экстраполяции (с фильтрацией возможных шумов), выдавая при наступлении критической ситуации (вне всякой очереди) установленный сигнал на определенное устройство отображения или включая специальную аварийную сигнализацию (мигание контрольной лампы, звук зуммера или сирены и т. п.).

Сложность и компоновка цифрового УКО зависят от состава и дислокации контролируемого оборудования. Если все контролируемые точки и пульт управления расположены в одном месте (на расстояниях порядка нескольких метров), то сигналы из точек контроля на пульт могут передаваться (по отдельным каналам) в аналоговом виде. Устройство связи с объектом (УСО), осуществляющее аналого-цифровые преобразования, равно как и устройство управления, устанавливается при этом непосредственно у пульта, на котором располагаются клавиатура и устройство отображения. При больших удалениях контрольного пульта от точек контроля целесообразно преобразование в цифровой код осуществлять непосредственно на местах измерений или использовать цифровые датчики. Система передачи инфор-

мации от точек контроля на пульт может быть при этом значительно удешевлена за счет использования локальных концентраторов.

Локальный концентратор представляет собой простейшее цифровое устройство с памятью. К нему отдельными линиями (проводами) может подключаться большое число цифровых датчиков (чаще всего от 32 до 128). Принимая приходящие цифровые коды от этих датчиков (возможно, одновременно), концентратор автоматически приписывает к каждому такому коду номер идентифицирующей этот код линии (или, что то же самое, номер соответствующего датчика). Занумерованные подобным образом коды передаются по направлению к контрольному пункту по одному каналу (проводной линии) без опасности перепутывания точек контроля, к которым относятся соответствующие коды. Передачу поступающей «наверх» информации концентратор может производить либо в порядке ее поступления, либо в соответствии с иной дисциплиной обслуживания датчиков (устанавливаемой заранее). При этом не исключено, что некоторые данные будут просто игнорироваться и не передаваться дальше, другие будут передаваться вне всякой очереди и т. п.

При большом числе точек контроля могут строиться многоуровневые системы концентрации. При этом данные из концентраторов первого уровня передаются не прямо на пульт, а в концентраторы второго уровня, затем (если надо) — в концентраторы третьего уровня и т. д.

При передаче на большие расстояния цифровые сигналы ослабляются и искажаются. Поэтому на приемном конце (на пульте, а возможно, и на концентраторах высших уровней) ставятся специальные *усилительно-формирователь ные устройства*, восстанавливающие величину и форму исходных сигналов. Кроме того, для ввода приходящих сигналов в ЭВМ часто оказывается необходимым согласовать их с принятым в ЭВМ данного класса *интерфейсом ввода-вывода*. Поэтому принятые сигналы могут подвергаться дополнительным электрическим и логическим (форматным) преобразованиям. При преобразовании формата последовательный код (приходящий по одной линии) может быть преобразован в параллельный или (чаще) в последовательно-параллельный код. Как уже отмечалось в предыдущих главах, последовательно-параллельный код (побайтовый с контрольным разрядом) принят в качестве стандартного для вводно-выводного интерфейса ЕС ЭВМ и большинства других современных ЭВМ.

В сложных системах автоматизированного контроля, использующих универсальные ЭВМ, для сопряжения принимаемой информации с магнитным интерфейсом используются специальные *мультиплексоры*. Особенно удобны программируемые мультиплексоры, которые с помощью изменения программы (без каких-либо аппаратных переделок) могут настраиваться на любые

интерфейсы, производя любые заданные преобразования форматов данных. Так, включенный в номенклатуру устройств ЕС ЭВМ программируемый мультиплексор Барс может осуществлять произвольные переформатирования данных, поступающих по 128 двоичным входам, в данные, выдаваемые по 128 двоичным выходам, например, преобразовать 128 последовательных кодов, поступающих одновременно (разряд за разрядом) по всем входным каналам, в сгруппированные по 8 штук и преобразованные в байтовую форму коды, выдаваемые параллельно по 16 байтовым выходным каналам.

Мультиплексор Барс комплектуется разработанными специально для него локальными концентраторами (до 128 штук на один мультиплексор). К каждому из таких концентраторов можно подсоединять до 128 различных цифровых датчиков, так что максимальное число точек контроля даже при одноуровневой концентрации может достигать $128^2 = 16\,384$ в расчете на один мультиплексор. Надо иметь в виду, что при большой частоте опроса датчиков скорость мультиплексора (для Барса 2 млн. операций в секунду) может оказаться недостаточной для обработки (переформатирования) поступающей информации. В этом случае общее число точек контроля, приходящееся на один мультиплексор, должно быть соответственно уменьшено. Вместе с тем, при наличии у ЭВМ, управляющей работой пульта, нескольких входных каналов, к ней может быть подключено одновременно соответственное число мультиплексоров, лишь бы скорость работы ЭВМ оказалась достаточной для переработки всей поступающей информации в нужном темпе.

Описанная сложная структура устройств автоматизированного контроля с использованием сложных мультиплексоров и универсальных ЭВМ употребляется лишь в тех случаях, когда требуется достаточно сложная обработка поступающей информации. В более простых случаях, когда требуется лишь отображение текущих значений контролируемых параметров (особенно при небольшом их числе), пульт может состоять фактически из одного алфавитно-цифрового дисплея с произвольно адресуемой памятью и простейшим мультиплексором, обеспечивающим прием и запись в память поступающей информации. При необходимости автоматического контроля и сигнализации о достижении теми или иными параметрами заданных граничных значений устройства управления пульта дополняются специальными регистрами и конечными автоматами (см. гл. I), осуществляющими периодическую сверку текущих значений контролируемых параметров с хранимыми на регистрах или в памяти их граничными значениями.

Специальные комбинационные схемы или конечные автоматы могут использоваться для выработки и проверки тех или

иных логических условий (важных с точки зрения контроля), которым должны (или которым не должны) удовлетворять значения контролируемых параметров.

Заметим еще, что в системах автоматизированного контроля наряду с автоматическими датчиками могут употребляться различного рода устройства для ручного ввода информации. В системе Барс в качестве таких устройств могут употребляться даже обычные телефонные аппараты. Другим весьма распространенным устройством для ручного ввода является датчик состояния неавтоматизированного оборудования. Он представляет собой просто многопозиционный переключатель (или несколько переключателей), различным позициям которого сопоставляется заранее та или иная информация о состоянии контролируемого оборудования, например, «работа в 3-м режиме», «простой по причине отсутствия энергоснабжения» и т. д.

9.7. Автоматизированные и автоматические системы управления (АСУ)

Автоматизация контроля обычно не является самоцелью, а служит лишь одним из важных средств *управления* контролируемым объектом. Управление состоит в *целенаправленном воздействии* на управляемые параметры системы. Воздействие на управляемые параметры осуществляется с помощью различных технических средств, начиная с простейших (кнопки, рубильники, переключатели, рычаги, рукоятки) и кончая сложными *исполнительными механизмами* с электрическими, пневматическими или гидравлическими приводами. Простейшие средства служат для ручного воздействия, сложные, как правило, допускают управление с помощью сигналов, передаваемых с пульта оператором или автоматическим управляющим устройством (в частности, управляющей ЭВМ).

При использовании цифровых управляющих сигналов возможны два основных способа их воздействия на исполнительные механизмы. В первом способе требуемая величина управляемого параметра передается целиком на исполнительный механизм, где с помощью специального цифро-аналогового преобразователя (чаще всего преобразователя цифра — угол поворота вала) трансформируется в требуемую *установку* параметра. Во втором способе информация в исполнительный механизм передается отдельными импульсами (двоичными сигналами), каждый из которых вызывает определенное *элементарное действие* (переключение реле, передвижение шагового двигателя на один шаг вперед или назад и т. д.). При близком расположении точек воздействия и управляющего устройства (пульта) цифро-аналоговые преобразования могут выполняться на централизованном устройстве свя-

зи с объектом и передаваться на точки воздействия в аналоговом виде.

Передача управляющих сигналов от системы управления к управляемому объекту носит название *прямой связи*, а передача информации в обратном направлении — *обратной связи* (рис. 9.6). Обратную связь в системах управления осуществляют разобранные выше системы автоматизированного контроля.

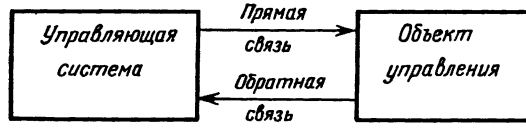

Рис. 9.6

Даже при достаточно высоких уровнях автоматизации управления как в составе самой управляющей системы, так и в каналах прямой и обратной связи могут оказаться люди (операторы, диспетчеры, лица, осуществляющие ручной ввод информации, и т. д.). Такие системы мы будем называть *автоматизированными*, в отличие от *автоматических систем*, полностью исключающих участие человека во всех звеньях управления. Фактически подобная частичная автоматизация управления реализуется в большинстве систем автоматизированного контроля. Результаты работы таких систем, как правило, используются сидящими за пультами диспетчерами (операторами) для выработки управляющих воздействий на контролируемый объект. Такие воздействия — команды диспетчера — могут быть переданы голосом по громкоговорящей связи и исполняться вручную. Следующий уровень автоматизирует канал прямой связи: команды набираются диспетчером на пульте и автоматически передаются на нужные точки воздействия на объекте управления.

При наличии на пульте достаточно мощного вычислительного устройства оно может быть использовано для выработки управляющих воздействий. В так называемом *режиме советчика* машина выдает эти воздействия на одно из пультовых устройств отображения в качестве советов диспетчеру (оператору). Оценка этих советов, решение на их использование и передача на объект управления остается при этом за человеком. Устранив диспетчера в качестве передаточного звена, замыкают весь контур управления на машину, получая тем самым (при отсутствии датчиков ручного ввода) полностью автоматическую систему управления. При этом диспетчер может быть сохранен для наблюдения за работой системы и, возможно, для временного перехода на ручное управление при выходе системы из строя.

При автоматизации управления относительно простыми *сосредоточенными* (расположенными в малом объеме) объектами широко применялись и продолжают применяться до сих пор аналоговые вычислительные устройства. Появление дешевых микропроцессоров и микро-ЭВМ приводит к их постепенному вытеснению цифровыми устройствами, главными преимуществами которых являются высокая точность и универсальность. Последнее свойство особенно ценно, поскольку оно позволяет менять алгоритмы управления без переделок аппаратуры. В условиях же непрерывного ускорения научно-технического прогресса с подобными ситуациями приходится сталкиваться все чаще.

Что же касается автоматизации управления сложными *распределенными* объектами, то здесь преимущества цифровой техники в большинстве случаев делают ее использование фактически единственно возможным решением. Заметим, что для сложных объектов часто приходится строить многоуровневые системы автоматизации. При этом отдельными единицами оборудования могут управлять локальные системы. Задачей же систем более высокого уровня является выработка относительно небольшого числа установок, определяющих режимы работы низовых систем, с целью согласовать их совместную работу по тому или иному комбинированному критерию. В современных многоуровневых системах управления сложными технологическими процессами могут работать одновременно десятки управляющих ЭВМ.

Различают несколько основных видов автоматизированного управления. В так называемом *программном управлении* обратная связь отсутствует. Простейшим устройством такого рода является обычный автоматический светофор, переключающий свет через заданные промежутки времени независимо от ситуации, реально складывающейся на регулируемом перекрестке. Более сложные случаи — станки с числовым программным управлением (ЧПУ) и программно-управляемые роботы-манипуляторы — будут описаны ниже.

Один из наиболее простых видов автоматического управления с обратной связью имеет целью поддержание в меняющихся условиях заданных постоянных значений тех или иных параметров (уровня воды в котле, скорости вращения вала, температуры и влажности воздуха в помещении и т. п.). Такого рода задачи задолго до появления ЭВМ и даже аналоговых машин решались (в простых случаях) несложными устройствами, называемыми *автоматическими регуляторами*. Широкую известность получил, например, имеющий более чем двухвековую историю *регулятор Уатта*, использовавшийся для поддержания заданного числа оборотов вала паровых машин.

Более сложными являются задачи *оптимального управления*, насчитывающие сегодня много разновидностей. Самая простая из

них — задача *экстремального регулирования*, целью которой является поддержание экстремального (максимального или минимального) значения одного из регулируемых параметров. В более сложном случае речь может идти о поддержании (в меняющихся условиях) экстремального значения некоторой заданной функции от регулируемых параметров. Задача систем, *оптимальных по быстродействию*, состоит в том, чтобы найти управление (последовательность управляющих воздействий), которое обеспечивает наиболее быстрый перевод объекта управления из одного заданного состояния в другое. При этом приходится учитывать различные ограничения, накладываемые на управление свойствами объекта или особенностями системы управления.

В теории и практике автоматизированных систем управления (АСУ) принято различать *автоматизированные системы управления технологическими процессами* (АСУТП) и *автоматизированные системы организационного управления* (АСОУ). Первые имеют своей задачей текущее управление работой оборудования, а вторые — весь комплекс задач управления человеческими коллективами (в том числе долгосрочное планирование). Поскольку в современном производстве задачи управления людьми и оборудованием тесно связаны между собой, на практике далеко не всегда легко провести разграничительную линию между этими двумя типами систем. Например, задача управления станочной неавтоматизированной линией, которая должна рассматриваться как задача АСОУ, превращается в задачу АСУТП при замене станков с ручным управлением на станки с ЧПУ, хотя содержательная часть задач планирования и текущего управления линией (а не отдельными станками) при этом практически не меняется.

Поэтому в дальнейшем при описании математических задач управления мы будем использовать несколько иной принцип классификации, относя к классу технологических систем все системы с пренебрежимо малой задержкой влияния входных сигналов на выходные параметры системы. Тем самым задачи планирования (особенно долгосрочного) будут автоматически отнесены к задачам организационного управления. Однако при таком подходе к классу организационных систем следует относить системы управления технологическими процессами с длительными циклами производства (в химии, металлургии и других областях). Вообще, учитывая историю развития теории автоматического регулирования, следует считать, что классические технологические системы управления имеют дело с относительно быстропротекающими процессами в отдельных единицах оборудования, а организационные системы — с более медленными процессами взаимодействия людей и отдельных единиц оборудова-

ния в сложных технических комплексах (цех, завод; отрасль, все народное хозяйство в целом).

Заметим также, что, говоря о технологических процессах в этой главе, мы имеем в виду процессы материального производства. Развитие машинной информатики расширило и породило заново другие виды технологии, а именно *технологии обработки информации*. Вопросы автоматизации информационных технологий (системы обработки данных) будут рассматриваться в следующей главе.

9.8. Автоматизация испытаний

Задача испытаний, независимо от того, имеет ли она дело с относительно простым изделием или со сложным объектом, во многом сходна с задачей управления. Как и в задаче управления, к объекту испытаний применяются те или иные управляющие воздействия (передаваемые по каналам прямой связи от автоматизированной системы или человека-испытателя). Точно так же регистрируются (через канал обратной связи) и обрабатываются результаты этих воздействий. Поэтому системы для автоматизации испытаний строятся в основном по тем же принципам, что и системы автоматизированного управления.

Имеются, однако, по крайней мере два существенных различия. Во-первых, при автоматизации управления управляющие воздействия направлены на то, чтобы обеспечить в каком-то смысле наилучшие условия функционирования управляемого объекта. При автоматизации же испытаний, наоборот, стремятся, как правило, поставить испытуемый объект в наихудшие условия, не исключая в случае необходимости искусственного создания аварийных ситуаций. Второе отличие состоит в том, что результаты испытаний во многих случаях не требуют мгновенной обработки (в темпе проведения испытаний). Они могут поэтому фиксироваться на тех или иных машинных носителях и переноситься в вычислительный центр для последующей обработки.

В качестве примера подобной организации автоматизации испытаний можно указать систему Темп ЭК, предназначенную для автоматизации летных испытаний самолетов. Заранее рассчитанные режимы испытаний приводятся в исполнение летчиком-испытателем во время испытательного полета. В течение всего полета сотни датчиков через специальную систему сбора информации фиксируют свои показания на магнитных лентах, которые после окончания полета переносятся в стационарный ВЦ для автоматической обработки на универсальных ЭВМ. Режимы следующего испытательного полета рассчитываются в соответствии с результатами этой обработки.

9.9. Математическая постановка задач автоматического управления

Состояние объекта управления A мы будем задавать конечным множеством меняющихся во времени параметров $y_1(t), \ldots, y_n(t)$, одни из которых в общем случае могут быть непрерывными, а другие — дискретными. Для простоты обозначений это множество скалярных параметров удобно заменить одним векторным параметром $y(t)$. Объект управления может подвергаться внешним воздействиям двух видов. Одни из них, которые мы назовем *возмущениями*, представляют собой заданные функции времени $u_1(t), \ldots, u_m(t)$, не подверженные управлению. Другие представляют целенаправленные *управляющие воздействия*, формируемые *системой управления* B. Параметры управления $v_1(t), \ldots, v_k(t)$ вырабатываются системой B в результате обработки результатов $z_1(t), \ldots, z_l(t)$ измерения состояния объекта и, может быть, также возмущений $u_1(t), \ldots, u_m(t)$, измерительной системой C и передаются в качестве управляющих воздействий на объект управления.

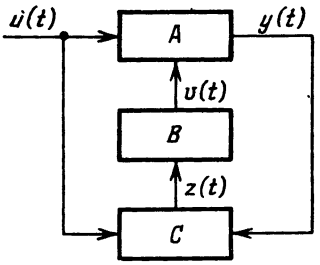

Рис. 9.7

Описанные взаимодействия объекта A и системы управления B изображены на рис. 9.7. Предположим для простоты, что все взаимодействия в описанной схеме реализуются мгновенно (без задержек), а ее поведение изучается на интервале времени от $t=0$ до $t>0$. Введем также в расмотрение вектор $x(t)=(x_1(t), \ldots, x_p(t))$, характеризующий состояние системы управления B. Тогда функционирование системы, изображенной на рис. 9.7, может быть описано системой соотношений:

$$y(t)=F(y(0),\ u(t),\ v(t),\ t), \quad t\geqslant 0, \tag{9.33}$$

$$x(t)=G(x(0),\ z(t),\ t), \quad t\geqslant 0, \tag{9.34}$$

$$z(t)=f(y(t),\ u(t)), \quad t\geqslant 0, \tag{9.35}$$

$$v(t)=g(x(t)), \quad t\geqslant 0. \tag{9.36}$$

Если правые части третьего и четвертого соотношений, характеризующие измерительную и исполнительную части системы управления, могут рассматриваться как обычные (точечные) функции, то с первыми двумя дело обстоит иначе. В самом деле, нетрудно видеть, что значения $y(t)$ и $x(t)$ при любом заданном значении $t=a$ зависят не только от значений $u(t)$, $v(t)$ и $z(t)$ при $t=a$, но, вообще говоря, и от их значений во все пред-

шествующие моменты времени ($0 \leqslant t \leqslant a$). Иными словами, F и G (в отличие от f и g) представляют собой не обычные функции, а функционалы, аргументами которых являются не значения функций $u(t)$, $v(t)$, $z(t)$ в момент времени t, а сами эти функции.

Эти функционалы представляют собой обобщения на общий (непрерывно-дискретный) случай автоматных отображений, рассматривавшихся нами в гл. I. Поскольку операции с подобными функционалами являются достаточно сложными, предпочитают другие формы записи соотношений, характеризующих работу систем A и B. Идея здесь по существу та же, что и в случае автоматов: выразить текущее состояние объекта с памятью (характеризуемое набором возможных значений параметров состояния объекта) через предыдущее (а не через начальное) состояние. В дискретном случае (при наличии дискретного времени) понятие предыдущего состояния естественно определено. Что же касается непрерывного случая, то, как известно, средством, связывающим текущее значение (непрерывного) параметра $x(t)$ в некоторой точке t с его предыдущими значениями, является производная этого параметра по времени в этой точке (точно, производная слева, т. е. $\lim\limits_{\Delta t \to 0} \frac{\Delta x}{\Delta t}$ при $\Delta t < 0$). Поэтому вместо векторных функциональных уравнений в непрерывном случае обычно пользуются системами обыкновенных дифференциальных уравнений, задающими так называемые *динамические системы.*

Предположим для простоты, что измеряемые параметры $z_i(t)$ совпадают с объединением параметров $y_j(t)$ и $u_j(t)$, а $v(t) = x(t)$. Тогда система, изображенная на рис. 9.7, представима в виде объединения двух динамических систем:

$$\frac{dy(t)}{dt} = \varphi(y(t), u(t), v(t), t), \tag{9.37}$$

$$\frac{dv(t)}{dt} = \psi(y(t), u(t), v(t), t) \tag{9.38}$$

с дополнительно заданными начальными условиями

$$y(t)|_{t=0} = y(0), \quad v(t)|_{t=0} = v(0). \tag{9.39}$$

При этих условиях из уравнений (9.37) и (9.38) можно найти состояния управляемого объекта и управляющей системы как функции возмущений $u(t)$ и времени t:

$$y(t) = r(u(t), t), \quad v(t) = s(u(t), t). \tag{9.40}$$

Заметим, что функции φ, ψ, r, s являются обычными (векторными) функциями. Если управляющая система B не имеет памяти, то управление $v(t)$ вместо соотношения (9.38) может быть

задано более простым явным соотношением

$$v(t)=\xi(y(t),\ v(t),\ t). \tag{9.41}$$

После подстановки в (9.37) функции ξ вместо $v(t)$ можно получить уравнения управляемого объекта A в *свободной форме* (не содержащей в явном виде управления $v(t)$):

$$\frac{dy(t)}{dt}=\lambda(y(t),\ u(t),\ t). \tag{9.42}$$

Для иллюстрации введенных определений рассмотрим простой пример. Предположим, что имеется запас y некоторого материала, потребляемого со скоростью u и производимого со скоростью v (с целью пополнения запаса). Скорость изменения запаса dy/dt будет равна разности скоростей его пополнения и расходования:

$$dy/dt=v-u. \tag{9.43}$$

Расход $u=u(t)$ представляет собой некоторую заданную функцию времени и не является объектом управления. Что же касается скорости производства v, то ею можно управлять в соответствии с тем или иным законом. Естественно желание выбрать такой закон управления, который обеспечил бы поддержание некоторого запаса $0<a<y$ при любом расходе $u(t)$. Этому условию удовлетворяет закон

$$v=u-k(y-a), \tag{9.44}$$

где k — некоторая положительная константа, а $y(0)=y_0\geqslant a$.

После внесения управления система (9.43) преобразуется в свободную динамическую систему, задаваемую уравнением

$$dy/dt=-k(y-a). \tag{9.45}$$

Решением этого уравнения является функция $y(t)=a+ce^{-kt}$, которая при любом начальном условии $y(0)=y_0$ стремится к a при $t\to\infty$, причем $c\geqslant 0$ при $y_0\geqslant a$, так что $y(t)\geqslant a$ при всех $t\geqslant 0$.

В теории управления динамическими системами представляют интерес два класса задач, называемых обычно *задачами анализа* и *синтеза*. Задача анализа применяется к свободным динамическим системам и состоит в изучении различных аспектов их поведения. Задача синтеза ставится применительно к системам, имеющим хотя бы один управляющий параметр v_i. Смысл этой задачи состоит в нахождении такого закона управления, который обеспечит некоторое наперед заданное поведение системы.

Простейшая задача анализа (задача Коши) состоит в том, чтобы при заданных начальных условиях $y_1(0)=a_1$, $y_2(0)=a_2$, ...

..., $y_n(0)=a_n$ определить траекторию системы, т. е. систему функций $y_1(t)$, $y_2(t)$, ..., $y_n(t)$, удовлетворяющих уравнениям, которыми задается рассматриваемая динамическая система, и имеющих заданные начальные значения.

Как доказывается в теории, задача Коши для системы обыкновенных дифференциальных уравнений с непрерывными правыми частями всегда имеет единственное решение. Имеется большое число методов фактического численного решения задачи Коши, обеспечивающих нахождение последовательных точек искомой траектории с любой заданной точностью. В основе большинства этих методов лежит следующая идея: исходя из начальной точки $M(0)=(y_1(0), y_2(0), \ldots, y_n(0))$, строится последовательно ряд точек $M(t_i)$ искомой траектории, где $t_i=t_{i-1}+\Delta t$ $(i=1, 2, \ldots, t_0=0)$, причем для вычисления координат $y_i(t_i)$ очередной точки используют координаты $y_j(t_{j-k})$ построенных ранее точек и значения производных $y_j'(t_{j-k})$ в этих точках, которые могут быть получены из уравнений, задающих систему.

Простейший (хотя и очень несовершенный) метод основывается на использовании приближенного соотношения $y_i(t+\Delta t)\approx \approx y_i(t)+y_i'(t)\Delta t$. Пусть, например, в уравнении (9.45) $k=1$, $a=1$. Построим несколько последовательных точек траектории с начальным условием $y(0)=0$ и шагом $\Delta t=0{,}1$.

Записывая уравнение $y'=1-y$, получаем

$$y'(0)=1-y(0)=1,$$

откуда

$$y(0,1)\approx y(0)+y'(0)\Delta t=0+1\cdot 0{,}1=0{,}1;$$

$$y(0{,}2)\approx y(0{,}1)+y'(0{,}1)\Delta t=0{,}1+(1-0{,}1)\cdot 0{,}1=$$

$$=0{,}1+0{,}09=0{,}19;$$

$$y(0{,}3)=y(0{,}2)+y'(0{,}2)\Delta t=0{,}19+(1-0{,}19)\cdot 0{,}1\approx 0{,}27.$$

Точное решение имеет вид

$$y=1-e^{-t},$$

что дает

$$y(0{,}1)\approx 0{,}095;\ y(0{,}2)\approx 0{,}181;\ y(0{,}3)\approx 0{,}260.$$

Повышение точности при приближенном решении требует уменьшения шага Δt или перехода к более точным методам (Рунге — Кутта, Адамса и др.).

Вторая важнейшая задача анализа свободных динамических систем — анализ их устойчивости. Общий смысл понятия устойчивости состоит в том, что малые изменения начальных условий и входных функций не должны приводить к большим изменениям определяемых ими траекторий.

В теории авторегулирования особое значение приобрела так называемая *асимптотическая устойчивость*. Суть ее состоит в следующем. Пусть $M(t)=(y_1(t),\ y_2(t),\ \ldots,\ y_n(t))$ — траектория, заданная некоторыми начальными условиями $y_i(0)=a_i$ $(i=1,\ 2,\ \ldots,\ n)$ или, иначе, начальной точкой $M_0=(a_1,\ \ldots,\ a_n)$. Давая малые изменения координатам $a_1,\ a_2,\ \ldots,\ a_n$, построим новую начальную точку $\bar{M}_0$ и определяемую ею траекторию (переменную точку) $\bar{M}(t)$. Говорят, что система *локально* асимптотически устойчива в точке M_0, если расстояние между точками $\bar{M}(t)$ и $M(t)$ стремится к нулю при $t\to\infty$. Термин «локальный» применительно к устойчивости означает, что устойчивость имеет место лишь по отношению к малым изменениям начальных условий.

Если устойчивость сохраняется при любых изменениях начальных условий, то говорят об устойчивости системы *в целом*; в этом случае в пределе при $t\to\infty$ все траектории, независимо от своих начальных точек, сливаются в одну — система как бы «забывает», из какого начального положения она начала развиваться.

9.9.1. Линейные динамические системы. Обе задачи анализа — определение траекторий и анализ на устойчивость — получают исчерпывающее решение для систем, определяемых линейными дифференциальными уравнениями с постоянными коэффициентами. В общем виде такая система может быть задана следующим образом:

$$\begin{aligned}
\frac{dy_1}{dt} &= a_{11}y_1 + a_{12}y_2 + \ldots + a_{1n}y_n + u_1(t),\\
\frac{dy_2}{dt} &= a_{21}y_1 + a_{22}y_2 + \ldots + a_{2n}y_n + u_2(t),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
\frac{dy_n}{dt} &= a_{n1}y_1 + a_{n2}y_2 + \ldots + a_{nn}y_n + u_n(t).
\end{aligned} \tag{9.46}$$

В теории линейных систем с постоянными коэффициентами особое значение имеет определитель

$$D(s)=\begin{vmatrix}
s-a_{11} & -a_{12} & -a_{13} & \ldots & -a_{1n}\\
-a_{21} & s-a_{22} & -a_{23} & \ldots & -a_{2n}\\
\ldots & \ldots & \ldots & \ldots & \ldots\\
-a_{n1} & -a_{n2} & -a_{n3} & \ldots & s-a_{nn}
\end{vmatrix},$$

представляющий собой полином степени n от s. Он называется обычно *характеристическим полиномом* рассматриваемой системы.

Общее решение системы (9.46) записывается в виде

$$y_i(t)=c_{i1}\varphi_{i1}(t)+c_{i2}\varphi_{i2}(t)+\ldots+c_{in}\varphi_{in}(t)+\\ +\psi_{i1}(t)+\psi_{i2}(t)+\ldots+\psi_{in}(t),\quad t=1,\ 2,\ \ldots,\ n.$$

Каждая из функций $\varphi_{ik}(t)$ соответствует определенному корню характеристического полинома $D(s)$, а каждая функция ψ_{il} — свободному члену $u_l(t)$. Вещественному корню $s=r$ соответствует функция $\varphi(t)=e^{rt}$, а паре комплексных корней $s=p\pm iq$ — пара функций $\varphi(t)$: $e^{pt}\cos qt$ и $e^{pt}\sin qt$. Функции $u_l(t)=e^{rt}$ соответствует функция $\psi_{il}(t)=be^{rt}$, функциям $\sin px$, $\cos px$ или $a\sin px+b\cos px$ соответствует функция $A\sin px+B\cos px$; если $u_l(t)=a_0t^k+a_1t^{k-1}+\ldots+a_n$, то $\psi_{il}(t)=A_0t^k+A_1t^{k-1}+\ldots$ $\ldots+A_n$ и т. д. Если функция $u_l(t)$ есть сумма функций указанного вида, то и $\psi_{il}(t)$ будет представляться в виде соответствующей суммы.

Наконец, нужно иметь в виду, что все функции $\varphi_{ik}(t)$ должны быть различны как между собой, так и с любой из функций $\psi_{il}(t)$. Если это не выполняется, то различие достигается умножением соответствующих функций φ_{ik} в первом случае и функций $\psi_{il}(t)$ — во втором на множители вида t^m с минимальной возможной степенью m, способной обеспечить требуемое различие.

В качестве примера рассмотрим систему

$$x'=x+y+e^t,\quad y'=y-e^t.$$

Ее характеристический полином $D(s)$,

$$\begin{vmatrix} s-1 & -1 \\ 0 & s-1 \end{vmatrix}=(s-1)^2,$$

имеет двойной корень $s=1$. Поскольку функции φ_{ik} должны строиться для каждого корня, то в данном случае их должно быть две. Для того чтобы обеспечить их различие, умножаем одну на t. Получаем две функции: e^t и te^t. Свободные члены e^t и $-e^t$ снова порождают функции ψ_{il} вида be^t. Между собой они могут совпадать, однако необходимо обеспечить их различие по отношению к уже построенным функциям φ (постоянные коэффициенты при установлении сходства и различия во внимание не принимаются). Для этой цели достаточно умножить их на t^2. Поскольку коэффициент b неизвестен, обе функции ψ можно объединить в одну.

Итак, решение системы можно искать в виде

$$x=ae^t+bte^t+ct^2e^t,\quad y=de^t+fte^t+gt^2e^t,$$

где a, b, c, d, f, g — некоторые постоянные коэффициенты. Для их определения подставляем найденные значения x и y в уравнения:

$$ae^t+be^t+bte^t+ct^2e^t+2cte^t=ae^t+ct^2e^t+ct^2e^t+de^t+fte^t+$$

$$+gt^2e^t+e^t,\ de^t+fe^t+fte^t+gt^2e^t+2gte^t=de^t+fte^t+gt^2e^t-e^t.$$

Приравнивая коэффициенты при одинаковых членах, получаем систему уравнений:

$$a+b=a+d+1, \quad d+f=d-1,$$
$$b+2c=b+f, \quad f+2g=f,$$
$$c=c+g, \quad g=g,$$

откуда $g=0$; $f=-1$; $c=-1/2$; $d=b-1$. Коэффициенты a и b могут оставаться произвольными.

Общее решение системы можно представить в виде

$$x=(a+bt-t^2/2)e^t, \; y=(b-1-t)e^t.$$

Значения коэффициентов a и b можно определить из начальных условий. Пусть $y(0)=y_0$ и $x(0)=x_0$. Тогда $a=x_0$, $b-1=y_0$, $b=1+y_0$. Таким образом, траектория с начальной точкой $\mu_0(x_0, y_0)$ задается уравнениями

$$x=[x_0+(1+y_0)t-t^2/2]e^t, \quad y=(y_0-t)e^t.$$

Вопрос об асимптотической устойчивости линейных систем с постоянными коэффициентами получает исчерпывающее решение: для асимптотической устойчивости такой системы в целом необходимо и достаточно, чтобы все корни ее характеристического полинома $D(s)$ имели отрицательные вещественные части. В только что рассмотренном примере вещественные части обоих корней полинома $D(s)$ равны $+1$, и потому асимптотическая устойчивость не имеет места.

Для уравнения (9.45) характеристический полином $D(s)$ равен, как нетрудно проверить, $s+k$. Единственный корень $s=-k$ этого полинома отрицателен (так как $k>0$), поэтому данная система обладает свойством устойчивости в целом. Ее решение, в соответствии с описанными выше правилами, следует искать в виде $y=be^{-kt}+c$. После подстановки в уравнение (9.45) получаем

$$-kbe^{-kt}=-k(be^{-kt}+c-a),$$

откуда $c=a$. Полагая $t=0$, находим $y_0=b+c=b+a$, откуда $y_0-a=b$. Общее решение системы записывается в виде

$$y=(y_0-a)e^{-kt}+a.$$

При $t\to\infty$, независимо от начального значения y_0, имеем $y\to a$, что наглядно демонстрирует свойство асимптотической устойчивости системы (уравнения) в целом.

Переходя к задачам синтеза, остановимся прежде всего на классической задаче теории автоматического регулирования, а именно на задаче построения такого закона управления, при котором рассматриваемая динамическая система оказывается ло-

кально асимптотически устойчивой. Один из наиболее часто встречающихся частных случаев этой задачи заключается в том, чтобы поддерживать с определенной степенью точности заданные значения основных переменных: $y_1(t) \approx a_1$, $y_2(t) \approx a_2$, ..., $y_n(t) \approx a_n$.

Как и задачи анализа, задачи синтеза наиболее полно изучены для линейных систем с постоянными коэффициентами. Удобным математическим аппаратом в этом случае является так называемое *преобразование Лапласа*. Суть этого преобразования состоит в том, что для любой непрерывной функции $y(t)$, определенной для значений $t > 0$, может быть определено ее изображение (по Лапласу) $Y(s)$, задаваемое формулой

$$Y(s) = \int_0^\infty e^{-st} y(t)\, dt. \tag{9.47}$$

По изображению всегда можно восстановить исходную функцию, используя формулу

$$y(t) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{st} Y(s)\, ds, \tag{9.48}$$

где a — любая неотрицательная константа, превосходящая вещественные части всех особых точек (точек разрыва) функции $Y(s)$.

Применение преобразования Лапласа сильно упрощается тем, что может быть построен специальный словарь для наиболее употребительных функций и операций, позволяющий в большинстве случаев выполнять преобразования без применений формул (9.47) и (9.48). Выпишем для примера несколько строк из этого словаря:

$$y_1(t) + y_2(t) \leftrightarrow Y_1(s) + Y_2(s);$$

$$ay(t) \leftrightarrow aY(s);$$

$$y'(t) \leftrightarrow -y(0) + sY(s);$$

$$y''(t) \leftrightarrow -y'(0) - sy(0) + s^2 Y(s);$$

$$a \leftrightarrow a/s;$$

$$t^n \leftrightarrow \frac{(n+1)!}{s^{n+1}};$$

$$e^{\alpha t} \leftrightarrow 1/(s-a);$$

$$\sin at \leftrightarrow a/(s^2 + a^2);$$

$$\cos at \leftrightarrow s/(s^2 + a^2), \quad a = \text{const}.$$

Используя преобразование Лапласа и обратное ему преобразование, получаем еще один способ решения систем обыкновенных линейных дифференциальных уравнений с постоянными коэффициентами. Например, применяя преобразование Лапласа к уравнению (9.45), находим $-y(0)+sY(s)=-k(Y(s)-a/s)$, откуда $Y(s)=ak/(s(s+k))+y(0)/(s+k)$. Поскольку $ak/(s(s+k))=$ $=a/s-a/(s+k)$, то, применяя обратное преобразование, получаем

$$y(t)=a-ae^{-kt}+y(0)e^{-kt}=a+[y(0)-a]e^{-kt},$$

что совпадает с решением, найденным выше.

Применим преобразование Лапласа к общей системе обыкновенных линейных дифференциальных уравнений с постоянными коэффициентами:

$$\begin{gathered}(s-a_{11})Y_1(s)-a_{12}Y_2(s)-\ldots-a_{1n}Y_n(s)=u_1(s)+y_1(0),\\ -a_{21}Y_1(s)+(s-a_{22})Y_2(s)-\ldots-a_{2n}Y_n(s)=u_2(s)+y_2(0),\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ -a_{n1}Y_1(s)-a_{n2}Y_2(s)-\ldots+(s-a_{nn})Y_n(s)=u_n(s)+y_n(0).\end{gathered}$$

Решая полученную систему линейных алгебраических уравнений, придем к следующим выражениям:

$$Y_i(s)=[u_1(s)+y_1(0)]\frac{N_{i1}(s)}{D(s)}+[u_2(s)+y_2(0)]\frac{N_{i2}(s)}{D(s)}+\ldots$$
$$\ldots+[u_n(s)+y_n(0)]\frac{N_{in}(s)}{D(s)},\quad i=1,2,\ldots,n,$$

где $D(s)$ — характеристический полином рассматриваемой системы, $N_{ij}(s)$ — полиномы от s степени на единицу меньше, чем $D(s)$, т. е. $n-1$, представляющие собой алгебраические дополнения элементов j-го столбца определителя.

Функция $N_{ij}(s)/D(s)=F_{ij}(s)$ называется *передаточной функцией* рассматриваемой системы по отношению к входному параметру u_j и выходной основной переменной y_i. Эта функция имеет определенный физический смысл. Если принять

$$u_j(t)=\begin{cases}\infty & \text{при } t=0,\\ 0 & \text{при } t\neq 0,\end{cases}\qquad \int_0^\infty u_i(t)\,dt=1,$$

то, как нетрудно видеть, $u_j(s)=1$. Полагая $u_k(t)=0$ при $k\neq j$ и $y_1(0)=y_2(0)=\ldots=y_n(0)=0$, получаем

$$Y_i(s)=F_{ij}(s)\cdot 1=F_{ij}(s).$$

Иными словами, передаточная функция $F_{ij}(s)$ может трактоваться как реакция системы по i-му выходу на единичный импульс (δ-функцию), подаваемый на j-й вход при нулевых начальных

условиях и отсутствии входных сигналов по всем остальным входам.

Если при нулевых начальных условиях положить $u_j(t) = a_j = \text{const}$, $u_k(t) = 0$ при $k \neq j$, то система дает постоянные сигналы $y_i(t) \equiv b_i$ по каждому выходу. Отношение $b_i/a_j = p_{ij}$ называется *коэффициентом усиления системы по паре* j-й вход — i-й выход. Можно показать, что $b_i/a_j = F_{ij}(0)$.

Аналогично, если при нулевых начальных условиях принять $u_j(t) = a_j \sin wt$, а $u_k(t) \equiv 0$ при $k \neq j$, на выходах появятся сигналы $y_i(t) = b_i \sin(w + d_i)t$. Отношение $b_i(\cos d_i + i \sin d_i)/a_j = p_{ij}(w)$ называют (комплексным) *коэффициентом усиления при частоте* w. Можно показать, что $p_{ij}(w) = F_{ij}(w_i)$. Функция $F_{ij}(w_i)$ называется *частотной характеристикой* системы по паре j-й вход — i-й выход. Задание этой характеристики вместе с коэффициентом усиления p_{ij} определяет соответствующую передаточную функцию системы.

Приведенные результаты показывают, каким образом может быть экспериментальным путем определена передаточная функция линейной системы в случае, когда соответствующая система уравнений (9.46) неизвестна. Для решения задачи о синтезе линейной системы управления представляют с помощью передаточных функций как объект управления, так и регулятор. Предположим для простоты, что речь идет о системах с одним входом и одним выходом.

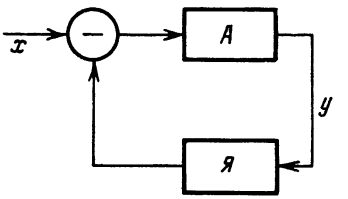

Рис. 9.8

Пусть $F_1(s)$ — передаточная функция объекта: $X(s)$ и $Y(s)$ изображение (по Лапласу) его входной и выходной функций. Вход регулятора, задаваемого передаточной функцией $F_2(s)$, присоединяем к выходу объекта, а вырабатываемый им сигнал $F_2(s)Y(s)$ вычитаем из входного сигнала $X(s)$. Такое присоединение регулятора называется *обратной связью* (от выхода к входу) (рис. 9.8). Уравнения (в изображениях) для полученной таким образом новой системы запишем в виде

$$Y(s) = F_1(s)[X(s) - F_2(s)Y(s)],$$

откуда

$$\frac{Y(s)}{X(s)} = \frac{F_1(s)}{1 + F_1(s)F_2(s)} = F_3(s),$$

где $F_3(s)$ — передаточная функция новой системы.

Для обеспечения устойчивости достаточно подобрать функцию так, чтобы нули знаменателя вновь полученной передаточной функции имели отрицательные вещественные части. Например, если $F_1(s) = 1/(s - a)$, $a > 0$, то, выбирая $F_2(s) =$

$=c/(s+b)$, получаем в качестве $F_3(s)$ функцию

$$\frac{1/(s-a)}{1+c/[(s-a)(s+b)]}=\frac{s+b}{s^2+(b-a)s+c-ab}.$$

Легко видеть, что при $b>a$ и $c>ab$ вещественные части обоих корней знаменателя будут отрицательны. Таким образом, обратная связь с передаточной функцией $F_2(s)$ превращает первоначально неустойчивую систему в устойчивую.

Более сложным образом ставится и решается задача синтеза в том случае, когда требуется сконструировать регулятор, обеспечивающий не только устойчивое, но и целенаправленное и тем более оптимальное регулирование. Одна из наиболее известных постановок задачи целевого регулирования заключается в следующем. Пусть на множестве основных переменных $y_1, y_2, \ldots, y_n$ и входных параметров $u_1, u_2, \ldots, u_m$ задана некоторая целевая функция $f(y_1, \ldots, y_n; u_1, \ldots, u_m)$. Требуется сконструировать регулятор, который обеспечил бы поддержание при изменениях входных параметров постоянного (с некоторой точностью) значения целевой функции f. Пример подобной задачи дает рассмотренная выше задача управления запасами. Управление, заданное соотношением (9.44), обеспечивает (при $y_0=a$) равенство a целевой функции $f(y, u)=y$.

Одна из известных постановок задач в теории оптимального управления состоит в том, чтобы обеспечить (при изменениях входных параметров u_1) минимальное или максимальное значение целевой функции. Например, пусть в той же задаче управления запасами необходимо найти управление, обращающее в минимум функцию

$$f=by+cu/y,$$

первый член которой оценивает расходы на содержание запаса y, а второй оценивает потери, которые могут быть вызваны нехваткой запасов при внезапном росте спроса u. Минимум этой функции (при заданном u) находится из уравнения

$$b-cu/y^2=0,$$

откуда $y=\sqrt{cu/b}$. Подстановка в уравнение (9.43) дает

$$y'=\frac{\sqrt{c/b}}{2\sqrt{u}}\cdot u'=v-u, \text{ или } v=u+\frac{cu'}{2\sqrt{bcu}}.$$

Для обеспечения устойчивости регулирования добавляем член $-ky$. Требуемый закон управления запишется в виде

$$v=\frac{cu'}{2\sqrt{bcu}}+u-ky.$$

Перепишем уравнение (9.43):

$$\frac{dy}{dt} = \frac{cu'}{2\sqrt{bcu}} - ky;$$

его общее решение имеет вид

$$y = Ae^{-kt} + \sqrt{cu/b},$$

что и обеспечивает асимптотическое стремление к оптимальному регулированию. Управление будет оптимальным в целом, если $y_0 = \sqrt{cu_0/b}$.

Хотя рассмотренные примеры относятся к числу линейных систем с постоянными коэффициентами, многие из описанных выше приемов, в частности метод преобразования Лапласа и использование передаточных функций, могут быть распространены на более широкие классы динамических систем.

9.9.2. Принцип максимума. В заключение опишем еще один класс задач по синтезу законов управления, основанный на применении принципа максимума Л. С. Понтрягина. Наиболее просто этот принцип применяется для динамических систем

$$\begin{aligned}
\frac{dy_1}{dt} &= f_1(y_1, y_2, \ldots, y_n; v_1, v_2, \ldots, v_m),\\
\frac{dy_2}{dt} &= f_2(y_1, y_2, \ldots, y_n; v_1, v_2, \ldots, v_m),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
\frac{dy_n}{dt} &= f_n(y_1, y_2, \ldots, y_n; v_1, v_2, \ldots, v_m).
\end{aligned} \tag{9.49}$$

Системы такого вида, правые части которых не содержат входных параметров v_i и не зависят явно от времени t, принято называть *автономными*. Система (9.49) может быть переписана в векторной форме:

$$\frac{dy}{dt} = f(y, v). \tag{9.50}$$

Решение $y(t)$ уравнения (9.50) принято называть *траекторией*. Управление v ищется для задач рассматриваемого типа в виде $v(t)$, т. е. исключается зависимость управления от основных параметров y_i (обратная связь).

На множестве управлений задается функционал

$$J = \int_{t_0}^{t_1} f_0(y(t), v(t))\, dt, \tag{9.51}$$

оценивающий качество управления. Пределы интегрирования (t_0 и t_1) представляют собой моменты времени, когда точка $y(t)$

находится в начале и в конце оцениваемого отрезка траектории. Простейшим случаем оценочного функционала J, получающимся при $f_0(y, v) \approx 1$, является время $\int_{t_0}^{t_1} dt = t_1 - t_0$ прохождения данного участка траектории.

Количество возможных вариантов $v(t)$ ограничивается замкнутым множеством P допустимых значений. Задача оптимального управления в этом случае ставится следующим образом: в множестве P допустимых управлений найти такое, для которого функционал J принимает наименьшее возможное значение. Для решения этой задачи исходная система (9.49) дополняется $(n+1)$-м уравнением

$$y_0' = f_0(y_1, y_2, \ldots, y_n; v_1, v_2, \ldots, v_m), \tag{9.52}$$

правая часть которого совпадает с подынтегральным выражением оценочного функционала J.

Вводятся вспомогательные неизвестные $\psi_0, \psi_1, \ldots, \psi_n$, для которых выписываются дифференциальные уравнения

$$\frac{\partial \psi_i}{\partial t} = -\sum_{k=0}^{n} \frac{\partial f_k(y, v)}{\partial y_i} \psi_k, \quad i = 0, 1, 2, \ldots, n. \tag{9.53}$$

После их решения в общем виде (с постоянными интегрирования) строится функция

$$H(\psi, y, v) = \sum_{k=0}^{n} \psi_k f_k(y, v).$$

Возможные оптимальные управления находятся на основе принципа максимума Понтрягина:

Для того чтобы управление $v(t)$ было оптимальным, необходимо, чтобы для любого значения t, $t_0 \leqslant t \leqslant t_1$, функция $H(\psi, y, v)$ переменного вектора v достигала максимума в точке $v = v(t)$. При этом величина ψ_0 постоянна и $\psi_0 \leqslant 0$.

В приведенной формулировке принцип максимума дает только необходимые условия оптимальности управления. Для нахождения необходимых и достаточных условий требуются дальнейшие его уточнения. Однако и в приведенном виде принцип может с успехом применяться для решения практических задач.

Предположим, например, что нужно переместить на горизонтальном участке дороги с пренебрежимо малым трением из одной точки в другую транспортное средство, двигатель которого может сообщить ему любое ускорение v, по величине не превос-

ходящее единицу *). Критерием оптимальности будет служить время, затраченное на перемещение. Иными словами, в качестве оценочного функционала выбираем

$$J = \int_{t_1}^{t_2} dt.$$

Уравнение движения объекта будет иметь вид $y_1'' = v$. Вводя обозначение $y_2 = y_1'$, запишем уравнения движения в обычном виде: $y_1' = y_2$, $y_2' = v$. После дополнения уравнением вида (9.52) придем к системе

$$y_0' = 1, \quad y_1' = y_2, \quad y_2' = v.$$

Уравнения (9.53) в данном случае принимают вид

$$\frac{d\psi_0}{dt} = \frac{\partial(1)}{\partial y_0}\psi_0 + \frac{\partial y_2}{\partial y_0}\psi_1 + \frac{\partial v}{\partial y_0}\psi_2 = 0,$$

$$\frac{d\psi_1}{dt} = \frac{\partial(1)}{\partial y_1}\psi_0 + \frac{\partial y_2}{\partial y_1}\psi_1 + \frac{\partial v}{\partial y_1}\psi_2 = 0,$$

$$\frac{d\psi_2}{dt} = \frac{\partial(1)}{\partial y_2}\psi_0 + \frac{\partial y_2}{\partial y_2}\psi_1 + \frac{\partial v}{\partial y_2}\psi_2 = \psi_1.$$

Отсюда $\psi_0 = \text{const}$; $\psi_1 = \text{const}$; $\psi_2 = \psi_1 t + c$ ($c = \text{const}$). Для функции H получим выражение

$$H = \psi_0 + \psi_1 y_2 + \psi_2 v = \psi_0 + \psi_1 y_2 + (\psi_1 t + c)v.$$

Поскольку $-1 \leqslant v \leqslant 1$, то максимум функции H достигается при $v = -1$, если $\psi_1 t + c < 0$, и при $v = +1$, если $\psi_1 t + c > 0$. Будучи линейной, функция $\psi_1 t + c$ на заданном интервале может менять знак не более одного раза. Поэтому на этом интервале возможны лишь следующие последовательности оптимальных управлений: (-1), $(+1)$, $(-1, +1)$ или $(+1, -1)$.

Выбор нужной последовательности и момент переключения управления в двух последних случаях происходят в зависимости от значений y_1 и y_2 на концах рассматриваемого интервала. Пусть, например, $t_1 = 0$; $y_1(0) = 0$; $y_2(0) = 1$; $y_1(t_2) = 2$; $y_2(t_2) = 0$. Закон движения на участке от 0 до τ, получаемый из уравнений движения, запишется в виде

$$y_2(t) = v(t) + y_2(0), \quad y_1(t) = vt^2/2 + ty_2(0) + y_1(0),$$

или

$$y_2 = vt + 1, \quad y_1 = vt^2/2 + t.$$

*) Решение этой задачи очевидно, так что последующие вычисления служат лишь для иллюстрации применения принципа максимума.

Выбор нужной последовательности управлений можно осуществить методом перебора. На начальном участке $[0, \tau]$ выбираем управление $v = +1$ (режим разгона) и записываем уравнения движения в виде

$$y_2 = t + 1, \quad y_1 = t^2/2 + t.$$

На участке $[\tau, t_2]$ выбираем управление $v = -1$ (режим торможения). Уравнения движения представятся в виде

$$y_2 = -t + c_1, \quad y_1 = -t^2/2 + c_1 t + c_2.$$

Для точки t_2 получаем соотношения

$$y_2(t_2) = -t_2 + c_1 = 0, \; y_1(t_2) = -t_2^2/2 + c_1 t_2 + c_2 = 2,$$

откуда $c_1 = t_2$; $c_2 = 2 - t_2^2/2$. Для точки τ из двух законов движения имеем

$$y_2(\tau) = \tau + 1 = -\tau + c_1, \quad y_1(\tau) = \tau^2/2 + \tau = -\tau^2/2 + c_1\tau + c_2,$$

откуда $2\tau = c_1 - 1 = t_2 - 1$, или $\tau = (t_2 - 1)/2$. Подставляя это значение в последнее уравнение, найдем

$$\left[\frac{1}{2}(t_2 - 1)\right]^2 + \frac{1}{2}(t_2 - 1) - t_2\left(\frac{t_2 - 1}{2}\right) - \left(2 - \frac{t_2^2}{2}\right) = 0,$$

или $t_2^2 + 2t_2 - 9 = 0$. Следовательно, $t_2 = -1 \pm \sqrt{1 + 9} = \pm\sqrt{10} - 1$. Из двух корней реальный физический смысл имеет лишь положительный. Таким образом,

$$t_2 = \sqrt{10} - 1, \; \tau = (t_2 - 1)/2 = \sqrt{2{,}5} - 1.$$

Следовательно, разгоняясь на временно́м интервале $[0, \sqrt{2{,}5} - 1]$ с максимальным ускорением $v = 1$ и затем тормозя с таким же максимальным ускорением, мы достигнем требуемой точки $y_1 = 2$ за минимально возможное время, равное $\sqrt{10} - 1$.

Случай неавтономных динамических систем можно, вводя новое основное переменное $y_{n+1} = t$, свести к уже изученной задаче для уравнений, не содержащих явно t в правых частях.

9.9.3. Цифровое управление непрерывными динамическими системами. Непрерывные технологические процессы и многие другие объекты управления, встречающиеся на практике, хорошо описываются с помощью дифференциальных уравнений. Аналоговые вычислительные устройства и простые механические регуляторы, которые в течение достаточно длительного времени были единственным средством автоматизации, также естественным образом описываются языком дифференциальных уравнений. Применение для целей управления электронных цифровых машин вызвало необходимость определенных видоизменений тео-

рии автоматического управления. Поскольку при применении цифровых машин измерения параметров, характеризующих объекты управления, так или иначе производятся в дискретные моменты времени, производные dy_i/dt естественным образом заменяются разделенными разностями $\Delta y_i/\Delta t$. В результате возникает аппроксимация системы дифференциальных уравнений соответствующей системой *конечноразностных уравнений*.

Теории этих двух классов систем (особенно в линейном случае) в значительной мере параллельны друг другу. Подобно тому как линейную систему дифференциальных уравнений можно свести к системе линейных алгебраических уравнений с помощью преобразования Лапласа, то же самое может быть сделано применительно к конечно-разностным уравнениям с помощью так называемого *z-преобразования*.

Если функция $f(t)$ на интервале $0 \leqslant t < \infty$ задана в дискретном множестве точек с интервалом дискретности d, то ее z-преобразование $F(z)$ задается формулой

$$F(z) = \sum_{n=0}^{\infty} f(nd)\, z^{-n}, \tag{9.54}$$

где z — комплексная переменная, $z = x + iy$. *Обратным z-преобразованием*, задаваемым формулой

$$f(kd) = \frac{1}{2\pi i} \oint F(z)\, z^{k-1} dz, \quad k = 0, 1, 2, \ldots \tag{9.55}$$

(где замкнутый контур интегрирования в комплексной плоскости проходит внутри области сходимости ряда (9.54), охватывая начало координат), можно восстановить исходную функцию $f(t)$ в заданном дискретном множестве точек $t = kd$ $(k = 0, 1, 2, \ldots)$. В силу известной теоремы Коши, для любого полинома $P(z)$ имеем $\oint P(z)\, dz = 0$ при любом (замкнутом) контуре интегрирования. То же самое будет при замене $P(z)$ на дробь $A/(z-a)^p$ при $p = 2, 3, \ldots$ И лишь при $p = 1$, если контур интегрирования охватывает точку a, имеем $\oint \frac{A}{z-a} dz = A \cdot 2\pi i$. Если под знаком интеграла (9.55) стоит рациональная функция $P(z)$ от z, то ее всегда можно представить в виде суммы полинома и дробей вида $A/(z-a)^p$. При этом, в силу сделанного предположения о контуре интегрирования, все точки разрыва подынтегральной функции (т. е. точки $z = a$) будут находиться внутри этого контура *). Тем самым функция $f(kd)$, вычисляемая по формуле (9.55), будет равняться просто сумме числителей дро-

*) Ряд (9.54) сходится вне некоторой окружности с центром в начале координат, так что в этой области функция $F(z)$ не имеет разрывов.

бей вида $A/(z-a)$ в разложении подынтегральной функции $F(z)\cdot z^{k-1}$.

Легко видеть, что z-преобразование функции $f(nd)$ со «сдвинутым» аргументом, т. е. функции $f(nd-kd)$, равно $f(-kd)+z^{-1}f(-k+1)d+\ldots+z^{-(k-1)}f(-d)+z^{-k}F(z)$. Зная начальные условия, т. е. значения функции $f(nd)$ при $n=-k, -k+1, \ldots, -1$, легко найдем требуемое z-преобразование. Как и для преобразования Лапласа, может быть построен словарь прямых и обратных преобразований распространенных функций, что позволяет способом, аналогичным описанному в п. 9.9.1, находить решения систем разностных уравнений.

Поясним сказанное на простом примере. Пусть требуется решить разностное уравнение $y(nd)=2y(nd-d)+x(nd)$ при начальном условии $y(-d)=1$. После применения к обеим частям уравнения z-преобразования получим $Y(z)=2+2z^{-1}Y(z)+X(z)$, откуда $Y(z)=\frac{z}{z-2}(2+X(z))$. Пусть $x(nd)$ представляет собой единичный начальный импульс, т. е. $x(0)=1$, $x(nd)=0$ при $n>1$. Тогда по формуле (9.54) $x(z)=1$. Так как $Y(z)z^{k-1}=\frac{3z^k}{z-2}=P(z)+\frac{3\cdot 2^k}{z-2}$, где $P(z)$ — некоторый полином от z, то по формуле (9.55) получаем $f(kd)=3\cdot 2^k$ $(k=0, 1, \ldots)$. Тем самым мы нашли требуемое решение заданного конечно-разностного уравнения. Заметим, что из полученной формулы для $f(kd)$ нельзя получить начального значения для $k=-1$. Это обстоятельство имеет место и в общем случае, так что при необходимости учитывать значения решения для отрицательных значений аргумента (соответствующих начальным условиям) их следует просто добавлять к решению непосредственно из условия задачи.

По аналогии с п. 9.9.1 решаются также задачи анализа и синтеза систем автоматического управления. При этом управляющая ЭВМ должна моделировать некоторую систему уравнений в конечных разностях, описывающую ее поведение как автоматического дискретного регулятора. На универсальной ЭВМ может быть смоделирована любая подобная система. Однако неудобство универсальной (однопроцессорной) машины заключается в том, что все операции по фактическому решению моделируемой ею системы она выполняет последовательно. В результате не исключено, что быстродействия ЭВМ может оказаться недостаточно для выполнения функций дискретного регулятора с требуемым (достаточно малым) интервалом дискретности A. Поэтому в качестве дискретных авторегуляторов иногда предпочитают использовать специализированные ЭВМ и, прежде всего, так называемые *цифровые дифференциальные анализаторы* (ЦДА).

В ЦДА наряду с обычными для универсальных ЭВМ устройствами (например, сумматорами) широко используются *цифровые интеграторы*, т. е. устройства, реализующие описанный в начале § 9.9 простейший метод приближенного решения задачи Коши для обыкновенных дифференциальных уравнений. Как ясно из его описания, этот метод решает, строго говоря, не систему обыкновенных дифференциальных уравнений, а аппроксимирующую ее систему конечно-разностных уравнений, т. е. выполняет (будучи реализован в ЭВМ) функции дискретного авторегулятора. Благодаря возможности иметь на каждую операцию подобного «дискретного интегрирования» свой собственный интегратор скорость работы ЦДА (в режиме авторегулятора) может быть значительно выше по сравнению с построенной на той же элементной базе универсальной ЭВМ. Впрочем, того же самого эффекта можно добиться, используя универсальную многопроцессорную ЭВМ.

9.10. Станки с числовым программным управлением

Одним из важнейших направлений автоматизации технологических процессов является *числовое программное управление* (ЧПУ) металлорежущими станками и другим оборудованием. Для реализации ЧПУ станок должен быть снабжен исполнительными механизмами (приводами), осуществляющими взаимное перемещение режущего инструмента и обрабатываемой детали. Существуют два основных принципа построения таких механизмов. Во-первых, это *шаговые двигатели*, которые, получая двоичный сигнал (+1), осуществляют перемещение (обычно вращение) на один элементарный шаг в прямом или обратном направлении (иногда только в прямом). Осуществляя (с помощью точных, практически безлюфтовых передач) связь с перемещаемым объектом, шаговый двигатель способен обеспечить высокую точность (измеряемую микронами) элементарных перемещений этого объекта (инструмента или стола). Второй принцип — это *следящие системы*, представляющие собой двигатели, устанавливающиеся в определенную позицию (чаще всего угол поворота) в зависимости от получаемого ими цифрового (полноразрядного) кода этой позиции. Существуют и промежуточные решения, когда маломощный (информационный) шаговый двигатель определяет устанавливаемую позицию, а аналоговое следящее устройство устанавливает в ту же позицию мощный (силовой) двигатель, перемещающий объект управления.

Решающим для выбора того или иного принципа построения исполнительного механизма является, с одной стороны, точность, а с другой — развиваемое усилие. Последнее важно не только вследствие необходимости преодоления усилий, вызыва-

емых рабочим ходом (процессом резания), но и для обеспечения быстроты холостых ходов (перемещениях в новую рабочую позицию), что связано с приданием управляемому объекту значительных ускорений, а следовательно, и соответствующих усилий. Одно время первенство в одновременном обеспечении двух, вообще говоря, взаимно противоречивых требований точности и силы (быстроты) было за силовыми шаговыми двигателями. Однако успехи в создании мощных и достаточно высокоточных цифровых следящих систем составили им в последние годы серьезную конкуренцию.

Заметим, что некоторые исполнительные механизмы в станках с ЧПУ — так называемые *позиционные устройства* — обеспечивают только холостые ходы рабочего органа станка, т. е. его перемещение в нужную для работы позицию. Это имеет, например, место в случае сверлильных станков с ЧПУ, у которых обработка (сверление) выполняется только после установки рабочего органа в требуемую позицию.

Однокоординатные рабочие устройства обеспечивают рабочий ход режущего инструмента в одном направлении, например, продвижении сверла в глубь просверливаемого отверстия или движение суппорта токарного станка (перемещение резца) в направлении, параллельном оси вращения обрабатываемой детали.

Многокоординатные рабочие устройства обеспечивают одновременное перемещение рабочего органа в нескольких направлениях. Это имеет, например, место в трехкоординатных фрезерных станках с ЧПУ, обеспечивающих обработку сложных криволинейных поверхностей. Многокоординатные устройства называют иногда поэтому *контурными криволинейными системами* ЧПУ в отличие от контурных *прямоугольных* (естественнее было бы сказать прямолинейных) систем, каковыми являются однокоординатные устройства.

В последние годы получают все большее распространение рабочие устройства, способные в процессе работы автоматически менять режущий инструмент (сверла, фрезы и т. п.), производя тем самым различные технологические операции без перемещения детали с одних станков на другие. Подобные станки, получившие наименование *обрабатывающих центров*, пуждаются, разумеется, в дополнительных (программно-управляемых) приводах для автоматической установки в рабочую позицию нужного инструмента.

Рабочая программа, непосредственно управляющая рабочими устройствами, в простейшем случае должна состоять (в зависимости от вида привода) из последовательностей либо однобитовых кодов (при шаговом приводе), либо полноразрядных кодов (при следящем приводе). Впрочем, в последнем случае с целью экономии, имея на приводе регистр-сумматор, можно пе-

редать полный код только один раз — в начале работы, а затем ограничиться передачей лишь *приращений* этого кода. Соответственно должна быть построена и рабочая программа.

Учитывая большую точность работы современных станков с ЧПУ (малость элементарных перемещений), нетрудно видеть, что подобный простейший способ организации рабочей программы привел бы к непомерному раздуванию ее размеров. Поэтому на практике обычно поступают иначе, снабжая станки с ЧПУ специальными электронными устройствами — *интерполяторами* (сегодня, как правило, лишь цифровыми). Благодаря наличию таких устройств рабочая программа может содержать информацию лишь об относительно небольшом числе так называемых *опорных* точек, характеризующих позицию рабочих органов станка в процессе обработки детали. Сигналы же, непосредственно управляющие приводами (исполнительными механизмами), вырабатываются интерполяторами.

Об интерполяции достаточно подробно говорилось выше (§ 9.3). Поэтому мы ограничимся здесь лишь некоторыми замечаниями. Простейшим видом интерполяции является, как известно, линейная интерполяция. Несмотря на ее очевидные недостатки, прежде всего отсутствие *гладкости* (т. е. непрерывности производной) при сопряжениях двух соседних прямолинейных отрезков, линейная интерполяция находит достаточно широкое распространение в силу своей простоты. Кроме того, отсутствие гладкости обрабатываемой поверхности, очевидно, не будет иметь места в однокоординатных (прямолинейных) системах. Что же касается криволинейных (многокоординатных) систем, то отсутствие гладкости при линейной интерполяции может быть до известной степени скомпенсировано увеличением числа опорных точек, хотя это и ведет к нежелательному увеличению размеров рабочих программ.

В двух других, широко применяющихся на практике методах интерполяции — параболической и круговой — обеспечивается гладкость (непрерывность первой производной) на стыках интерполяционных участков. Рассмотрим сначала случай параболической интерполяции. Если интерполяционный полином на предыдущем интервале (между точками $A_{i-1}(x_{i-1}, y_{i-1})$, $A_i(x_i, y_i)$) выражался формулой $y_{i-1} = a_{i-1}x^2 + b_{i-1}x + c_{i-1}$, то полином $y_i = a_i x^2 + b_i x + c_i$ для следующего участка (между точками $A_i(x_i, y_i)$, $A_{i+1}(x_{i+1}, y_{i+1})$) должен удовлетворять соотношениям $y_i = a_i x_i^2 + b_i x_i + c_i$, $y_{i+1} = a_i x_{i+1}^2 + b_i x_{i+1} + c_i$, $y_i' = 2a_{i-1}x_i + b_{i-1} = 2a_i x_{i+1} + b_i$. Из этих соотношений и находятся коэффициенты искомого интерполяционного полинома.

В случае круговой интерполяции (интерполирующая функция — уравнение окружности) принцип нахождения окружности для интерполяции на следующем участке лучше всего уяснить

себе из рис. 9.9. Если O_{i-1} — центр интерполирующей окружности для участка между точками A_{i-1} и A_i, то центр интерполирующей окружности для следующего участка находится пересечением линии радиуса A_iO_{i-1} первой окружности с перпендикуляром C_iO_i к отрезку A_iA_{i+1} в его середине. Радиусы окружностей после нахождения центров находятся по известной формуле для расстояния между двумя точками. При применении интерполирующих полиномов порядка выше второго можно обеспечить гладкость сопряжения участков по второй и более высоким производным. Однако на практике подобный уровень гладкости сопряжения обычно не бывает нужен.

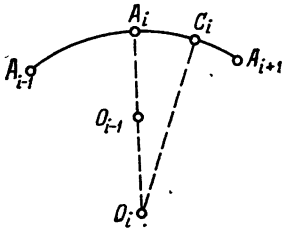

Рис. 9.9

Как при наличии интерполятора на станке, так и при его отсутствии управление работой станка может осуществляться от простейшего *программного устройства*, способного читать рабочую программу, записанную на том или ином *машинном* носителе. На большинстве станков с ЧПУ, работающих по этому принципу, в качестве такого носителя употребляется стандартная 8-дорожечная перфолента, хотя не исключено использование и других носителей (прежде всего магнитных лент и гибких дисков).

При такой простейшей организации программного устройства возникают два основных неудобства. Первое связано с необходимостью *синхронизации* темпа поступления команд от программного устройства и скоростью работы станка. Это требует либо пошагового управления движением носителя с программой (что проще всего и, главное, без потери информационной емкости носителя реализуется на перфоленте), либо наличия специальной буферной (электронной) памяти. Второе неудобство еще более существенно. Дело в том, что в процессе работы станка рабочий инструмент постепенно изнашивается, меняя свои первоначальные размеры (это особенно сказывается в случае шлифовальных кругов). Необходимы поэтому соответствующие *оперативные корректировки* рабочей программы.

Подобные корректировки требуют отхода от описанного простейшего способа организации числового программного управления, если мы хотим полностью устранить ручной труд для соответствующей переналадки рабочего органа станка. Радикальным средством решения этой проблемы является управление станком (или группой станков одновременно) непосредственно от управляющей ЭВМ. Появление дешевых микропроцессоров позволило вставлять изготовляемые на их основе микро-ЭВМ непосредственно в каждый станок с ЧПУ.

Для организации обратной связи от процесса обработки создаются специальные *автоматизированные измерительные комплексы.* Производя автоматические измерения готовых деталей (в случае изготовления партий относительно простых деталей) или обрабатываемой поверхности в процессе работы (в случае сложных деталей с длительным циклом изготовления), получают исходную информацию об изменениях заданных параметров в результате износа рабочего инструмента. И прежде чем эти изменения выйдут за пределы установленных допусков, в управляющих ЭВМ будут произведены необходимые корректировки рабочих программ. Тем самым для числового программного управления возникает принципиально новое качество — *адаптивность.* Для координации работы управляющих ЭВМ станков и измерительных комплексов, выполняющих различные операции единого технологического процесса, может возникнуть необходимость в специальной управляющей ЭВМ второго уровня. Тем самым осуществляется переход от автоматизации работы отдельных единиц оборудования к автоматизации технологических линий, производственных участков и целых цехов.

Следует заметить, что, хотя наибольшее распространение числовое программное управление получило для металлорежущих станков, сам принцип ЧПУ имеет гораздо более широкую сферу применения. Его с успехом применяют в деревообработке, для управления штамповочным, линейным, гальваническим производством и во многих других случаях. Первоначально выражались надежды, что принцип ЧПУ найдет свое главное применение в индивидуальном производстве. Эти надежды пока не оправдались главным образом потому, что процесс программирования для оборудования с ЧПУ (несмотря на имеющийся значительный прогресс) остается все еще достаточно трудоемким. Образно выражаясь, во многих случаях оказывается быстрее сделать деталь вручную, чем составить и, главное, отладить программу для станка с ЧПУ *).

Поэтому сегодня оборудование с ЧПУ находит главное свое применение в мелкосерийном и среднесерийном производстве. Разумеется, его можно с успехом использовать и в массовом производстве. Однако в этом случае главное преимущество оборудования с ЧПУ — именно *гибкость* (т. е. возможность без переделок оборудования быстро перестраиваться на различные виды работы) — не будет фактически использоваться. А саму автоматизацию массового производства в большинстве случаев можно осуществить более эффективно за счет

*) Это, разумеется, не относится к очень сложным деталям с длительным циклом изготовления. Кроме того, сегодня имеются работы, которые могут быть выполнены только на станках с ЧПУ.

узкоспециализированного (а потому, как правило, более дешевого и высокопроизводительного) оборудования.

Дальнейшее продвижение оборудования с ЧПУ в индивидуальное производство будет связано с процессом автоматизации подготовки и отладки программ для него. В перспективе этот процесс должен изготовлять рабочие программы для оборудования с ЧПУ непосредственно с конструкторской документации, автоматически рассчитывая все технологические режимы. Для проверки же правильности составленных рабочих программ все стадии обработки деталей по этим программам желательно автоматически демонстрировать в регулируемом (чаще всего в сторону ускорения) темпе с возможностью автоматического расчета и выдачи на дисплей любых показателей, характеризующих процесс (температуру нагрева резца, величину прогиба детали и т. п.). Пока автоматизация программирования для оборудования с ЧПУ в целом (за исключением отдельных примеров полуучебного характера) еще далека от такого уровня. Опишем теперь кратко современное состояние этого вопроса применительно к станкам с ЧПУ.

9.10.1. Автоматизация программирования для станков с ЧПУ. Каждый тип станка с ЧПУ в зависимости от числа и характера приводов и других особенностей имеет свой собственный машинный (в данном случае — станочный) язык программирования. В набор команд такого языка могут входить перемещения по трем осям координат, повороты вокруг осей, изменения скорости вращения (шпинделя, сверла, фрезы и др.), величина подачи, выбор определенного инструмента для работы (в револьверных станках и обрабатывающих центрах), управление режимом охлаждения и др. Программирование непосредственно в машинном языке представляет собой весьма трудоемкую задачу. Ее облегчение производится в основном теми же методами, какими решается задача упрощения программирования для ЭВМ в общем случае (см. гл. V).

Первый уровень упрощения состоит во введении машинно-ориентированного языка с символическими обозначениями команд (без двоичного кодирования), что сильно упрощает программирование и уменьшает число ошибок. Дополнительно в случае языков для ЧПУ в машинно-ориентированном языке допускается использование метода опорных точек, даже если станок, для которого составляется программа, не имеет собственного интерполятора. Программа-ассемблер (называемая при работе с ЧПУ еще *постпроцессором*) не только произведет необходимое кодирование, но и введет в программу (в соответствии с принятым методом интерполяции) все промежуточные точки. Команды перехода от одной опорной точки к другой, употребляющиеся в исходной программе, могут рассматриваться как

специфические для ЧПУ макрооператоры. На практике для станков без индивидуальных интерполяторов используются также групповые интерполяторы, преобразующие ленты с закодированными макрооператорами в ленты со станочными (самыми мелкими) операциями.

Следующий уровень автоматизации программирования — использование проблемно-ориентированных (на проблемы ЧПУ) входных языков и трансляторов с них. В ЧПУ используется обычно так называемая *кросс-трансляция*, когда процесс трансляции осуществляется не на управляющей машине, которая будет реализовать эту программу (тем более что такая машина может и не существовать), а в универсальной (как правило, достаточно большой) ЭВМ, установленной в специальном ВЦ, который снабжает программами для оборудования с ЧПУ целый завод, а иногда и группу заводов.

Как и все алгоритмические входные языки, входные (проблемно-ориентированные) языки для ЧПУ содержат, во-первых, средства для *описания объектов*, с которыми они оперируют, во-вторых, средства для выражения *действий* (операций) над этими объектами. Общим для всех входных языков ЧПУ является то, что в качестве объектов в них прежде всего выступают обрабатываемые изделия (на разных стадиях обработки). С этой целью могут описываться точки (с помощью задания их координат или в виде пересечения заданных линий), прямые линии (с помощью двух точек или точки и углового коэффициента), дуги окружностей, а также (в развитых языках) любые кривые и поверхности, задаваемые их уравнениями. Вводятся также символические обозначения для инструментов различных типов (фрезы, сверла и т. д.), а также термины, задающие различную технологическую информацию (тип интерпретатора, величина подачи и др.).

Операторы, идущие после описания, разделяются в зависимости от типа обработки. Для контурной обработки на фрезерных станках с ЧПУ широко употребляется оператор ИДИ, после которого следует перечень элементов пути, который должен проделать режущий к р а й фрезы. Например, ФРСЛ; ИДИ/ПР1, ОКРЗ, ПР4 ДО ТЧК 5 означает, что край фрезы (остающейся слева от обрабатываемого контура) должен пройти отрезок прямой 1, дугу окружности 3 и отрезок прямой 4 до точки 5. Разумеется, все эти элементы должны быть предварительно точно описаны. Задача транслятора состоит в том, чтобы по этим указаниям (дополненным данными в описании сведениями о типе фрезы, величине подачи и другой технологической информации) вычислить фактически необходимый путь ц е н т р а фрезы и выдать на запись (с учетом типа используемого станка и интерполятора) в кодированном виде рабочую программу. Холостые ходы указываются обычно опера-

тором ИДИ В или ИДИ НА (с указанием идентификаторов точки или кривой). Если отсутствуют дополнительные указания, холостой ход совершается по кратчайшему пути.

В СССР для программирования работы фрезерных станков с ЧПУ разработаны и используются языки СПС-Т, СПС-ТАУ и др. Поскольку языки для токарных станков, в отличие от фрезерных, описывают лишь тела вращения, они допускают определенные упрощения описаний. Например, нет необходимости описывать любые поверхности, поскольку любая поверхность вращения может быть задана соответствующей линией. Еще проще в принципе входные языки программирования для сверлильных станков, хотя многошпиндельная обработка может внести дополнительные трудности в разработку трансляторов. Использование современных входных языков для ЧПУ позволяет сократить трудоемкость подготовки программ в 10—20 раз и стоимость — в 5—15 раз.

Заметим, что все упомянутые входные языки служат лишь для описания движений инструмента и детали при ее обработке на станке. Что же касается оптимизации этих движений с учетом необходимых технологических ограничений (по скорости, точности и др.), то она является предметом *технологического проектирования*, автоматизация которого представляет собой самостоятельную (и притом достаточно трудную) задачу. В перспективе эти два направления автоматизации должны слиться в единый процесс.

9.11. Промышленные манипуляционные роботы

Манипуляционные роботы, или *манипуляторы*, представляют собой многоцелевые программно-управляемые устройства, имитирующие в той или иной мере человеческую руку. Основные области их применений включают установочные, транспортные и сборочные операции. Достаточно широкое применение находят они и в ряде других работ (сварка, покраска и др.). Главным рабочим органом робота-манипулятора служит механическая «рука», оканчивающаяся «кистью», которую принято называть *схватом*. В отличие от человеческой кисти с пятью пальцами, схват современных роботов имеет в большинстве случаев два, а иногда три механических пальца. Зато кисть робота обычно снабжается возможностью неограниченного вращательного движения (весьма удобного для операций завинчивания и отвинчивания, сверления и др.), чего кисть человека, как известно, лишена.

Кроме «вращательного сустава» в сочленении со схватом, рука робота имеет обычно еще один или несколько «суставов», позволяющих изгибать руку. Наконец, для успешного выполне-

ния роботами своих функций употребляются еще выдвижные сочленения, позволяющие удлинять или сокращать те или иные участки руки.

В зависимости от назначения промышленного робота число степеней свободы движения его руки колеблется сегодня от трех до двенадцати (наиболее часто — от четырех до семи). Во всяком случае кинематическая схема руки робота достаточно сложна, что создает определенные трудности как в программировании ее движений, так и особенно — в их оптимизации. Задача состоит в согласовании движений во всех суставах, с тем чтобы обеспечить оптимальную траекторию движения схвата (обычно наиболее быструю). При ее решении приходится учитывать ограничения подвижности в суставах, ограничения на ускорения, вызванные ограниченной мощностью приводов, а также ограничения, связанные с обходом возможных препятствий на пути движения руки.

Решение этой задачи сводится в конечном счете к оптимизации решения некоторой системы обыкновенных дифференциальных уравнений, для чего может быть использован описанный выше принцип максимума. Проблемы программирования манипуляционных роботов связаны, прежде всего, с задачей описания траекторий и законов движения их исполнительных органов (рук). В этой части задачи программирования роботов в целом аналогичны подобным задачам для станков с ЧПУ. Для облегчения процесса транслирования с языков высокого уровня в специализированные машинно-ориентированные языки, описывающие движения руки робота, целесообразно вводить наборы макрооператоров, которые описывают согласованные движения в суставах для выделенных элементарных движений руки в целом.

Привод для осуществления движений, как и в станках ЧПУ, у роботов осуществляется либо с помощью шаговых двигателей, либо с помощью следящих систем. Для управления выдвижными сочленениями могут употребляться одноступенчатые или многоступенчатые (на 1/2 полного движения, на 1/4 и т. д.) электромагнитные (соленоидные) или гидравлические приводы. Гидравлические (а иногда для маломощных роботов — пневматические) системы применяются для осуществления движений и других типов (особенно схвата). Компоновка приводов бывает различной. У маломощных роботов двигатели предпочитают размещать прямо в суставах. У мощных роботов двигательный блок чаще всего размещается у «плечевого» сустава, а передача усилий осуществляется механическими тягами или гидроприводом.

Роботы-манипуляторы первого поколения представляют собой программно-управляемые механизмы в чистом виде (без обратной связи). Поэтому их применение ограничивается теми областями, где объекты, с которыми работают роботы, строго п о з и-

ционированы, ибо подобный «неочувствленный», строго запрограммированный робот не способен правильно захватить смещенный из заданной позиции предмет. Главная область применения таких роботов — это машиностроение и приборостроение. Будучи установлены (в жестких позициях) рядом с другим программно-управляемым оборудованием, такие роботы способны перемещать обрабатываемое изделие от одной единицы оборудования к другой. Тем самым станки с ЧПУ, прессы и другое программно-управляемое оборудование превращается в *автоматические технологические линии.*

В отличие от автоматических линий докибернетической эпохи, эти линии обладают большой гибкостью: после смены программ, без всяких переделок оборудования, линия может начать выпускать совершенно иную продукцию, чем прежде. Для согласования работы всех единиц оборудования, составляющих линию, обычно используется дополнительная управляющая ЭВМ.

При автоматизации сборочных операций добиться строгого позиционирования удается далеко не всегда. То же самое происходит при постановке руки робота на самодвижущуюся тележку для обслуживания нескольких единиц оборудования или при автоматизации подъемно-транспортных операций. Для этих целей требуются «очувствленные» роботы, способные в какой-то мере адаптироваться к изменениям окружающей обстановки, вводя соответствующие корректировки в параметры управляющих программ.

Подобные *адаптивные роботы* (относящиеся к так называемому второму поколению роботов) снабжаются простейшими «органами чувств». Это, прежде всего, *тактильные* (осязательные) *датчики* на схвате, способные регистрировать прикосновения схвата к другим предметам. Благодаря большому числу таких датчиков (обычно несколько десятков) по их показаниям можно во многих случаях восстановить ориентировку при нарушениях позиционирования. Подобную же роль играют различного рода аналоги органа зрения. В простейшем случае — это системы датчиков, способных реагировать на специальные *метки*, нанесенные на предметы, с которыми оперирует робот. Датчики эти могут быть магнитными или оптическими.

У так называемых *интеллектуальных роботов*, или роботов третьего поколения, «органы чувств» (особенно зрение) получают дальнейшее развитие. Они способны распознавать достаточно сложные изображения (сегодня, правда, в основном имеющие правильные геометрические формы), в частности изображения машинных деталей, рассматриваемых под различными ракурсами и при различном освещении. Важной особенностью интеллектуальных роботов является также возможность самостоятельно вырабатывать (и, разумеется, осуществлять) планы до-

стижения тех или иных (относительно простых) целей и менять эти планы в соответствии с изменениями ситуации. О методах решения указанных задач мы расскажем ниже (в главе об искусственном интеллекте). Сейчас же заметим только, что интеллектуальные роботы открывают широкие перспективы для автоматизации самых разнообразных процессов. Однако на пути их широкого внедрения в народное хозяйство стоит пока достаточно высокая их стоимость (в десятки раз превосходящая стоимость роботов первого поколения). Однако успехи микроэлектроники открывают перспективы резкого снижения стоимости ЭВМ высокой производительности, а вместе с ними и интеллектуальных роботов.

Следует подчеркнуть, что манипуляционными роботами отнюдь не ограничивается круг устройств, которые современная наука и техника понимает под термином «робот». Сюда сегодня принято относить все устройства с «человекоподобным» поведением, не обязательно связанным с наличием «рук». Существуют, например, роботы для подводных исследований, роботы-автопилоты (способные вести самолет на сверхнизких высотах и в других сложных условиях) и т. д. Возможности и перспективы подобного рода роботов связаны с исследованиями по искусственному интеллекту, о которых говорится в последней главе настоящей книги.

9.12. Электронная вычислительная техника в быту

Появление микропроцессоров вызвало резкое уменьшение габаритов и стоимости универсальных управляющих ЭВМ (в случае, когда они не нуждаются во внешней памяти и оперативной памяти большой емкости). Это обстоятельство привело к огромному расширению круга применений таких машин и, в частности, к началу широкого их внедрения в бытовую технику. Перечислим кратко основные направления такого внедрения.

Обыкновенный телефонный аппарат, снабженный простейшим микропроцессором с небольшой памятью, предоставляет абоненту по крайней мере две дополнительные услуги. Одна из них — свойство повторного вызова. Если после набора номера он оказывается занятым, микропроцессор запоминает номер и повторяет вызов до тех пор, пока не произойдет соединение. Вторая услуга — возможность вместо длинных номеров использовать при вызове короткие (обычно двузначные) коды. Соответствие между часто вызываемыми номерами и их двузначными кодами устанавливает (и по своему желанию меняет) сам абонент. Обычно используется таблица до ста номеров (более ста номеров закодировать двузначными кодами нельзя). При установке дополнительной аппаратуры на АТС встроенный в аппарат мик-

ропроцессор может предоставить абоненту и третий вид услуг. Речь идет об автоматической переадресации вызовов, приходящих на данный аппарат, на любой другой номер (по которому в данный момент находится владелец аппарата).

Вторым бытовым прибором, являющимся объектом автоматизации, является телевизор. Встроенный в телевизор микропроцессор способен превратить его экран в своеобразный домашний дисплей, на который можно выводить буквенно-цифровую и графическую информацию. Ввод такой информации можно производить с помощью специальной приставки, снабженной соответствующей клавиатурой, а также, в случае необходимости, рукоятками, позволяющими перемещать по экрану точечный курсор или целые изображения (например, изображение теннисной ракетки). Используя эту приставку, с помощью специальных программ (встроенных в сменяемые миниатюрные ДЗУ) можно сделать подобный автоматизированный телевизор партнером по самым разнообразным играм (шахматы, шашки, «теннис» и др.).

Будучи снабжен электронными часами, микропроцессор может осуществлять включение и выключение телевизора, а также переключения с одной программы на другую в любые заданные пользователем моменты времени. Очень важным новым качеством, которое приобретает телевизор с микропроцессором и соответствующим ЗУ, является способность воспринимать буквенно-цифровую и графическую информацию, передаваемую не по широкополосному телевизионному, а по узкополосному каналу. Используя аппаратуру разделения обычных (эфирных) каналов или специальную кабельную разводку, можно выводить на экраны телевизоров информацию по индивидуальным запросам. Разумеется, для этой цели при телецентре должен быть организован специальный автоматический справочно-информационный центр. Для передачи запросов в этот центр можно использовать обычную телефонную связь. Если при этом микропроцессор телефонного аппарата соединен с микропроцессором телевизора, то из центра через запрашивающий абонентский телефон может быть выдан в телевизионный приемник (для автоматической настройки) номер подканала, по которому будет выдана запрашиваемая информация*).

Третьим объектом автоматизации является пишущая машинка. Если снабдить ее дисплеем с ЗУ и возможностью автоматической печати, то владелец машинки получает возможность редактирования текста на экране дисплея перед его окончательной печатью. С этой целью печатаемый первоначально сырой (неотредактированный) текст запоминается в ЗУ и отобража-

*) Информация может выдаваться и непосредственно через телефонную линию.

ется на дисплее. Автор текста с помощью перемещаемого курсора и специальных редакторских клавиш вносит те или иные исправления в текст (вписывает или вычеркивает буквы, слова или целые фразы, меняет их местами, делает переразбивку абзацев и т. п.). После того как текст отредактирован, электрифицированная пишущая машинка печатает его с большой скоростью. Возможен также вариант передачи отредактированного текста по телефону (с использованием модема) для печати текста на аналогичной машинке в любом другом месте, вариант автоматического изготовления машинно-читаемых копий (на месте или у удаленного абонента) и т. д.

Имея в виду относительно высокую стоимость обычных дисплеев, их можно заменять автоматизированными по описанному выше образцу бытовыми телевизорами.

Следующими объектами автоматизации являются такие бытовые приборы, как стиральные машины, холодильники, электроплиты и др. С помощью встроенных управляющих микро-ЭВМ можно осуществлять программное управление этими приборами, например, включать и выключать горелки электроплиты в назначенное время, программно менять интенсивность нагрева и т. п.

Большой эффект дает автоматизация с помощью управляющей микро-ЭВМ и соответствующей системы датчиков системы приготовления бензовоздушной смеси и управления зажиганием в обыкновенном автомобильном моторе. Учитывая не только скорость вращения вала двигателя, но и его (угловое) ускорение, а также температуру воздуха, атмосферное давление и ряд других параметров, микро-ЭВМ обеспечивает гораздо более экономичные режимы работы двигателя (экономия топлива до 10—12%), а также устранение неполного сгорания топлива и связанного с этим загрязнения окружающей среды.

При наличии в городе центра автоматического управления уличным движением, способного передавать автоматически (по радио) информацию о текущей дорожно-транспортной ситуации (закрытых проездах, автомобильных пробках и т. п.), оказывается полезной бортовая микро-ЭВМ—штурман. Вводя в нее задание на поездку (координаты начального и конечного пунктов), можно в считанные секунды получить от ЭВМ оптимальный маршрут движения (с учетом текущей ситуации).

Все указанные применения ЭВМ в быту уже нашли свою реализацию в мировой и частично отечественной практике, хотя процесс широкого внедрения вычислительной техники в быт находится еще лишь в начале своего развития (не считая карманных калькуляторов). Имеются и другие бытовые применения ЭВМ, однако и того, что уже перечислено, достаточно, чтобы оценить перспективы прогресса в этом направлении.

Глава X

АВТОМАТИЗАЦИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

10.1. Общие сведения об информационных технологиях

Технологические процессы, которые рассматривались в предыдущей главе, относились к материальному производству с материальными исходными и конечными продуктами. Имеется, однако, большое число процессов, где основной перерабатываемой продукцией является информация. Разумеется, перерабатываемая информация связана с определенными материальными носителями (до последнего времени главным образом с бумагой). Однако главную роль в процессах, о которых идет речь, играет не непосредственно сам носитель, а связанная с ним информация.

Подобные процессы, примерами которых могут служить проектно-конструкторские работы или редакционно-издательская деятельность, будем называть *информационными технологиями*. Особую роль среди информационных технологий играет технология организационного управления, которой будет посвящена отдельная глава.

Автоматизация информационных технологий приводит к вытеснению бумажных носителей, доводящему их роль до разумных (удобных человеку) пределов. Это положение следует особо подчеркнуть, поскольку абсолютно неправомерно связывать с безбумажной информатикой полное изгнание бумаги как носителя информации. Книги, газеты, журналы, будучи удобными и привычными для человека, надолго сохранят в основном свою нынешнюю форму (разве лишь обычная бумага будет заменена более прочным и более дешевым пластиковым эквивалентом). Максимальному сокращению подлежит прежде всего канцелярская переписка, а также многочисленное переписывание и перепечатка информации в процессе подготовки к публикации.

Чтобы высказанное положение прочнее закрепилось в памяти, мы начнем наше изложение с области, где оно проявляется в наиболее полном, понятном и законченном виде.

10.2. Автоматизация редакционно-издательской деятельности

Как известно, в традиционной технологии редакционно-издательского дела приходится многократно переписывать и перепечатывать материал в процессе его подготовки к изданию. Во-первых, это делает сам автор. Во-вторых, редактор правит полученный текст, после чего его обычно снова требуется перепечатать. Дополнительные исправления (чаще всего в сторону укорочения) могут производиться при монтаже номера газеты или журнала. Наконец, при типографском наборе, как правило, возникают ошибки, что вызывает необходимость изготовления верстки, ее корректировки и исправления замеченных ошибок набора. Лишь после этого начинается в достаточной мере автоматизированный процесс печати.

При комплексной автоматизации многие из указанных этапов попросту исчезают. Вручную (точнее, в диалоге с ЭВМ) происходит лишь процесс редактирования и монтажа материала. Выглядит это следующим образом. Материал для печати поступает от авторов либо на машинно-ориентированном носителе (в случае применения ими автоматизированных пишущих машинок, описанных в § 9.12), либо читается и автоматически вводится в ЭВМ с помощью читающих автоматов (если материал поступил в обычном машинописном виде). В случае, когда материал передается в редакцию по телетайпу, можно, не производя его распечатку, с помощью несложной дополнительной доработки телетайпной станцией осуществить его непосредственный ввод в ЭВМ.

Во всех случаях поступающий в редакцию сырой (неотредактированный) материал вводится в ЭВМ автоматически или почти автоматически (ручной труд сводится к загрузке поступивших машинописных текстов в читающий автомат или к постановке на редакционную ЭВМ текстов на машинных носителях). Введенный в ЭВМ материал распределяется между редакторами, каждый из которых работает за алфавитно-цифровым дисплеем с набором возможностей для редактирования и может управлять выводом нужного материала (не только редактируемого текста, но и различного рода справок из машинного редакционного архива).

Отредактированные тексты могут вызываться в преобразованном виде на специальный дисплей для монтажа страниц или газетной полосы. Преобразование, о котором идет речь, состоит просто в замене абзацев прямоугольниками в соответствующем масштабе. Аналогичное преобразование делается с рисунками. Подобный компоновочный дисплей должен сопровождаться средствами для перемещения абзацев (например, световым пе-

ром). В случае, если материал не вмещается в требуемый формат, принимается решение о дополнительном редактировании того или иного материала. Этот процесс может повторяться несколько раз.

После окончания монтажа (и редактирования) материал поступает в автоматические наборные машины (непосредственно от ЭВМ или с помощью переноса приготовленных на ЭВМ копий на машинных носителях, воспринимаемых наборными машинами). Ввиду большой надежности автоматики необходимость в корректировке верстки практически исключается, хотя в случае большой ответственности печатаемого материала просмотр пробного экземпляра в принципе не исключается.

Редакционный автоматизированный архив, о котором шла речь выше, в современных автоматизированных редакциях (например, редакции газеты «Нью-Йорк Таймс») может содержать все материалы, печатавшиеся ранее в газете. Строится он обычно как информационно-поисковая документальная двухуровневая система дескрипторного типа (см. гл. VI). Благодаря тому, что в словарь дескрипторов обычно включаются все собственные имена (фамилии, географические названия и др.), он может достигать огромных размеров (многие десятки и даже сотни тысяч дескрипторов). Благодаря наличию подобного архива редакция может вести статистику опубликованных материалов и соответственным образом планировать новый материал. Разумеется, архив полезен и для многих других целей (унификация транскрипций иностранных имен и названий, проверка правильности ссылок и т. п.).

Заметим, что в рассматриваемой системе бумажная информация фигурирует только на входе (и то не всегда) и на выходе. Вся же промежуточная бумажная стихия полностью уничтожается.

10.2.1. Автоматизация издательств переводной литературы. При автоматизации издание переводов иностранной литературы в принципе не отличается от уже описанной схемы, если переводчик работает вне издательства, однако при этом в значительной мере сокращаются возможности автоматизации самого перевода. Для сколько-нибудь качественного автоматического перевода требуется ЭВМ большой мощности. При наличии такой ЭВМ (с соответствующим программным обеспечением) работа может строиться следующим образом. Подлежащее переводу издание разброшюровывается и с помощью читающего автомата его содержание вводится в редакционную ЭВМ. Редактор-переводчик (эти обязанности желательно совместить в одном лице) снабжается пультом с двумя алфавитно-цифровыми дисплеями. На первый выводится очередной абзац текста на исходном языке, а на второй — его перевод, сделанный машиной. Редактор-

переводчик читает оба текста и делает, в случае необходимости, соответствующие поправки.

Переведенный и отредактированный таким образом в диалоговом режиме текст перевода подвергается далее тем автоматизированным операциям, которые были описаны выше.

Отметим, что автоматизация перевода в диалоговом режиме имеет два весьма ощутимых преимущества. Во-первых, можно запускать в работу (и получать практическую отдачу) далекие от совершенства автоматические программы перевода. Важно лишь, чтобы их использование повышало производительность труда переводчика и убыстряло весь процесс издания переводной литературы. Второе преимущество состоит в том, что в диалоговом режиме возникает информация об относительной частоте редакторских поправок различного типа, что дает разработчикам программ автоматического перевода весьма ценные данные о желательных направлениях совершенствования программ и тем самым значительно убыстряет практический прогресс в этой области.

Речь здесь идет, разумеется, о переводах прозы, а не поэзии. Но даже в областях художественной прозы построение систем полностью автоматизированного высококачественного перевода требует вложения в систему большого объема знаний о мире. Решение этой задачи связано с общей проблемой искусственного интеллекта и разбирается ниже (в гл. XII).

10.3. Автоматизация научных исследований

Научные исследования, как известно, охватывают много различных форм работы. Это, прежде всего, экспериментальные исследования и обработка их результатов, дедуктивные построения, связанные зачастую со сложными вычислениями, подбор и ознакомление с необходимыми публикациями, отпечатками, авторскими свидетельствами и т. п. Наконец, появление ЭВМ породило принципиально новый метод исследования — *эксперименты с информационными моделями* (кибернетическое моделирование). Этот метод занимает промежуточное положение между классическим экспериментальным и классическим дедуктивным методами.

Его отличие от классического экспериментального метода состоит в том, что эксперименты производятся не с самими (материальными) объектами исследований, а с их описаниями на том или ином формализованном, понятном машине языке с соответствующим программным обеспечением. От классического дедуктивного метода его отличает то, что при наличии формализованного описания объекта исследований (что характерно

для дедуктивного метода) с этим описанием не производятся никакие дедуктивные построения (формальные преобразования, оформленные в соответствии с правилами логики доказательства). С описанием производятся именно эксперименты, которые с чисто формальной стороны совершенно аналогичны экспериментам с реальными (материальными) объектами.

Системы автоматизации, имеющие дело с перечисленными экспериментами, и, прежде всего, системы автоматизации экспериментов получили общее наименование *систем автоматизации научных исследований* (АСНИ). Иногда под это название подводятся и системы испытаний, описанные в предыдущей главе.

10.3.1. Автоматизация экспериментальных исследований. По существу, об этом предмете уже достаточно подробно говорилось в предыдущей главе, ибо автоматизация научных исследований — это в первую очередь автоматизация измерений. Необходимо подчеркнуть лишь несколько моментов, характерных именно для научных исследований. Во-первых, наряду с рутинными измерениями, для которых достаточны раз и навсегда заданные конфигурации измерительных систем, в научном эксперименте довольно часто приходится перекомпоновывать экспериментальные установки, а следовательно, и вносить соответствующие изменения в систему автоматизации измерений. В этом случае особые удобства представляют системы автоматизации, набирающиеся из отдельных модулей, как это имеет, например, место в упоминавшейся выше (в гл. II) системе Камак. Достаточно оперативной должна быть и система автоматизации программирования для быстрого изготовления необходимых комплектов программ (особенно для вторичной, наиболее сложной обработки).

Второй момент связан с *настройкой* экспериментальной установки и *управлением экспериментом*. В ряде случаев автоматизация именно этих участков работы является решающим фактором в повышении производительности труда экспериментатора. С подобным положением приходится сталкиваться, например, при экспериментах на ускорителях элементарных частиц. Установки по автоматизации управления экспериментом строятся в целом по тем же самым принципам, что и системы автоматизации управления классическими (материальными) технологическими процессами. Ввиду их огромного разнообразия, как и в случае АСУТП, их невозможно описать сколько-нибудь полно не только в пределах одного параграфа, но и целой книги. Общие же принципы построения АСУТП и соответствующий математический аппарат уже были описаны в предыдущей главе.

Третий момент связан с *планированием эксперимента*. Общей целью такого планирования является уменьшение затрат ресурсов и времени для получения результата, ради чего и про-

изводится эксперимент. Универсальных рецептов для оптимального планирования экспериментальных исследований, разумеется, не существует. Приведем лишь одну идею подхода к такому планированию. Предположим, что целью эксперимента является выбор из множества M одной из априорных гипотез, объясняющих исследуемое явление. Пусть тем или иным способом до начала эксперимента установлены априорные вероятности $p_1, p_2, \ldots, p_n$ справедливости указанных гипотез *). Степенью неопределенности нашего знания об объекте исследования (в данном случае о теории, которой должна стать одна из гипотез) является так называемая *энтропия* заданного распределения вероятностей, выражаемая формулой

$$H = -\sum_{i=1}^{n} p_i \log p_i, \tag{10.1}$$

где логарифм берется по любому фиксированному основанию, большему единицы, например (и чаще всего) по основанию 2.

Если из априорной энтропии H вычесть математическое ожидание H_k энтропии распределения вероятностей φ_j гипотез после проведения k-го эксперимента (из имеющегося априорного набора экспериментов), то в результате получим приращение информации

$$\Delta I_k = H - H_k.$$

Естественно, при прочих равных условиях (в частности, при равной сложности экспериментов), стремиться к выбору такого эксперимента, для которого это приращение получает максимально возможную величину: $\Delta I = \max\limits_k \Delta I_k$.

Четвертый момент, который должен учитываться при автоматизации научных исследований, особенно трудно повторяемых и дорогостоящих, состоит в желательности создания и хранения (на машинных носителях) банков данных первичных (необработанных) экспериментальных данных. Дело заключается в том, что по истечении определенного времени могут появиться новые, более совершенные методы обработки полученных данных, которые позволят добыть новые сведения из старого экспериментального материала (без новых экспериментов). Не исключено и такое положение, когда старые экспериментальные данные могут быть обработаны под новым углом зрения, который был в свое время неактуален. Например, данные геофизической разведки, обработанные первоначально с целью поиска

*) При отсутствии какой-либо априорной информации о величинах полагают $p_1 = p_2 = \ldots = p_n = 1/n$.

нефтяных месторождений, могут быть обработаны впоследствии с целью нахождения месторождений урана. Разумеется, накапливаемые (в цифровом виде) банки данных первичной экспериментальной информации должны быть соответствующим образом каталогизированы и сделаны доступными (лучше всего через сети ЭВМ).

10.3.2. Автоматизация вычислительных работ. Главным в развитии этого аспекта автоматизации является «доставка» вычислительных услуг непосредственно на каждое рабочее место научного работника. Для простейших вычислений эта задача решается развитием и широким распространением микрокалькуляторов. Для более сложных вычислений организуются рабочие места с алфавитно-цифровыми дисплеями, связанными с более мощной вычислительной техникой и с сетями ЭВМ. При этом программное обеспечение (прежде всего сервисная часть пакетов прикладных программ и языки манипулирования данными) должно строиться таким образом, чтобы каждый исследователь мог осуществлять диалог с машиной на привычном для него языке.

10.3.3. Автоматизация информационного обеспечения науки и техники. Через те же рабочие места, которые открывают доступ к вычислительным мощностям, исследователи должны иметь доступ к мощным банкам данных, накапливающим разнообразную научно-техническую информацию. Прежде всего, это всевозможные виды публикаций (статьи, монографии, отчеты, авторские свидетельства и т. п.). При современном состоянии вычислительной техники наиболее целесообразно подобные банки организовывать по двухконтурной схеме: в первом контуре (во внешней памяти ЭВМ) хранятся рефераты и поисковые образы, а полные тексты (во втором контуре) представлены в виде микрофишей с возможностью быстрого автоматического изготовления копий обычного (удобного для читателя) размера. В случае необходимости подобные копии могут высылаться почтой, а в пределах помещения, в котором расположен банк данных, возможно, и автоматической пневмопочтой.

Для быстрейшего и целенаправленного развития науки и техники принципиально важно, чтобы (автоматизированные) банки данных о готовых научных результатах дополнялись *банками проблем*. Подобные проблемы могли бы выставляться как непосредственно с производства, так и различными проектно-конструкторскими и исследовательскими коллективами. При наличии таких банков резко упрощается планирование научной работы, увеличивается ее отдача и взаимосвязь различных разделов науки и, наконец, может быть, самое главное — существенно облегчается внедрение результатов научных исследований в практику.

Среди автоматизированных банков данных должны занять свое место и банки экспериментальных данных, в том числе банки первичной экспериментальной информации, о которых говорилось выше. Само собой разумеется, что эффективное использование автоматизированных банков данных (включая международный обмен научно-технической информацией) предполагает широкое развитие сетей ЭВМ и автоматизированных рабочих мест (терминалов) для доступа к ним.

10.3.4. Автоматизация дедуктивных построений. В последние годы достигнуты определенные успехи в автоматизации дедуктивных построений (логического вывода) в диалоговом режиме. В первую очередь это касается формульных преобразований в алгебре и анализе.

Однако и в доказательствах сложных теорем (например, в знаменитой теореме о четырех красках или в некоторых проблемах теории чисел) вычислительные машины уже сыграли хотя пока еще и вспомогательную, но тем не менее весьма важную роль (такую, что обойтись без их помощи было бы практически невозможно). Более подробно проблемы автоматизации доказательства теорем разбираются ниже (в гл. XII). Здесь же мы специально подчеркнем, что реальное практическое значение сегодня имеет не задача полной замены математика-исследователя машиной, а задача увеличения производительности его труда. Поэтому главное направление в развитии автоматизации дедуктивных построений сегодня — это диалоговые системы.

10.3.5. Кибернетическое моделирование. Спектр задач, которые решаются методами кибернетического моделирования, весьма широк. Сегодня практически ни одна серьезная большая система (будь то в материальном производстве, организованном управлении или в обороне) не реализуется без предварительного всестороннего моделирования на ЭВМ. При моделировании технических систем, характеризующихся числовыми параметрами, широко используется аппарат систем обыкновенных дифференциальных и разностных уравнений, уравнений математической физики, теории вероятностей и других разделов современной математики. Описать весь этот аппарат в рамках одной книги, разумеется, невозможно. Заметим лишь еще раз (об этом уже говорилось в гл. V), что для описания таких систем и *машинных экспериментов* с этими описаниями созданы специальные алгоритмические языки (с трансляторами), в том числе отечественные языки *слэнг* и *недис*. Результатами машинных экспериментов обычно являются гистограммы экспериментальных законов распределения тех или иных параметров, характеризующих свойства моделируемых систем и, главное, их поведение в тех или иных условиях.

Не вдаваясь более подробно в описание этого вида моделирования, остановимся несколько подробнее на моделировании систем, характеризующихся так называемыми *качественными параметрами*, т. е. параметрами, способными принимать лишь конечные множества значений (например: отлично, хорошо, удовлетворительно, плохо или холодно, тепло, жарко). Такими параметрами в основном характеризуются различного рода биологические и социальные системы. Заметим, что, какие бы названия ни имели подобные параметры, их всегда можно заменить эквивалентными числовыми значениями, например, писать вместо «отлично» 5, вместо «плохо» 2 и т. п. Кроме того, не исключено, что отдельные характеризующие систему параметры могут принимать обычные числовые (непрерывные или дискретные) значения.

Заметим, что при наличии обычных числовых параметров их всегда можно «загрубить», заменив их (с соответствующим приближением) должным образом подобранными качественными значениями, например, вместо точной величины кровяного давления довольствоваться качественными выражениями «пониженное», «нормальное», «повышенное». В случае необходимости большей точности число градаций качественных оценок может быть увеличено. Учитывая приведенную возможность, мы ограничимся описанием одного вида модели, имеющей лишь качественные параметры, которая была впервые предложена автором в 1969 г.

Прежде чем приступить к моделированию той или иной системы с качественными параметрами, необходимо выполнить несколько этапов предварительной подготовки, объединяемых (вместе с этапом собственно моделирования) общим термином *системный анализ*.

На первом этапе производится так называемая структуризация изучаемого объекта (системы), т. е. разложение его на отдельные подсистемы, блоки и элементы. Например, если объектом изучения является человеческий организм, то в нем выделяются отдельные органы (печень, желудок, селезенка и т. д.), различного рода системы регулирования (например, система регулирования содержания сахара в крови или температуры тела). Выделяются факторы *внешнего воздействия* на изучаемый объект. Например, в случае, когда объектом является человеческий организм, такими факторами могут быть режим питания, различного рода лечебные и профилактические процедуры и т. д.

Следующим этапом является *параметризация* изучаемого объекта. На этом этапе четко устанавливается система параметров (с областями их значений), которые характеризуют как сам объект, так и все внешние воздействия на него. Заметим, что один структурный элемент может характеризоваться несколь-

ми параметрами. Например, такой элемент воздействия на человеческий организм, как утренняя гимнастика, целесообразно характеризовать по меньшей мере тремя параметрами: типом гимнастики, ее интенсивностью и, наконец, степенью регулярности ее исполнения (для каждого из этих параметров вырабатывается своя шкала качественных оценок).

Как структуризация, так и параметризация изучаемого объекта не имеют строгой регламентации своего выполнения. Их выполнение поэтому в значительной мере опирается на опыт и интуицию исследователя. Общие требования к этим этапам состоят в том, чтобы никакие важные для функционирования изучаемого объекта элементы и параметры не выпали из рассмотрения и чтобы степень детализации значений параметров давала нужный уровень «загрубления» модели. Ясно ведь, что, как правило, нет смысла точно учитывать в модели значения одного параметра (иметь много градаций для них), когда другой, не менее важный для функционирования модели параметр не может быть даже сколько-нибудь точно измерен (например, темперамент человека).

В процессе структуризации и параметризации объекта обычно выделяются *главные параметры*, которые имеют решающее значение для цели исследования. В случае человеческого организма такими параметрами могут быть, например, степень работоспособности, общее самочувствие и т. п.

После выполнения параметризации начинается основной этап составления модели — *установление зависимостей* между параметрами. Вид таких зависимостей определяется в значительной мере видом самих параметров. Если параметры выражаются обычными числами (вещественными или целыми), то между ними могут существовать детерминированные или стохастические (случайные) зависимости, выражаемые строгими математическими закономерностями.

Для моделей с качественными параметрами зависимости — это алгоритмы (системы правил), позволяющие по распределению вероятностей значений параметров в предшествующие моменты дискретного времени вычислить такие распределения в любой текущий момент времени. Чаще всего такие зависимости даются *логиковременны́ми правилами* вида: если параметры $x_1(t)$, $x_2(t)$, ..., $x_n(t)$ в моменты времени $t-1$, $t-2$, ..., $t-k$ принимали некоторые заданные (вообще говоря, неоднозначно) значения, то параметр $x_m(t)$ в момент времени t примет значения a_1, a_2, ..., a_l соответственно с вероятностями p_1, p_2, ..., p_l. Если условия в левой части правила не выполняются, то обычно предполагается, что данное правило предполагает сохранение для параметра $x_m(t)$ в момент времени t того распределения вероятностей, которое существовало для него в предшествующий момент

времени $t-1$. Неоднозначность, о которой идет речь, означает, что для любого параметра $x_i(t)$ $(i=1, 2, \ldots, n)$ в любой момент времени $t-1$, $t-2$, ..., $t-k$ может допускаться не одно, а несколько различных значений.

Например, может быть высказано правило: если $x_1(t-1)=1$ или 2, а $x_2(t-1)=x_2(t-2)=3$, то $x_3(t)$ с вероятностью 0,8 примет значение 0 и с вероятностью 0,2 — значение 1 (поскольку $0{,}8+0{,}2=1$, то остальные значения параметра $x_3(t)$, если таковые существуют, имеют вероятности, равные нулю).

Если для $x_3(t-1)$ вероятности равенства 0 и 1 были равны 0,5, а вероятности равенств $x_1(t-1)=1$, $x_1(t-2)=2$, $x_2(t-1)=3$, $x_2(t-2)=3$ равны соответственно 0,1; 0,2; 0,6; 0,5, то событие $x_2(t-1)=x_2(t-2)=3$ имеет вероятность $p_1=0{,}6\cdot 0{,}5=0{,}3$. Событие $x_1(t-1)=1$ или $x_2(t-1)=2$ имеет вероятность $p_2=0{,}4+0{,}2=0{,}6$ (поскольку, в отличие от классической математической логики, в данном случае мы имеем дело с исключающим «или», т. е. не допускающим одновременное выполнение обоих условий $x_1(t-1)=1$ и $x_1(t-1)=2$).

Вероятность p_3 выполнения условия, стоящего в левой части нашего правила, будет равна, очевидно, произведению p_1p_2, т. е. 0,18. Тогда, в соответствии с данным правилом, вероятность того, что $x_3(t)=0$, будет равна $0{,}18\cdot 0{,}8+(1-0{,}18)\cdot 0{,}5=0{,}144+0{,}41=0{,}554$, а вероятность того, что $x_3(t)=0$, будет равна $0{,}18\cdot 0{,}2+(1-0{,}18)\cdot 0{,}5=0{,}036+0{,}41=0{,}446$.

При наличии нескольких (возможно, даже противоречащих друг другу) правил для вычисления распределения вероятностей одного и того же параметра $x_m(t)$ им придаются нормированные весовые коэффициенты λ_i (сооставляющие в сумме единицу). Для вероятностей каждого значения прогнозируемого параметра, вычисленных (по соответствующим правилам) аналогично тому, как это сделано в только что разобранном примере, находится средневзвешенное значение $p=\sum_{i=1}^{s}\lambda_i p_i$ (где сумма берется по всем правилам, определяющим значения параметра $x_m(t)$).

Все правила, устанавливающие зависимость от предыстории значений всех параметров, характеризующих исследуемый объект, записываются в определенном формализованном входном языке и вводятся в память ЭВМ. Туда же вводится необходимый отрезок предыстории изменения значений этих параметров до текущего момента (с которого начинаются эксперименты с моделью), а также значения всех параметров внешних воздействий (или законы их изменения) на все время, в течение которого исследуется поведение моделируемого объекта. Используя эти данные, специальная система (пакет) программ определяет (в соответствии с описанной выше процедурой) законы распределе-

ния вероятностей значений параметров, характеризующих объект последовательно для моментов времени $t = 1, 2, \ldots, T$. Распределения (прежде всего главных параметров) в момент времени $t = T$ выдаются (в качестве результата машинного эксперимента) на дисплей или на печать. Разумеется, при желании можно вывести распределения и в другие моменты времени.

Пакет для экспериментов с моделью строится таким образом, чтобы было возможно оперативно (в диалоговом режиме) изменять любые элементы модели. В первую очередь это касается внешних воздействий. Однако в случае необходимости можно столь же оперативно вносить изменения в систему правил, весовые коэффициенты и т. п. Тем самым создается возможность многократного повторения эксперимента в различных условиях. Так, построив модель человеческого организма (и притом индивидуального, а не среднего), можно на машинных экспериментах подобрать наилучшие последовательности внешних воздействий, с тем чтобы в возможно более короткий срок и по возможности без нежелательных побочных эффектов излечить его от той или иной болезни (если это вообще в принципе возможно). Аналогично, построив модель сложной социальной системы, можно подобрать в машинном эксперименте такое управление, которое приводит систему в желательное состояние. Очень важно при этом, что в машинном эксперименте (особенно над биологическими и социальными объектами) можно воспроизводить такие условия, которые в реальном эксперименте получить невозможно прежде всего в силу связанного с экспериментом риска, а также, возможно, и просто в силу длительности и высокой стоимости экспериментов на реальных объектах.

Разумеется, успех машинного моделирования и практическая ценность его результатов определяются в первую очередь качеством самой модели. Поэтому необходимо сказать несколько слов о том, каким образом строятся системы определяющих модель правил. Если не удается найти какие-либо правила, справедливость которых можно было бы установить с точностью, с которой устанавливается справедливость физических теорий, то в запасе остается, во всяком случае, *метод коллективных экспертных оценок*, который мы будем называть также *методом коллективного мозга*.

Сущность этого метода в основе своей предельно проста и понятна: правила, определяющие модель, задают специально подобранные люди — эксперты, исходя из собственного запаса знаний, опыта и интуиции. Возражение, что людям свойственно ошибаться и что поэтому ценность подобных моделей сомнительна, на первый взгляд звучит достаточно убедительно. Однако вспомним, что лечение живых людей, как правило, происходит на основании опыта и интуиции одного человека (лечащего вра-

ча) или весьма небольшого коллектива (консилиума) врачей. В описанной же модели имеется возможность сконцентрировать на выборе метода лечения больного (и притом, подчеркнем еще раз, индивидуального, а не среднестатистического больного) знания, опыт и интуицию многих тысяч (а если надо, то и больше) лучших специалистов. При этом особо подчеркнем, что каждый из привлекаемых экспертов отвечает за свою группу параметров, так что при огромном общем числе привлекаемых экспертов за тот или иной конкретный параметр отвечает относительно небольшое число (обычно от 3 до 12) действительно лучших специалистов в данной области.

Важнейшая особенность модели, таким образом, состоит в возможности неограниченного дальнейшего развития специализации знаний без потери *целостного* взгляда на объект исследования. Что же касается сохранения в модели специфических *индивидуальных* черт исследуемого объекта, о котором уже дважды упоминалось выше, то дело здесь просто в выборе соответствующих параметров, которые эту индивидуальность характеризуют. Применительно к человеку такими параметрами могут служить типы наследственности и темперамента, характер работы, болезни и травмы, перенесенные в прошлом, т. д. Кроме того, конкретные значения того или иного параметра общего характера (например, величины кровяного давления) также характеризуют индивидуального больного и выбираются в модели применительно именно к нему, а не к кому-либо другому.

Очень важным является вопрос о наличии (особенно в медицине и социологии) различных школ, а зачастую и прямо противоположных взглядов по одному и тому же вопросу. Предложенной модели в принципе это ни в коей мере не вредит, поскольку речь идет не о нахождении единственно правильного результата, а лишь о распределении вероятностей возможных исходов. Более того, чтобы не впасть в одностороннее суждение при наличии разных школ по одному и тому же вопросу, к формулировке правил, составляющих модель, следует привлекать лучших специалистов *всех* этих школ. Правда, наличие прямо противоположных правил увеличивает энтропию ответа (т. е. меру нашего незнания об исследуемом объекте). Но каждый согласится, что лучше получить расплывчатую картину, *объективно обусловленную* недостаточным уровнем знания об исследуемом объекте, чем впасть в иллюзию ложного детерминизма.

В связи с последним замечанием отметим еще одну важную особенность описываемой модели. В состав ее программного обеспечения включаются специальные программы, которые анализируют, какого рода вопросы в наибольшей мере способствуют росту неопределенности результатов работы с моделью. С этой

целью должны быть измерены уменьшения энтропии конечных (на момент времени T) распределений при замене группы противоречащих друг другу правил (относящихся к предсказанию значений одного и того же параметра) тем или иным строго детерминированным правилом. По величине такого уменьшения можно ранжировать по относительной важности задачи уточнения закономерностей зависимостей одних параметров от других. Тем самым появляется возможность целенаправленного управления развитием соответствующей области знания.

Важным является вопрос о выборе относительных величин весовых коэффициентов λ_i, с помощью которых «взвешиваются» оценки, даваемые различными экспертами. Не вдаваясь в подробности, заметим лишь, что при этом учитываются оценки экспертов, даваемые ими своему уровню компетентности в задаваемых им вопросах, и места по уровню компетентности, которые они дают своим коллегам.

Используется также *ретроспективный анализ,* т. е. применение данного набора правил для моделирования случаев, уже имевших место ранее и соответствующим образом документированных. По мере накопления банков данных в соответствующих областях знания этот метод будет приобретать все большее значение. Его можно применять не только для корректировки весовых коэффициентов уже имеющихся в модели правил, но и для пополнения ее новыми правилами. Программное обеспечение должно позволять выполнять подобные операции в диалоговом режиме без существенных задержек в работе системы моделирования. Наличие подобной возможности позволяет, в частности, вводя нулевые весовые коэффициенты для оценок, данных теми или иными экспертами, исключать тем самым учет их мнений из процесса моделирования.

Специальные программы, анализирующие динамику измерений распределений вероятностей значений главных параметров моделируемого объекта во времени, позволяют находить те моменты времени, в которые происходят наиболее резкие их повороты к худшему. Тем самым внимание исследователя концентрируется на тех моментах, когда особенно важно изменить намеченные внешние воздействия на объект моделирования с целью достижения лучших результатов управления. Благодаря этому достигается убыстрение процесса диалоговой оптимизации управления моделируемым объектом.

Хотя мы вели наше изложение применительно к моделям с качественными параметрами, легко заметить, что все описанные приемы применимы в принципе к моделям с параметрами любой природы. Ведь в конечном счете при реальных вычислениях на ЭВМ (если не иметь в виду аналитические методы) всегда имеют дело с конечным числом значений любого параметра, будь он

целочисленным или непрерывным. Поэтому в конечном счете (при численных расчетах) с любым параметром можно оперировать как с качественным в нашем смысле (т. е. принимающим конечное число значений). Другое дело, что языки для описания зависимостей между параметрами с очень большим числом возможных значений, равно как и языки для описания законов распределения вероятностей этих значений, целесообразно изменить по сравнению с языками, ориентированными на «истинно» качественные параметры (с малым числом значений). То же самое касается и методов (программ) для фактического нахождения распределений вероятностей значений параметров.

10.4. Автоматизация обработки данных в медицине

В 70-е годы начался процесс быстрого проникновения электронной вычислительной техники в медицину. Особенно усилился он после появления и широкого распространения микропроцессоров. Микропроцессоры и микро-ЭВМ, встроенные в измерительную аппаратуру, позволили резко ускорить выполнение необходимых для медицины исследований, улучшить их, а в ряде случаев получить принципиально новое качество. С помощью ЭВМ в настоящее время производится быстрый и качественный анализ крови, анализируются кардиограммы и энцефалограммы, делаются гистологические исследования и многое другое.

Рентгеновский томограф представляет собой установку, где узконаправленное рентгеновское излучение обследует поворачивающееся тело. Для этого излучающая рентгеновская головка перемещается по окружности кольца, в которое постепенно мелкими шагами вдвигается исследуемое тело (благодаря чему исследуется много круговых разрезов). Датчик, воспринимающий прошедшее сквозь тело рентгеновское излучение, через аналого-цифровой преобразователь подключен к ЭВМ, которая с помощью достаточно сложных вычислений дает картину изменения плотности тканей (с точностью до 1%) на каждом из участков исследуемого тела.

Результаты этих расчетов выводятся на экран дисплея (чаще всего — цветного) в виде наглядных изображений любого интересующего врача разреза. Обработка на ЭВМ использует то обстоятельство, что благодаря круговому перемещению излучателя каждая точка разреза многократно пронизывается лучом с разных направлений. Это позволяет применить интегрирование и устранение помех, благодаря чему резко усиливается контрастность и четкость изображения. Использование же при отображении на дисплее разных цветов для тканей разных плотностей

позволяет врачу особенно наглядно видеть скрытые дефекты и зачатки болезненных изменений тех или иных органов.

Главной частью звуковизора является звукоизлучающая (и одновременно звукоприемная) головка, прикладываемая к исследуемому участку тела. В головке находятся излучатели ультразвука — пьезокристаллы, посылающие короткие импульсы ультразвука в глубь тела по различным направлениям. Отраженные от границ, разделяющих друг от друга различные ткани, импульсы воспринимаются чувствительными элементами головки. Измеряются амплитуда и время задержки прихода отраженных импульсов. Обрабатывая результаты этих измерений, ЭВМ восстанавливает (с точностью до 1 мм) пространственную картину исследуемого объема и отображает ее на экране дисплея.

При этом с помощью так называемого *стробирования* (т. е. отсекания всех сигналов, кроме тех, которые пришли в заданные временны́е интервалы) можно удалить из демонстрируемого на экране дисплея изображения все, что находится дальше или ближе исследуемого объекта. В результате врач может увидеть на экране бьющееся человеческое сердце, которое совершенно не скрывает грудная клетка. Дисплей звуковизора обычно снабжается перемещаемым с пульта *курсором* (световой точкой). Подводя курсор к тем или иным точкам изображения, можно с помощью ЭВМ практически мгновенно вычислять координаты этих точек, а по ним — размеры и положение различных интересующих врача объектов (органов, опухолей и т. п.).

Головки звуковизоров выполняются либо в виде плоского поля неподвижных излучателей (сегодня до 400 штук), переключаемых электронным коммутатором, либо в виде перемещающегося механически излучателя или, чаще всего, вытянутой в прямую линию группы излучателей (что позволяет осуществлять механическое сканирование не в двух, а лишь в одном направлении).

Важное значение для развития медицины имеет автоматизация ведения медицинской документации. Прежде всего, речь идет о первичной документации, т. е. об историях болезней. Создание автоматизированных банков историй болезней создает большие удобства для лечащих врачей, разумеется, при том непременном условии, чтобы они были снабжены средствами доступа к банкам (лучше всего дисплеями) непосредственно на рабочих местах. Необходима также максимальная автоматизация процесса пополнения банка новыми данными, возникающими после каждого очередного обследования больного. С этой целью современная диагностическая аппаратура, снабженная микро-ЭВМ, может быть относительно легче доработана для автоматической регистрации результатов обследования на носителе, пригодном для непосредственного переноса на ЭВМ, ведущих автоматизированный банк

данных. Возможно также и непосредственное подключение измерительной (диагностической) аппаратуры к ЭВМ для автоматического пополнения (актуализации) банка данных.

Разрабатываются универсальные измерительные кресла; усадив в него больного, можно сразу снять и автоматически передать в ЭВМ много данных (кровяное давление, температуру, кривые, характеризующие пульс, шумы сердца, магнитограммы и т. д.). Другой источник для пополнения банка данных — это ответы самого больного на те или иные стандартные вопросы (о самочувствии, ранее перенесенных болезнях и т. п.). В мировой практике постепенно устанавливается порядок, что такие вопросы (с перечнем возможных ответов) высвечиваются автоматически в определенном порядке на дисплее, установленном в приемной врача. Пациенту, ожидающему приема, необходимо при очередном вопросе нажать клавишу с номером нужного ответа. Чтобы идентифицировать личность пациента, перед началом опроса ЭВМ должна получить его регистрационный номер (номер истории болезни). Ввод этого номера в принципе можно поручить самому пациенту (после ввода номера на экране для контроля желательно иметь не только этот номер, но и имя пациента). Однако более удобен и, главное, более защищен от ошибок другой способ, когда у пациента имеется регистрационный жетон со всеми необходимыми для его идентификации данными. На пульте рядом с дисплеем монтируется специальный *регистратор*, автоматически считывающий данные с этого жетона.

Подобный программно-управляемый диалог можно употреблять и при вводе данных, формируемых врачом. С этой целью на экране дисплея у врача высвечивается наименование очередного требуемого действия (например, прослушивание шумов сердца) и перечень всех мыслимых результатов этого действия. Выполнив действие, врачу достаточно нажать клавишу с номером установленного им результата (признака), чтобы этот результат оказался автоматически занесенным в историю болезни.

Врачу при этом предоставляется широкий выбор возможных процедур осмотра, которые он запускает (набором номера требуемой процедуры) в зависимости от симптоматики заболевания. Если врач не удовлетворяется ни одной из имеющихся стандартных процедур, он может применить любую предпочитаемую им индивидуальную процедуру. В этом случае, правда, ему потребуется гораздо больше времени, чтобы ввести в ЭВМ (с клавиатуры дисплея) даже специальные укороченные коды сделанных им в результате осмотра заключений.

Для упрощения работы врача в этом случае могут использоваться специальные многоклавишные пульты или их имитация на дисплее с возможностью отметки обнаруженного симптома или заключения световым пером, а также устройства графиче-

ского ввода, на поле которого нанесены наименования возможных заключений. Возможен и другой путь, когда сделанные заключения записываются на диктофон, а затем вводятся в ЭВМ вспомогательным персоналом в специальном диктофонном центре (он может обслуживать не одно, а несколько лечебных учреждений).

Кроме того, врачу должна быть предоставлена возможность с помощью персонала, обслуживающего автоматизированный банк данных, включить свою индивидуальную процедуру осмотра в число стандартных и при повторных ее применениях пользоваться уже описанным выше упрощенным (программно-управляемым) вводом. Само собой разумеется, что врач по своему желанию может вызвать на экран своего дисплея любой раздел истории болезни осматриваемого пациента, в том числе и разделы, заполненные в процессе текущего осмотра. Нужно подчеркнуть, однако, что это не относится к записям, сделанным на диктофон. В этом состоит основной недостаток диктофонного метода: сделанные с его помощью записи смогут быть вызваны врачом на дисплей лишь после их обработки в диктофонном центре. Правда, в последние годы достигнуты значительные успехи в автоматизации ввода произвольной информации в ЭВМ непосредственно голосом. Однако подобные системы пока еще достаточно сложны, дороги и потому не получили еще сколько-нибудь широкого применения *).

10.4.1. Автоматизация диагностики и выбора метода лечения. Автоматизация диагностики в принципе может быть организована тем же самым методом коллективного мозга, который был описан выше (см. п. 10.3.5). Единственное отличие состоит в том, что в правилах, определяющих зависимость (качественных) параметров, которые характеризуют диагноз, от параметров, характеризующих симптоматику и результаты обследований, отсутствует время. Однако, если требуется не просто установить диагноз, а дать прогноз развития заболевания, то описанная в п. 10.3.5 методика может действовать без всяких изменений. То же самое касается и коллективного (автоматизированного) консилиума, на примере которого и была объяснена указанная методика. Заметим лишь, что при автоматизации диагноза большую роль могут сыграть не только модели с качественными параметрами, но и различного рода количественные модели и подмодели.

При автоматизации врачебных назначений (с коллективным консилиумом или без него) существенное значение имеет нали-

*) Имеются относительно простые системы автоматизации ввода отдельных слов из ограниченного словарного набора. Однако такие устройства фактически (после стандартизации процедуры) эквивалентны уже описанному программно-управляемому вводу.

чие у автоматизированного места работы врача возможности автоматического изготовления *твердых копий*, т. е. зафиксированных на бумаге рецептов и предписаний врача, которые больной может забрать с собой. При этом текст назначения вызывается (по короткому коду) из памяти ЭВМ на экран дисплея, прочитывается и в случае необходимости корректируется врачом*). Лишь после этого (в результате нажима на специальную кнопку) изготовляется копия текста с экрана дисплея. При дальнейшем развитии безбумажной информатики возможна прямая передача выписанных рецептов на автоматизированные рабочие места в аптеке.

10.4.2. Автоматизация медицинской статистики. При наличии автоматизированных банков данных всей первичной медицинской информации их объединение в сеть дает возможность автоматизировать любые мыслимые формы медицинской статистики. В организации этой работы нет никаких принципиальных отличий от организации других форм общегосударственной и территориальных систем учета. Поэтому мы не будем останавливаться на более детальной ее характеристике. Отметим лишь, что наличие возможности быстро выполнять анализ данных по многим миллионам историй болезней в любых *разрезах* позволяет подвести под различного рода медицинские рекомендации и методы несравненно более солидную базу, чем это делается сейчас на основании наблюдений и сравнения нескольких сотен или в лучшем случае нескольких тысяч различных больных.

10.4.3. Автоматизация управления медицинскими учреждениями. Автоматизация учета различного рода ресурсов, которыми располагает медицина (персонал, помещения, оборудование, лекарства и др.), позволяет существенно улучшить использование этих ресурсов. Такой эффект наблюдается у нас уже сейчас при автоматизации процесса распределения путевок в профсоюзные санатории в ряде районов страны. Задачи распределения решаются в таких системах в диалоговом режиме и, как правило, не требуют особо сложных математических методов (хотя задача оптимизации распределения путевок с учетом всех возможных требований представляет собой достаточно сложную задачу дискретного программирования). Важнейшее значение в подобных системах имеет дисциплина актуализации сведений об освобождении ресурсов (например, о досрочном отъезде отдыхающих) и о любых других изменениях (например, необходимости экстренного ремонта помещений), которые могут повлиять на распределение (путевок, мест в больницах и т. п.).

---

*) Это делается тем же способом, что и редактирование текстов, описанное в § 10.2.

Организация заказов и поставок расходуемых ресурсов (например, лекарств) делается в соответствии с общими принципами управления запасами, которые описываются в следующей главе.

Заметим еще, что принципы нахождения и ранжирования узких мест в нашем знании о моделируемых объектах, изложенные в п. 10.3.5, будучи применены к моделям человеческого организма в норме и в различных формах патологии, дают объективный исходный материал для ранжирования целей развития медицинской науки, т. е. в конечном счете для управления ее развитием. При этом влияние разнообразия форм патологии и других индивидуальных особенностей человеческого организма на ранжирование целей можно устранить, воспользовавшись медицинской статистикой для оценки степени общественной важности тех или иных из этих особенностей (прежде всего степени распространенности).

10.5. Автоматизация обучения

Во многих странах (особенно в Японии) уделяется большое внимание автоматизации обучения с помощью ЭВМ на всех уровнях — от детских садов до высшей школы. Основой служат так называемые *программированные учебники* и учебные пособия, которые могут принести немалую пользу и сами по себе. Отличие программированного учебника от обычного состоит прежде всего в более строгом дозировании изучаемого материала. Каждая порция снабжается хорошо продуманным набором контрольных вопросов и задач, на которые заранее заготавливаются серии возможных ответов, причем лишь один является правильным. Остальные ответы подбираются в соответствии с характерными ошибками, которые обычно делаются учащимися.

Прочитав очередную порцию материала и осмыслив ее, учащийся переходит к первому контрольному вопросу (или задаче), обдумывает его (или решает задачу) и выбирает из приведенных ответов тот, который представляется ему правильным. Осуществив выбор, он заглядывает в конец учебника и при правильном ответе переходит к следующему контрольному вопросу. В случае неправильного ответа, в зависимости от вида сделанной ошибки, учебник объясняет сущность этой ошибки и, если надо, отсылает учащегося к повторному изучению (или просмотру) уже изученных им ранее порций материала.

Из сказанного ясно, что самостоятельная работа с программированным учебником требует большой самодисциплины (требуется удержаться от заглядывания в конец учебника до решения контрольной задачи, строго следовать указаниям о возвра-

щении к уже пройденному материалу и т. д.). Поэтому на практике лучше всего организовать самостоятельную работу с указанными программированными учебниками (без заключительной оценочной и направляющей частей) на специальных автоматизированных местах, лучше всего, снабженных дисплеями с клавиатурой, присоединенными к ЭВМ, которая управляет процессом обучения. Поскольку обычные дисплеи пока еще достаточно дороги, для целей обучения часто используются упрощенные специализированные устройства отображения информации. На каждом дисплее ЭВМ высвечивает задание, которое должен выполнять сидящий за дисплеем (и снабженный учебником) учащийся (название порции материала, который надо изучить, или номер контрольного примера, который надо решить). Окончив выполнение задания, учащийся сообщает об этом ЭВМ, нажимая специальную кнопку, если это конец изучения очередной порции, или номер правильного ответа, если это было решение контрольного примера. После этого учащийся видит на дисплее либо просто следующее задание (обычно со словами одобрения), либо объяснение совершенной ошибки и опять-таки следующее задание.

При этом ЭВМ автоматически регистрирует совершенные ошибки и по программе, параметры которой определены учебником (и, возможно, скорректированы преподавателем), выставляет оценки учащимся. Во время самостоятельной работы учащихся преподаватель с помощью специального пульта может по желанию контролировать работу, выполняемую каждым учащимся, и получаемые им оценки, оказывать им дополнительную помощь, давать дополнительные разъяснения т. д. Возможно также, что некоторые разделы учащиеся изучают не по учебнику, а в устном изложении преподавателя, особенно если изучение сопровождается демонстрацией наглядных материалов, проведением опытов и т. д. В развитых системах автоматизации обучения предусматривается автоматическое управление показом демонстрационных материалов, в том числе отрывков учебных кинофильмов (иногда с их выводом в нужное время на индивидуальные дисплеи, хотя сегодня это обходится еще достаточно дорого).

Американской фирмой «Контрол Дейта» (Control Data) разработана система для обучения чтению в детских садах: на индивидуальные дисплеи выводятся картинки, изображающие различные предметы и явления, и обозначающие их английские слова. Ребенок пальцем указывает на дисплее слово, обозначающее выведенное изображение и получает (в наглядном виде) на дисплее ту или иную реакцию в зависимости от правильности или ошибочности данного ответа. В системе используется плазменный плоский дисплей (см. гл. IV), так что из ЭВМ на него выводятся лишь слова, а картинки высвечиваются на экран

дисплея сзади с помощью дешевого, хотя и управляемого от ЭВМ диапроектора.

При автоматизированном обучении число контрольных вопросов и задач к каждому разделу обычно стремятся увеличить таким образом, чтобы иметь возможность давать разные вопросы различным учащимся. При этом, разумеется, нет необходимости, чтобы каждый учащийся ответил на все контрольные вопросы. Более способным ученикам могут быть предложены для проработки дополнительные разделы или может быть задано больше контрольных вопросов (в том числе повышенной сложности), менее же способные могут ограничиться лишь необходимым минимумом.

В таком индивидуальном подходе к учащимся, возможно, заключается самое главное преимущество автоматизированного обучения. Ведь при традиционном (групповом) методе обучения преподаватель вынужден ориентироваться на некоторый средний уровень. В результате более способные ученики недорабатывают и скучают во время занятий, а менее способные перенапрягаются или попросту не усваивают всего требуемого материала. Второе (не менее важное) преимущество автоматизированного обучения состоит во всеобщем и постоянном *контроле* и объективной оценке успехов каждого, в отличие от выборочного периодического контроля, характерного для традиционного метода обучения. Указанные преимущества обеспечивают, как правило, гораздо большую эффективность автоматизированного обучения по сравнению с традиционным.

Вместе с тем нельзя не видеть и серьезные недостатки автоматизированного обучения в том варианте, который был только что изложен. Ясно, например, что далеко не одно и то же, выбрать правильный ответ на вопрос из уже готовых формулировок или самому сформулировать его. В изложенном варианте автоматизированная система не учит самостоятельно формулировать свои мысли, она не способна оценить ход решения задачи, а оценивает лишь результат этого решения. Таким образом, никак не оценивается оригинальность подхода к решению задачи, хотя, разумеется, такая оценка чрезвычайно важна для развития творческих способностей учащихся. Поэтому на нынешнем этапе развития автоматизированные системы обучения должны применяться в разумной пропорции с традиционными методами.

Заметим, что преодоление указанных недостатков нынешнего этапа автоматизации обучения обеспечивается развитием работ по искусственному интеллекту. Прежде всего — это работы по функционированию ЭВМ в естественных человеческих языках (с распознаванием смысла), а также работы по диалоговым системам автоматизации дедуктивных построений (логического вывода), о которых говорится в гл. XII. В этом направлении уже

достигнуты определенные результаты, однако сегодня еще несколько рано говорить об их широком внедрении в практику автоматизированного обучения.

Правда, когда учащийся излагает свои ответы на понятном для современных ЭВМ языке, решение всех указанных вопросов резко упрощается и реализуется на практике уже сегодня. Речь идет об автоматизации обучения программированию на различных входных языках ЭВМ (включая, разумеется, и обучение самим этим языкам). Подобные системы (например, отечественная система СПОК) ускоряют изучение входных алгоритмических языков современных ЭВМ в несколько раз, обеспечивая одновременно более прочное и глубокое овладение изучаемым материалом.

10.5.1. Управляемые ЭВМ — тренажеры. Системы автоматизированного обучения особенно эффективны тогда, когда речь идет об обучении определенным навыкам, например, пилотирования самолетов. Подобные системы называют обычно *тренажерами.* В простых случаях они могут быть выполнены средствами обычной автоматики, в сложных — основываются на использовании ЭВМ (порою даже очень мощных).

Поскольку число типов тренажеров очень велико, опишем (и притом кратко) только один из них, а именно авиационный (самолетный). Наиболее заметной частью такого тренажера является кабина управления, которая берется прямо из самолета требуемого типа. С помощью телевизионных установок на фонаре кабины воспроизводится обстановка, которую тренирующийся пилот должен был бы видеть в реальном испытательном полете. В простейшем случае воспроизведение обстановки производится передающей телевизионной камерой, имеющей возможность двигаться (с помощью управляемой системы подвесок) над моделью местности (включая модель аэродрома), выполненной в соответствующем масштабе. Динамика самолета и закономерности отображения обстановки (визуально через телекамеры и через систему установленных в кабине приборов) моделируются с помощью вычислительного комплекса, который чаще всего в тренажерах данного типа состоит из аналоговой и цифровой ЭВМ.

Воздействие пилота на находящиеся в кабине органы управления через соответствующие преобразователи передается на вычислительный комплекс и в преобразованном виде (в соответствии со смоделированной схемно-программной динамикой самолета и меняющейся окружающей обстановкой) отображается пилоту визуально и на приборную доску. Для полноты иллюзии в систему отображения включаются управляемые через вычислительный комплекс динамики, воспроизводящие сопровождающие действия пилота звуковые эффекты (шум заводных двигателей, стук убирающегося шасси, грохот взрыва при «аварии» и т. п.). В некоторых случаях пилотская кабина тренажера укрепляется на

длинной штанге, приводящейся в движение мощными сервоприводами. Управление этими сервоприводами также осуществляется вычислительным комплексом через модель динамики самолета, так что пилот будет испытывать на себе (в пределах подвижности конца штанги) результаты своих собственных действий (толчки, крены, перегрузки и т. п.).

Огромным преимуществом тренажеров является возможность многократного воспроизведения опасных и аварийных ситуаций, что, разумеется, нежелательно или даже вовсе невозможно в реальных испытательных полетах. Кроме того, несмотря на довольно большую стоимость современных авиационных тренажеров, «полеты» на них обходятся все же намного дешевле, чем реальные испытательные полеты (особенно на современных сверхзвуковых или большегрузных самолетах). Прежде всего этим обстоятельством, а также высокой эффективностью (в сочетании с безопасностью) обучения, объясняется широкая распространенность авиационных тренажеров. То же самое (а порою и в значительно большей мере) относится к тренажерам для других типов сложной современной техники (особенно космической). Перед тем как садиться за реальный пульт управления, всевозможные ситуации тщательно проигрываются на соответствующих тренажерах.

Появление микропроцессоров и дешевых микро-ЭВМ открыло широкую дорогу развитию тренажеров для обучения навыкам управления практически всеми машинами и технологическими процессами, в том числе и такими, для которых это ранее было экономически невыгодно.

10.6. Автоматизация проектно-конструкторских работ

Хотя автоматизация сложных проектно-конструкторских расчетов — ровесница вычислительной техники (первые ЭВМ делались в первую очередь для этой цели), однако *системы автоматизированного проектирования* (САПР) появились значительно позднее. Главное отличие этого нового этапа — переход от автоматизации расчетов к *комплексной автоматизации* всего процесса проектирования в целом. В полном виде современные САПР предусматривают полную автоматизацию всех расчетных и всех чертежных работ (кроме, быть может, эскизных набросков, выполняемых от руки, без всяких чертежных инструментов), а также практически полную (за исключением подписей и неформализованных пояснений) автоматизацию изготовления всей необходимой документации (чертежей, спецификаций, нормативов и т. п.).

Важной особенностью современного этапа в развитии САПР являются все большее слияние и взаимопроникновение собственно конструкторского и технологического этапов про-

ектирования. При ориентации проектов на автоматизированное производство (в первую очередь на программно-управляемое оборудование) проектно-техническая документация традиционного (ориентированного на человека) вида в значительной мере теряет свое значение. На первое место выходит технологическая документация на машинных носителях, которую можно непосредственно использовать для управления оборудованием с ЧПУ. Для целей контроля (осуществляемого человеком) будет готовиться только упрощенная документация крайне ограниченного объема (разумеется, в том случае, если человек не пожелает иметь большего). Тем самым в полной мере (в разумном для человека смысле этого слова) будет осуществлен переход к безбумажной технологии автоматизированного проектирования.

Чтобы проще и лучше представить себе современное состояние работ по САПР, мы начнем свое изложение идеализированным описанием полностью автоматизированного крупного КБ или проектного института относительно близкого будущего. Прежде всего подчеркнем, что полная автоматизация понимается здесь не в смысле полного исключения человека из процесса проектирования, а в смысле освобождения его от всех рутинных работ. Основой описываемого идеализированного САПР является многомашинный вычислительный комплекс, состоящий из крупных (общеинститутских) ЭВМ, мини-ЭВМ в отделах и отдельных группах и, наконец, из микро-ЭВМ, вмонтированных в *автоматизированные рабочие места* (АРМ) каждого сотрудника.

Все мини-ЭВМ, как правило, снабжаются автоматическими линиями по производству технической документации (чертежные автоматы, АЦПУ, копировальные и брошюровочные машины, выводные устройства, выпускающие носители с соответствующими программами для оборудования с ЧПУ). Еще более мощными линиями такого рода снабжается общеинститутский ВЦ. В ряде случаев подобный *центр подготовки технической документации* (ЦПТД) может исключить необходимость иметь подобные линии при отдельных мини-ЭВМ.

Вычислительный комплекс через Государственную сеть вычислительных центров должен иметь автоматический выход на централизованные специализированные банки данных по материалам, стандартным конструктивным элементам, комплектующим деталям, оборудованию и т. п. В случае отсутствия сети каждый проектный институт (или КБ) должен заводить упрощенные (ориентированные на свои нужды) банки такого рода. Нет нужды доказывать, что централизованные (пополняемые разработчиками всей страны и всеми доступными зарубежными материалами) банки данных могут обеспечить гораздо более полный и качественный сервис для проектантов, чем любые упрощенные доморощенные варианты.

Под автоматическим выходом на централизованные банки данных здесь понимается возможность для каждого сотрудника, не сходя со своего рабочего места, оперативно получить все необходимые ему сведения (включая чертежи) об интересующем его объекте. Поскольку данные получаются через институтский вычислительный комплекс, то при решении использовать запрошенный стандартный элемент в своем разделе проекта сотрудник не должен вводить никаких данных (кроме имени требуемого элемента) в ЭВМ. Нужно заметить, что борьба с ручным вводом информации в ЭВМ является одной из главных отличительных черт комплексной автоматизации вообще и САПР в особенности.

Вычислительный комплекс САПР должен быть обеспечен машинным автоматизированным архивом проектов, выполненных в данном институте или КБ. Из этого архива в установленном порядке (с минимальной задержкой после окончания разработок) должны пополняться соответствующие централизованные банки. Кроме того, сам институт (или КБ) может вести один или несколько (специализированных) централизованных банков данных (в соответствии с произведенным распределением). Ко всем этим данным в принципе также должна быть обеспечена возможность оперативного доступа непосредственно с рабочих мест.

Чтобы избежать возможных недоразумений, заметим сразу, что возможность оперативного доступа к данным ограниченного пользования реализуется только для лиц, имеющих на то соответствующее право. О средствах защиты информации от несанкционированного доступа уже говорилось выше (в гл. VI).

Помимо уже перечисленных банков данных, носящих в той или иной мере архивный характер, в системе должен иметься главный оперативный банк данных, а именно *банк данных текущих проектов* (выполняемых институтом или КБ в настоящее время). Этот банк включает в себя прежде всего графическую информацию от самой подробной до обобщенной (от чертежей отдельных деталей и конструктивных элементов до чертежей общего вида). К графической информации на всех уровнях «привязывается» вся другая информация (размеры, виды материалов, точность обработки и др.).

Банк должен быть снабжен программами, осуществляющими *агрегацию* и *дезагрегацию* данных. Смысл агрегации состоит в уменьшении степени подробности описания отдельных элементов конструкции при включении их в описания более высоких уровней (описывающих целые блоки или всю конструкцию в целом). Дезагрегация — обратный процесс, позволяющий выбирать из описаний высших уровней те или иные их части, восстанавливая детализированные описания этих частей. При этом возникает

естественная иерархически строящаяся система описания конструкции: на верхних уровнях — описания укрупненной документации (технические задания, чертежи общего вида и т. п.), на нижнем уровне — самая подробная документация (включая технологическую), описывающая первичные элементы, из которых строится данная конструкция. Документация более высоких уровней, описывающая сборки, должна включать в себя описания технологии сборки и соответствующих (питающих сборку) материальных потоков.

Все требования, высказанные до сих пор, удовлетворяются в иерархических базах данных (см. гл. VI). Однако для САПР иерархическая структура базы данных обычно оказывается недостаточной. Необходимо учитывать «горизонтальные» связи, возникающие между взаимодействующими друг с другом элементами конструкции,— связи, находящиеся на одном и том же уровне иерархии (пара сцепляющихся зубчатых колес, мотор и приводимый им в движение агрегат и т. п.). Подобные связи лучше и естественнее всего учитываются в сетевых структурах баз данных. С определенной степенью условности, но тоже достаточно просто, они могут реализоваться и в реляционных структурах с помощью введения специальных отношений соседства.

Если в «данных пересечениях» горизонтальных связей указаны границы соприкосновения соседствующих конструктивных элементов, то с помощью специального программного обеспечения нетрудно автоматически передавать вдоль таких связей данные об изменениях этих границ. Тем самым через базу данных изменения, вносимые в процессе конструирования на одних рабочих местах, будут автоматически сигнализировать о необходимости изменений на всех рабочих местах, связанных с ними логикой проекта. Путем приведения в действие процедур агрегирования и дезагрегирования сведения о произведенных изменениях могут автоматически передаваться на другие рабочие места по линии не только горизонтальных, но и вертикальных (иерархических) связей.

Благодаря наличию описанных автоматических связей рабочих мест через базу данных работа над проектом может быть организована в диалоговом режиме, где партнером системы в диалоге будет коллектив ответственных конструкторов примерно с тем же самым разделением функций по конструктивным элементам и уровням иерархии проекта, какое имеет место и в обычном, не автоматизированном проектировании. Различие состоит прежде всего в том, что каждый конструктор работает с системой в одиночку, без многочисленных помощников и вспомогательного персонала.

Получая исходные данные для работы через систему с вышестоящего уровня (самый верхний уровень — от ТЗ), ответствен-

ный конструктор работает в диалоговом режиме с системой через свой АРМ, в состав которого должен входить прежде всего графический дисплей со световым пером или произвольно перемещаемым курсором и с соответствующей клавиатурой. В случае необходимости одновременной работы с тремя проекциями на пульте могут быть смонтированы три графических дисплея. С помощью светового пера и клавиатуры конструктор может вводить в систему с экрана дисплея различного рода эскизные наброски конструкторских решений. При этом стандартные конструктивные элементы вместе со всеми необходимыми описаниями могут вызываться конструктором из соответствующих баз данных и включаться в состав проектируемого им блока.

Используя программы для геометрических преобразований, разнообразные расчетные программы, а также программы для автоматической компоновки требуемых конструктивных элементов в заданном объеме или на заданной площади и внося в случае необходимости те или иные поправки в сделанные эскизные наброски, конструктор оформляет в памяти системы порученный ему участок в цельный проект со всеми необходимыми описаниями. При изменениях других участков проекта он в соответствии с указанием своего руководителя вносит дополнительные поправки. Наконец, когда проект будет полностью завершен, по указанию руководителя проекта приводятся в действие автоматические линии по производству всей необходимой технической (и технологической) документации.

Важной особенностью описанной технологии автоматизированного проектирования является минимизация работы по вводу данных в систему. Необходимые данные для расчетов берутся из системы указанием их имени или просто идентификацией рассчитываемого объекта путем прикосновения светового пера к его изображению на дисплее. С помощью светового пера конструктор может двигать на экране отдельные элементы конструкции, добиваясь улучшения компоновки. Расчетные программы, равно как и программы геометрических преобразований, также вызываются по их именам, а наиболее простые из них — нажимом соответствующих кнопок на клавиатуре дисплея. Благодаря этому становится выгодно автоматизировать не только сложные, но и простые расчеты, которые при отсутствии комплексной автоматизации было невыгодно выполнять на ЭВМ, потому что экономия, полученная от автоматизации вычислений, не компенсировала потерю времени на ввод исходных данных.

Следует иметь в виду, что при проектировании сложных объектов (например, сверхзвуковых самолетов) нередки случаи, когда точные методы расчета еще до конца не разработаны. Тогда прибегают к *физическому моделированию* (например, к испытаниям моделей самолетов в аэродинамических трубах). При

комплексной автоматизации процесса проектирования все этапы физического моделирования (изготовление модели, планирование эксперимента, его реальное проведение, съем получаемых данных и их ввод в систему) также автоматизируются. На этапе изготовления модели в процесс включаются соответствующие автоматические линии с ЧПУ, на этапе самих испытаний — системы АСНИ. В случае подобного объединения автоматизированных систем различного назначения получаемая в результате система комплексной автоматизации называется обычно *интегрированной автоматизированной системой.*

В полной мере для большинства видов сложных проектов описанная технология еще не внедрена. Однако отдельные ее части получают все большее распространение. Для простых конструкций (фермы мостов, подпорные стенки, электросчетчики, простые автоматические регуляторы и др.) уже сегодня существуют полностью автоматические системы проектирования, которые выдают полный набор технической документации по техническому заданию. Достаточно широкое распространение получили системы автоматического раскроя (одномерного или двумерного), системы автоматического проектирования (размещения элементов и трассировки) печатных плат и больших интегральных схем, автоматического проектирования трубопроводов и др.

10.6.1. Проблемно-ориентированные комплексы. Из приведенного описания системы комплексной автоматизации проектно-конструкторских работ ясно, что для ее реализации необходима специфическая структура технического комплекса (включающего специализированные АРМ). Кроме того, значительная часть программного обеспечения практически не зависит от вида проектов, для которых оно предназначено,— прежде всего, программы работы с графической информацией и расширение операционной системы, обеспечивающие взаимодействие АРМ через специализированные базы данных. То же самое можно сказать о традиционных расчетных программах в рамках общеинженерных курсов, изучаемых в технических вузах.

Создавая системы генерации описанного «общепроектного» программного обеспечения применительно к различным возможным конфигурациям технических комплексов, мы фактически создаем своеобразные «заготовки» для автоматизации любых проектно-конструкторских работ. По предложению автора, подобные заготовки получили название *проблемно-ориентированных комплексов* (ПОК). Они могут быть ориентированы не только на автоматизацию проектирования, но и на другие системные применения ЭВМ (например, на автоматизацию испытаний). Получив такие комплексы, можно гораздо быстрее создать соответствующие автоматизированные системы, дополняя ПОК специализированными программами, прежде всего расчетного характера.

С этой целью в состав ПОК включаются средства той или иной технологии автоматизированного программирования (см. гл. V).

10.6.2. Автоматизация исследовательского проектирования. В последние годы все большее внимание уделяется автоматизации исследовательского этапа проектирования, т. е. того этапа, на котором оформляются основные идеи будущего проекта. При этом ЭВМ с соответствующим программным обеспечением активно участвует в процессе, автоматизируя значительную часть изобретательской деятельности.

При автоматизации исследовательского проектирования используется небольшое число базовых идей. Первая из них — создание *автоматизированного банка* различного рода явлений и процессов, известных современной науке. При этом стандартное программное обеспечение банка данных дополняется программами, позволяющими автоматизировать процесс комбинирования таких явлений в цепочке, приводящей к решению поставленной задачи. Например, если требуется сконструировать прибор, измеряющий поворотом стрелки указателя величину освещенности, ЭВМ может выбрать из банка и связать в цепь три хорошо известных физических явления: преобразование света в электрический ток, тока — в магнитное поле, а магнитного поля — в отклонение стрелки.

В хорошо организованных системах, использующих эту идею, в банк включаются известные технические реализации рассматриваемых явлений, а также необходимые количественные данные, позволяющие автоматически рассчитать «стыки» объединяемых в цепочку явлений с целью определения практической реализуемости выбираемых комбинаций и их ориентировочных параметров (например, точности). Наряду с известными техническими реализациями исходных явлений в банк могут быть включены другие их известные реализации, например, в живых организмах. Практика показывает, что при достаточно большой величине исходного банка явлений система способна автоматически находить новые технические решения, находящиеся на уровне изобретений.

Вторая базовая идея — построение так называемых *алгебр конструкций* и автоматизация с помощью специального программного обеспечения различного рода формальных преобразований в таких алгебрах. Конструкция представляется при этом в виде формулы (или системы формул) в некоторой алгебре. Возможные преобразования конструкции задаются с помощью так называемых *определяющих соотношений*, аналогичных соотношениям $a+b=b+a$, $a(b+c)=a\cdot b+a\cdot c$ и др. из обычной алгебры. При этом преобразовании, меняя конструкцию, не меняют ее функционального назначения, точно определяемого при задании соответствующей алгебры.

Например, любая комбинационная схема, как было показано в § 1.6, может быть задана системой формул булевой алгебры. Преобразования этих формул с помощью соотношений булевой алгебры меняют конкретные реализации комбинационных схем, не меняя задаваемых ими отображений. Другой пример: форма поперечного сечения (профиля) подпорной стенки (рис. 10.1) может быть задана системой уравнений $f_i(x, y) = 0$ $(i = 1, 2, \ldots, k)$ его границ. Подвергая левые части этих уравнений различного рода преобразованиям, не меняющим высоты H и усилия P, выдерживаемого стенкой, мы реализуем алгебру профилей подпорных стенок. Имея алгебру, можно ставить те или иные задачи нахождения оптимальных профилей, например профилей минимальной площади. Таким способом были найдены новые формы профилей, составившие предмет изобретений.

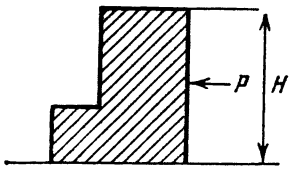

Рис. 10.1

Автором была решена задача построения алгебр алгоритмов и программ, позволившая автоматизировать глубокие преобразования схем преобразователей дискретной информации вместе с их математическим обеспечением.

Смысл использования алгебр конструкций на этапе исследовательского проектирования состоит в том, что, найдя тем или иным способом какую-нибудь реализацию полученного задания (обычно достаточно плохую), конструктор может последовательно улучшать найденное решение. При этом важно, что в процессе такого улучшения могут меняться не только отдельные количественные параметры, характеризующие конструкцию, но и появляться принципиально новые качественные решения. Например, с помощью алгебры алгоритмов и программ автоматически, с помощью ЭВМ, находится идея позиционной системы счисления, на изобретение которой в истории математики были затрачены многие сотни лет.

Объединение описанных базовых идей подводит серьезную основу для автоматизации изобретательской работы (в процессе диалога с человеком) в различных сферах проектно-конструкторской деятельности.

10.6.3. Автоматизация системы первичной подготовки технологических нормативов. Для автоматизации организационного управления важное значение имеет достоверная информация о нормативах трудовых и материальных затрат, необходимых для реализации проектов новых сооружений и конструкций. Так, если речь идет об организации производства новой машины, необходимы нормы расхода различных материалов и комплектующих изделий в расчете на одну машину, а также затраты вре-

мени работы оборудования и рабочих для ее производства в запроектированной технологии.

Подготовкой нормативной информации должен заканчиваться процесс изготовления технологической документации каждого проекта. Не вдаваясь в специфику расчетов окончательных нормативов, остановимся подробнее на методике *предварительной оценки* нормативов на ранних стадиях проектирования, т. е. в эскизном проекте или даже в аванпроекте. Такая оценка важна, во-первых, для решения вопроса о том, в каком направлении следует двигаться дальше, т. е. какие варианты развивать на следующих, более длительных и дорогостоящих этапах проектирования — в технологическом и рабочем проектах. Во-вторых, такая информация крайне важна для долгосрочного планирования производства (о чем пойдет речь в следующей главе).

В то же время дать даже грубую оценку нормативов (с точностью до 5—10%) на ранних стадиях проектирования совсем непросто. Более того, обычные методы получения подобной информации от проектантов зачастую просто не срабатывают. Успех достигается, как правило, лишь в том случае, когда в описанную выше схему комплексной автоматизации проектирования вводится специальный, направляемый ЭВМ диалог с коллективом проектантов. Прежде всего, ЭВМ задает вопросы главному конструктору — руководителю проекта — относительно того, из каких блоков и узлов будет состоять проектируемый объект и кто персонально будет отвечать за их проектирование. Далее, включая по мере необходимости в диалог ответственных исполнителей соответствующих разделов проекта, ЭВМ путем серии целенаправленных вопросов уточняет, какие *прототипы* из автоматизированного архива простых проектов или соответствующих централизованных банков данных ближе всего отвечают задачам, поставленным исполнителям. Затем последовательно извлекая из банка прототипов различные технологические нормативы, ЭВМ просит исполнителей или главного конструктора оценить в процентах отличие от них соответствующих нормативов нового проекта. При затруднениях в такой оценке (в частности, в случае отсутствия подходящего прототипа) ЭВМ предлагает следующий, более детальный уровень разузливания рассматриваемых частей проекта с указанием прототипов и относительной оценки нормативов для соответствующих более мелких узлов.

Полученные оценки автоматически переводятся из относительных показателей в абсолютные и нормативы по одинаковым позициям на разные узлы суммируются (например, по расходу электроэнергии). В результате получается полная оценка нормативов по всему проекту в целом.

10.6.4. Автоматизация процесса агрегации нормативов. При подготовке данных для систем планирования на различных уров-

нях (от предприятия до Госплана СССР) возникает необходимость *агрегации* (укрупнения) технологических нормативов. Существуют два вида агрегации. В первом из них в рамках одного и того же изделия (проекта) происходит агрегация по различным группам материалов, оборудования или рабочих специальностей. Например, нормы расхода на одно изделие стали различных марок агрегируется в норму суммарного расхода стали всех марок на это изделие.

При подобной агрегации выполняются операции двух типов. Во-первых, это пересчет агрегируемых нормативов на одни и те же единицы измерения. Скажем, в случае расхода стали различных марок — переход от квадратных метров к килограммам или тоннам. Уточнение выбора общей единицы измерения может производится в режиме диалога. В случае отсутствия общего натурального показания всегда возможен выбор общей единицы измерения за счет перехода к стоимостному показателю (рублю). Программы пересчета должны быстро и просто (обычно в режиме диалога) адаптироваться к любым возможным изменениям, в первую очередь — к изменениям цен. Второй тип операции — это прямое сложение агрегируемых нормативов, приведенных к общей единице измерения.

Агрегация второго вида предполагает объединение нормативов по группе изделий (проектов). Например, речь может идти об определении средней нормы v расхода стали на один легковой автомобиль, при условии, что выпускаются автомобили n различных марок с нормами расхода стали $v_1, v_2, \ldots, v_n$ соответственно. Для определения подобной усредненной нормы необходимо задать (обычно в диалоговом режиме) предполагаемые объемы x_i или относительные доли α_i выпуска автомобилей различных марок за период, на который рассчитывается усредненный норматив. При этом

$$\alpha_i = \frac{x_i}{x_1 + x_2 + \ldots + x_n}, \quad i = 1, 2, \ldots, n,$$

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n.$$

Управление агрегацией ведется с помощью задания соответствующих автоматизированных *классификаторов*, в случае необходимости изменяемых в диалоговом режиме. В системах автоматизированного планирования подобная агрегация ведется обычно на сетях ЭВМ по мере перехода от очередного уровня планирования к следующему по цепочке: предприятие, объединение, отрасль, Госплан. Возможно включение в эту цепочку цеха, а также территориальных уровней агрегации.

10.6.5. Автоматизированные системы контроля правильности нормативов. Поскольку первичная информация о нормативах ис-

ходит от людей, необходимы соответствующие *системы контроля* за ее правильностью. Ввиду невозможности полной автоматизации подобного контроля необходим контроль людьми требуемой квалификации. Чтобы не занимать в системе контроля слишком много людей, его необходимо максимально автоматизировать. Но даже при таком условии хорошо организованная система контроля предполагает, как правило, лишь выборочную проверку. Чтобы быть при этом достаточно эффективной и надежной, система контроля должна подкрепляться соответствующей *системой стимулирования*.

Опишем одну из возможных систем автоматизированного контроля, удовлетворяющих выдвинутым требованиям. Первая ее особенность — наличие строгой *персональной* ответственности за информацию (в данном случае — за нормативы). Вторая особенность — наличие *автоматизированного архива*, организованного так, чтобы по требованию системы контроля могла быть оперативно подобрана информация, поданная тем или иным лицом (или группой лиц) в течение любого периода на несколько лет назад. В случае необходимости проверки *мотивировки выбора* проверяемым исполнителем той или иной информации системе контроля открывается доступ к использовавшейся им системе автоматизированного проектирования (включая банки прототипов). Кроме того, если к моменту проверки проект уже реализован в производстве, системе контроля должен быть открыт доступ к автоматизированной системе управления производством с целью получения оттуда объективной информации о реально достигнутых нормативах.

Производя проверку правильности дававшейся информации, в соответствии со строго определенной процедурой группа контроля делает заключение о величине зарплаты всех проверенных лиц. В случае отсутствия обнаруженных нарушений добросовестности в выданной информации действует процедура медленного (из месяца в месяц) повышения зарплаты с установленного минимального уровня z до некоторого максимального (существенно большего, чем z) уровня Z*). В случае обнаружения нарушений достигнутая к моменту их обнаружения надбавка над минимальным уровнем зарплаты может быть полностью или частично снята и для ее восстановления до прежнего уровня потребуется снова проработать достаточно длительное время.

Принципиально важно при этом, что проверке подвергается не только текущая, но и прошлая информация. В противном случае при выборочности контроля могли бы возникать надежды как-то «проскочить» проверку, что, как показывает практика,

*) Такое повышение производится и тогда, когда никаких проверок не производилось, а значит, отсутствуют о б н а р у ж е н н ы е нарушения.

резко снижает эффективность контроля. Попытки уйти от ответственности в прошлом путем перехода на новую работу должны быть пресечены специальной общегосударственой процедурой установления начального уровня зарплаты на новом месте работы. Если меняющий работу хочет на новом месте сохранить заработанный на старом месте стаж добросовестной работы и соответственно повышенный уровень зарплаты, за ним должна быть сохранена ответственность за возможную недобросовестность, допущенную на прежнем месте работы. В противном случае он должен начинать на новом месте «с нуля», т. е. с минимального уровня зарплаты.

Наличие описанного «груза ответственности за прошлое» позволяет получить достаточно эффективное воздействие контроля на правильность всей нормативной информации при охвате выборочности проверками лишь 10—15% ее общего объема. Поскольку выработка нормативов составляет лишь относительно небольшую часть работы проектантов, а сама проверка все же несколько проще, чем первоначальная разработка, можно построить весьма действенную систему контроля, задействовав в ней относительно небольшое число людей (порядка нескольких процентов от общего числа людей, занятых проектно-конструкторской деятельностью).

Сделанную оценку не увеличивает сколько-нибудь существенно даже необходимость иерархического построения системы контроля и определенного процента дублирования контроля с целью контроля самих контролеров. Иерархическое построение означает наличие контрольных групп на разных уровнях управления (от уровня отдельного предприятия или организации до общегосударственного уровня). При этом на каждом очередном уровне иерархии контрольный аппарат должен быть существенно меньше (в 5—6 раз и более), чем на предыдущем, более низком уровне. Чтобы большинство нарушений вскрывалось контрольными органами нижних уровней иерархии, должна быть введена система зависимости уровня зарплаты от стажа добросовестной работы не только у проектантов, но и у контрольных органов, а также у руководителей организации, где эти проектанты работают. При вскрытии нарушений органами контроля «своего» уровня, эти контрольные органы (а также руководители соответствующего уровня) не только не теряют стаж добросовестной работы, но даже могут получить разовую прибавку к нему (или специальную премию). Если же нарушения вскрываются контрольными органами вышестоящего уровня, то процедура снижения достигнутого уровня зарплат распространяется в той или иной степени на контрольные органы и руководство проверяемого уровня (особенно если нарушения вскрыты в информации, уже проверенной контролем проверяемого уровня).

Мы остановились подробно на системах контроля правильности нормативной информации в силу ее определяющего значения для любых возможных систем планирования и управления производством, для эффективности всего народного хозяйства в целом. Нетрудно заметить, однако, что принципы, заложенные в описанную систему автоматизированного контроля, могут быть применены для организации эффективного контроля за добросовестностью работы не только при выработке нормативов, но и в любых других сферах человеческой деятельности.

10.7. Автоматизация в кредитно-финансовой системе

Автоматизация работы банков была одной из первых в мировой практике областей системного использования ЭВМ. Смысл ее состоит прежде всего в переводе счетов клиентов с бумажных носителей во внешнюю память ЭВМ. Сотрудники банка, работая на специальных автоматизированных рабочих местах (банковских АРМ), осуществляют ввод требований клиентов на те или иные операции с их счетами — прежде всего, переводы со счета на счет, а также вкладывание на счет или получение с него наличных денег. Поскольку сами операции со счетами выполняются автоматически, при такой автоматизации существенно увеличивается пропускная способность банков, а значит, и увеличивается производительность труда.

Переводы со счета на счет в масштабах одного автоматизированного банка выполняются тривиальным способом в пределах одной ЭВМ или одного многомашинного комплекса. Организация переводов между счетами, находящимися в разных банках, может осуществляться либо автоматически по каналу связи (если банковские ВЦ объединены в сеть), либо путем почтового обмена между банками информацией на машинных носителях. Во втором случае применяется соответствующая группировка операций с разными счетами на одном носителе с целью его более экономного использования.

Широкая автоматизация банковских систем в капиталистических странах позволила резко расширить круг их клиентов, обеспечить возможность распространения безналичных расчетов практически на все население. Первой формой безналичных расчетов явились обычные *чековые операции*. В этой форме каждый клиент банка получает от него чековую книжку. Расплачиваясь, владелец книжки вырывает из нее чек, пишет на нем уплачиваемую им сумму, расписывается и оставляет чек вместо денег. Лицо, получившее чек, проставляет на нем номер счета, на который должна быть переведена уплаченная сумма, и направляет

оформленный чек в банк, где находится его счет, а там в свою очередь, пересылая чек в выдавший его банк, обеспечивает соответствующий перевод со счета клиента на счет получателя.

По мере насыщения системы торговли, общественного питания и др. специальными кассовыми аппаратами создается возможность перехода к форме бесчековых расчетов. В этом случае используется специальная чековая книжка, обычно с фотографией и с вклеенным в нее кусочком магнитной ленты. В банке владельца книжки специальный автомат производит считывание и запись информации на эту ленту. На нее заносится, в частности, код банка и номер счета клиента, остаток вклада, а также, возможно, специальный (обычно двузначный) *контрольный код*, запоминаемый клиентом с целью исключить возможность использования без специальной аппаратуры его чековой книжки в случае ее попадания в чужие руки. Этот же автомат может по желанию клиента распечатать в книжке или на отдельном листе бумаги все операции с момента последней распечатки (снятия со счета и поступления на счет).

Чтобы расплатиться с помощью книжки, клиент вставляет ее в специальный кассовый аппарат и вводит в него свой контрольный код, после чего книжка становится доступной для операций на этом аппарате. В число таких операций входит съем необходимой суммы с величины остатка вклада, указанной на вклеенной магнитной ленте, и формирование платежного поручения банку владельца книжки для фактического перевода снятой суммы на счет получателя. Оформление платежного поручения может производиться либо путем изготовления соответствующего документа для последующей пересылки в банк, либо путем прямой (автоматической) связи в ВЦ банка владельца книжки. Подобные кассовые аппараты обычно связаны с ЭВМ, ведущей бухгалтерские расчеты у получателя денег. Это позволяет формировать не индивидуальные, а групповые платежные поручения на машинных носителях, подобно тому, как это делается при обмене информацией между автоматизированными банками.

Если клиент ввел в аппарат неправильный контрольный код, ему обычно предоставляется вторая попытка. После повторной неудачи формируется сигнал тревоги, извещающий, что книжкой пытается воспользоваться другой человек. В случае автоматической связи кассового аппарата с ВЦ банка контрольный код не записывается на ленту, а хранится непосредственно в банковском счете клиента; при этом система защиты счета при возможной утере книжки становится еще более совершенной.

Проводятся эксперименты по автоматической идентификации клиента по отпечаткам пальцев или по индивидуальным особен-

ностям его голоса. Однако, в силу большой сложности и дороговизны, подобные системы идентификации не получили сколько-нибудь заметного распространения в кредитно-финансовой деятельности, в то время как для ряда специальных целей подобные системы находят все более широкое применение (прежде всего в криминалистике).

При небольших суммах, записанных на встроенной ленте, можно обходиться вообще без контрольных кодов. На этом основан, например, многоразовый билет, используемый для оплаты поездок в общественном транспорте. При покупке билета на встроенный в него кусочек магнитной ленты записывается размер уплаченной суммы. В общественном транспорте устанавливаются унифицированные кассовые аппараты, оперирующие с билетом как с чековой книжкой (без контрольного кода). Все операции последовательного съема с уплаченной при покупке билета суммы распечатываются кассовым аппаратом тут же на билете. Одновременно выпечатываются признаки, по которым можно определить, где и когда произведена уплата (чаще всего время и номер аппарата).

Многие банки используют специальные автоматы, с помощью которых клиентам выдаются небольшие суммы наличных денег (по постоянным чековым книжкам). Такие автоматы устанавливаются обычно у входа в банк, так что ими можно воспользоваться и в то время, когда банк закрыт, в частности, поздно вечером или даже ночью.

По образцу банков автоматизируются и обычные сберкассы. В случае, когда по той или иной причине используются сберкнижки без встроенных магнитных лент, для идентификации личности клиента могут служить обычные удостоверения личности (чаще всего — паспорт). В этом случае на пульте оператора сберкассы производится автоматическая распечатка выполненных операций прямо на самой сберкнижке. Автоматизированные сберкассы сильно облегчают выполнение постоянных поручений клиентов по безналичным расчетам за квартиру, коммунальные услуги и т. п. При автоматизации начисления зарплаты и других видов выплат клиентам организации, осуществляющие эти выплаты, могут принимать постоянные или временные поручения о переводах получаемых сумм на тот или иной счет в выбранной клиентом сберкассе. Получение наличных денег по счету при комплексной автоматизации системы сберкасс можно производить в любой сберкассе, включенной в общую систему.

10.7.1. Социальное значение системы безналичных расчетов. Развитие системы безналичных расчетов создает большие удобства для пользователей и сильно сокращает количество находящихся в обороте бумажных денег. Отметим также, что в социа-

листических странах подобные системы могут если не полностью закрыть дорогу, то, во всяком случае, сильно ограничить такие явления, как воровство, взяточничество, спекуляцию. С этой целью необходимо создать наряду с обычной системой вкладов в сберкассах специальную систему индивидуальных безналичных вкладов, которые в законодательном порядке было бы запрещено пополнять с помощью наличных денег. Хотя брать с них наличные деньги в принципе не запрещалось бы, подобные вклады предназначались бы в первую очередь для безналичной оплаты покупок и услуг, предоставляемых государственными учреждениями (магазинами, ресторанами, транспортом и др.). Пополнение таких вкладов также должно быть возможно лишь за счет перечислений от государственных учреждений всех или определенной части законных выплат (зарплата, премии, гонорары, пенсии и т. п.).

Предоставляя преимущества в приобретении товаров и услуг за счет перечислений с подобных вкладов, можно соответствующим образом повышать реальную ценность законных доходов по сравнению с любыми видами незаконных доходов. Преимущества, о которых здесь идет речь, должны прежде всего обеспечить (в законодательном порядке) первоочередное удовлетворение спроса, покрываемого за счет безналичных вкладов. Для товаров и услуг, составляющих предмет вожделений любителей незаконных доходов (золото, автомобили, банкеты в модных ресторанах и т. п.), можно было бы даже пойти на полное запрещение отпуска их за наличные деньги. Другой мерой повышения реальной ценности законных доходов было бы установление меньших цен на товары и услуги в случае оплачивания их по безналичному расчету.

Нетрудно понять, что любое заметное повышение реальной ценности законных доходов явилось бы одним из самых мощных стимулов повышения общественной производительности труда. Разумеется, при этом необходимо обеспечить правильную оценку и оплату труда в соответствии с его количеством и качеством. Важнейшее значение здесь имело бы широкое использование систем контроля добросовестности, аналогичных описанной выше (п. 10.6.5) системе контроля добросовестности подготовки нормативной информации. Легко видеть, что такие системы пригодны для контроля не только чисто информационной, но и любых видов производственной деятельности.

Развитие системы безналичных персонифицированных расчетов, как это было показано автором еще в 1963 г., открывает реальный путь к постепенному переходу к полностью бесплатному распределению товаров и услуг.

10.8. Автоматизация учета

Одной из важнейших информационных технологий, в которой автоматизация на основе ЭВМ приобрела большой размах, является различного рода учетная работа. Учет тесно связан с контролем, являясь основой его информационной базы. Поэтому в ряде случаев учет трудно отделить от контроля. Так, описанные в § 9.6 автоматизированные системы контроля оборудования, помимо собственно контрольных (сигнализирующих) функций, обычно выполняют также учет времени работы оборудования и его простоев по тем или иным причинам. Задачей учета является периодическая фиксация, классификация и запоминание тех или иных факторов, значений различных параметров, характеризующих протекающие процессы, и т. п. При этом в автоматизированных системах первичного учета стремятся к максимальному сокращению труда, затрачиваемого на ввод исходной учетной информации в память ЭВМ. Обработка информации на этой стадии сводится обычно к выполнению простейших арифметических действий. Системы вторичной обработки учетных данных могут включать в себя процедуры группировки данных по тем или иным признакам, нахождение законов изменения различных параметров, корреляционных функций, выдачу различного рода таблиц, кривых, гистограмм и т. п.

Учет делится на материальный и финансовый. В первом случае используются натуральные единицы измерений (штуки, квадратные метры, тонны и т. п.), во втором — денежные (рубли и копейки). В автоматизированных системах учета, как правило, должна предусматриваться возможность быстрого пересчета учитываемых величин из одних единиц измерения в другие. С этой целью в память ЭВМ закладываются специальные таблицы, которые в случае необходимости легко заменяются без перепрограммирования самих процедур пересчета.

Для ведения автоматизированного учета создаются *классификаторы* учитываемых объектов (изделий, материалов, основных фондов и т. п.). В таких классификаторах каждому объекту приписывается с целью экономии памяти по возможности более короткий буквенно-цифровой код. При этом кодам с одинаковыми начальными (старшими) позициями соответствуют группы родственных объектов. При такой организации классификаторов сильно облегчается организация автоматизированного поиска. Если в каком-то звене экономики (цехе, предприятии, отрасли) употребляется лишь небольшая часть классифицированных объектов, то для него возможно построение укороченных *локальных* классификаторов. Для возможности выхода на другие звенья в составе программного обеспечения соответствующих автоматизированных систем предусматриваются автоматические *перекодировщики* из локальных классификаторов в общесоюзные.

Поскольку в условиях динамично развивающегося общества номенклатура учитываемых объектов подвержена постоянным изменениям, необходима *система автоматизированного ведения классификаторов*. Такая система подготавливает и распределяет между пользователями (на машинных носителях или по каналам связи) все изменения в классификаторах. При этом, в случае наличия перекодировщиков, они также должны автоматически подстраиваться под все происходящие изменения. Наиболее разработанными и унифицированными являются, как известно, *системы бухгалтерского учета*.

Остановимся на особенностях, которые привносит автоматизация в различные конкретные виды учета.

10.8.1. Складской учет. Операции учета на складах сводятся к фиксации *прихода* и *расхода* изделий, материалов или полуфабрикатов, хранящихся на складе, и к вычислению *остатка* по всем позициям номенклатуры в заданный момент времени. При большой *номенклатуре* вычисление остатка по каждой позиции этой номенклатуры после каждой приходно-расходной операции возможно лишь при условии автоматизации учета. Для такой автоматизации, как правило, не требуется ЭВМ с большим быстродействием. Требования же к внешней памяти определяются прежде всего размером складской номенклатуры. При современном состоянии дел с внешней памятью наиболее удобно *текущие остатки* (обновляемые после каждой приходно-расходной операции) по всем позициям номенклатуры хранить на магнитных дисках, а *архив* приходно-расходных документов (ведомостей) вести на магнитных лентах. По мере заполнения архивные ленты могут перерабатываться в соответствии с принятыми способами ведения безбумажного архива и помещаться на хранение в лентотеку.

Оформление приходно-расходных операций может производиться разными способами в зависимости от степени комплексности автоматизации не только самого складского учета, но и всех взаимодействующих с ним звеньев. При локальной автоматизации, когда на входе и выходе сохраняются бумажные документы, кладовщик оформляет на каждую операцию документ на пишущей машинке, снабженной перфоприставкой или непосредственно связанной с ЭВМ, так что ввод подготовленного документа в ЭВМ не требует дополнительных ручных операций по подготовке данных.

При комплексной автоматизации оформление приходно-расходных операций в принципе может осуществляться без всякой работы на клавиатуре. Простейший способ для этого — сопровождение материальных ценностей, поступающих на склад и выдаваемых со склада, *машинно-читаемыми сопроводительными документами* (накладными, заборными картами, специальными мет-

ками на упаковке и т. п.). С помощью специализированного вводного устройства — *складского регистратора* — осуществляется автоматическое считывание этих документов и оформление соответствующей приходно-расходной операции. При этом обычно производится автоматическое нанесение на документ отметок о проведенной операции, а возможно, и подготовка новых (машиночитаемых) документов.

При наличии непосредственной связи между ЭВМ склада и ЭВМ пользователей складом объем информации на подобных сопровождающих документах может быть сведен к минимуму, необходимому для идентификации абонента. Вся остальная информация может быть передана путем прямого межмашинного обмена.

Заметим, что автоматизация складского учета поднимается на более высокий уровень в *комплексно-автоматизированных складах*, где автоматизируются не только учетные, но и чисто технологические операции. В их число входит поиск нужного объекта или места на складе, погрузочно-разгрузочные и транспортные операции (в пределах склада). Автоматизация технологических процессов на складах выполняется с помощью специализированного подъемно-транспортного оборудования с ЧПУ, управляемого от центральной ЭВМ. В памяти ЭВМ содержится схема размещения продукции на складе, оформленная в виде специализированной базы данных с возможностью быстрого поиска места размещения любого требуемого объекта по его имени или инвентарному номеру.

10.8.2. Учет выпускаемой продукции. При наличии на производственном участке выходного склада учет продукции, выпускаемой этим участком, сводится к соответствующему складскому учету. При отсутствии таких складов для автоматизации учета произведенной продукции употребляются специальные автоматические и ручные датчики. На нефтепромыслах такие датчики фиксируют количество добытой нефти, на сборочных конвейерах — количество шагов конвейера, на отдельных станках и станочных линиях — количество изготовленных деталей и т. п.

С целью устранения случайных или преднамеренных искажений информации в систему учета вводятся различного рода процедуры контроля (особенно в случае ручных датчиков). Они основываются прежде всего на постоянном сравнении данных о выпуске продукции на отдельных участках с данными о поступлении этой продукции на последующие этапы ее обработки, хранения или транспортировки. Эта цепочка проверок доводится до выходных складов предприятия или процедур оформления документов на отгрузку готовой продукции внешним потребителям (по отношению к данному предприятию или объединению).

Важными источниками информации о выпущенной продукции являются открытие и закрытие нарядов на работы, производимые бригадами или отдельными рабочими. Для устранения повторного ввода информации и максимального его упрощения подобные процедуры выполняются на специализированных вводных устройствах — так называемых *регистраторах производства* (*регистраторах информации*).

Отечественный регистратор информации РИ-7501 предназначается для формирования, первичной обработки и регистрации алфавитно-цифровой информации на первичных документах и перфоленте, а также для ввода-вывода ее по телефонному каналу связи. Он осуществляет автоматический ввод с перфокарты и *перфожетона*, *с блока набора условно-постоянных признаков* (БН УПП). В условиях производства на БН УПП может быть набрана заранее дата, номер смены, номер цеха и другие признаки, сохраняющие свои значения в течение длительного времени. На перфокартах могут быть сформированы задания, а перфожетоны (емкостью по 15 десятичных разрядов) используются для идентификации рабочих. Вставляя перфокарту и перфожетон в соответствующие прорези, осуществляют операцию фиксации выдачи задания рабочему. В случае необходимости дополнительные данные могут быть введены вручную с алфавитно-цифровой клавиатуры.

В регистратор встроено специализированное вычислительное устройство, обеспечивающее возможность выполнения арифметических операций, автоматическое формирование порядкового номера сообщения, контроль достоверности вводимых данных, передачу и прием данных по выделенному каналу связи с ЕС ЭВМ (и с ЭВМ типа М-4030). Обработка данных, формирование пакета документа и сообщения осуществляются по программе, вводимой перед началом работы с 80-колонной перфокарты. Устройство изготавливает печатный документ (со скоростью 10 зн/с) на бумажной ленте шириной 30 см с количеством копий до 5 экземпляров.

10.8.3. Учет кадров, труда и зарплаты. Основой для учета труда и зарплаты являются автоматизированные системы (с ручным вводом) *кадрового учета* на отдельных предприятиях и объединениях. Из первичной базы данных, содержащей все требуемые законодательством анкетные данные, автоматически формируется вторичная база для системы учета труда и зарплаты. В нее включаются лишь те анкетные данные, которые влияют на способность выполнения той или иной работы, на зарплату и налоги (занимаемая должность, квалификация, стаж работы, количество детей и др.). К этой базе обращаются при различного рода назначениях и перемещениях. Ее можно использовать также при оформлении нарядов на те или иные конкретные работы.

Процедуры учета результатов труда используют данные учета продукции, закрытия нарядов, а также данные систем контроля качества и проверки исполнения, о которых пойдет речь ниже. Начисление зарплаты в автоматизированных системах производится в соответствии с установленными законом процедурами. Ввиду их многообразия и изменчивости вряд ли целесообразно подробно описывать эти процедуры. Заметим лишь, что автоматизация дает возможность гораздо более полной и достоверной оценки количества и качества труда и позволяет в полной мере довести хозрасчет до каждой бригады и даже до каждого рабочего. При автоматизированном начислении зарплаты сильно облегчаются также безналичные перечисления на счета в сберкассы, о которых говорилось при рассмотрении вопроса об автоматизации кредитно-финансовой системы. Облегчается также реализация различного рода мероприятий, направленных на уменьшение текучести кадров. В частности, в зависимости от размера непрерывного стажа работы на данном предприятии (или в данном учреждении) оно может осуществлять прогрессивно нарастающее кредитование соответствующих работников. Иными словами, такие работники могут делать покупки по безналичному расчету в кредит (до определенного уровня отрицательного сальдо на счете в сберкассе). О других мероприятиях такого рода мы говорим ниже, в пункте, посвященном общегосударственной системе учета.

10.8.4. Учет основных фондов. Автоматизация учета основных фондов производится обычно путем ручного первичного ввода паспортов на соответствующие фонды (землю, строения, оборудование и т. п.), а затем — внесения всех происходящих изменений. Поскольку при таком дифференциальном учете изменений возникающие ошибки могут накапливаться, время от времени производятся *переписи* (*инвентаризация*), в процессе которых осуществляются необходимые поправки. Заметим, что при переписи нет необходимости осуществлять новый первичный ввод всей информации. Более рациональна процедура *машинного вывода* измененных паспортов (лучше всего на дисплей) и сравнение их с наличным состоянием фондов. При отсутствии ошибок соответствующие паспорта оставляются без изменений, при наличии ошибок в них вносятся необходимые исправления, подобно тому, как это делается при автоматизированном редактировании текстов (см. § 10.2).

Исходный первичный ввод может быть также автоматизирован по мере развития систем автоматизированного проектирования. В процессе автоматизированного проектирования новых объектов (или реконструкции старых) должны автоматически приготовляться на машинных носителях паспорта всех вводимых заново или реконструируемых основных фондов.

Процедуры периодической инвентаризации необходимы не только для системы учета основных фондов, но и для всех систем, использующих дифференциальный метод учета. Это относится, в частности, к системам складского и кадрового учета, уже рассмотренным выше.

10.8.5. Учет в розничной торговле. Поскольку об автоматизации складского учета уже говорилось ранее, необходимо рассказать лишь о системе оперативного учета отдельных покупок *). Такая система, возможная лишь в условиях автоматизации, позволяет оперативно учитывать изменение спроса на товары, прогнозировать будущий спрос (путем экстраполяции) и тем самым своевременно реагировать на эти изменения с помощью коррекции заказов промышленности и сельскому хозяйству.

Наиболее просто система учета отдельных покупок реализуется при продаже штучных товаров (в том числе заранее расфасованных). Существуют два основных способа обеспечения возможности такого учета в процессе изготовления таких товаров. При первом способе к товару прикрепляется специальная *бирка* с условным кодом этого товара (единым в общегосударственном масштабе). При продаже бирка отрывается и бросается в специальный контейнер (почтовый ящик). После окончания работы содержимое контейнеров доставляется в ВЦ, где с помощью специальных вводных устройств бирки автоматически прочитываются и их содержимое автоматически вводится в ЭВМ. Разумеется, при этом должны быть обеспечены высокая надежность и достаточная простота считывания. Иными словами, бирки должны быть машинно-ориентированными.

Заметим, что способ учета с помощью бирок весьма эффективен при организации *учета неудовлетворенного спроса.* Имея в обороте запас специальных бирок, отличных от бирок на товарах, для всех видов товаров, которыми может торговать магазин, можно ввести следующую процедуру: если покупатель спрашивает товар, которого в данный момент нет в продаже, то продавец обязан опустить соответствующую бирку в специальный контейнер неудовлетворенного спроса. Обработка этих бирок производится в ВЦ тем же способом, что и обработка бирок на проданные товары.

Второй способ обеспечения учета единичных покупок — проставление кода на самом товаре в процессе его изготовления или на его фабричной упаковке. Код должен быть нанесен так, чтобы специальный кассовый аппарат мог произвести его автоматическое считывание и немедленный ввод в ЭВМ, в памяти которой имеется таблица цен на товары, из которой автоматически

*) Учет в оптовой торговле осуществляется в процессе банковских операций, об автоматизации которых уже говорилось выше.

выбираются и посылаются в кассовый аппарат цена покупаемого товара, а также итоговая сумма (если делается несколько покупок). Такая система особенно удобна тем, что таблица цен может меняться оперативно (даже в течение рабочего дня) без необходимости остановки работы для переинвентаризации.

Описанная система, продолжающая внедряться в мировой практике, достаточно дорога, поскольку требует переоборудования всей системы первичного учета (кассовых аппаратов), а также ЭВМ, работающей в режиме реального времени.

В упрощенном варианте код товара вводится кассиром вручную. При отсутствии непосредственной связи с ЭВМ это означает необходимость ручного ввода также и цены товара. При использовании полных кодов национального классификатора потребительских товаров (а при торговле товарами иностранного производства и межнационального) такая процедура усложняет работу кассира. Упрощение ввода возможно при использовании локальных классификаторов (в крупных универмагах даже разных для разных отделов), особенно таких, у которых цена товара является составной частью локального торгового кода (несущей нагрузку, идентифицирующую товар). Закрепление кассовых аппаратов за отделами позволяет впоследствии при обработке лент кассовых аппаратов на ЭВМ автоматически перейти к любому требуемому классификатору, которым пользуется данный магазин. Разумеется, при этом необходимо оснащение ВЦ специализированными вводными устройствами для автоматического считывания информации с лент кассовых аппаратов (после доставки этих лент в ВЦ).

Использование описанных систем автоматизации учета отдельных розничных продаж вместе с системами автоматизации складского учета дает возможность вести оперативную инвентаризацию товаров (с точностью по крайней мере до одних суток) по огромной специфицированной номенклатуре, достигающей многих сотен тысяч наименований. Поскольку структура спроса на подавляющее большинство товаров зависит от дислокации магазинов, данные подобной инвентаризации позволяют за счет оперативного перераспределения товаров между магазинами существенно увеличить суммарный объем продаж. Опыт зарубежных фирм показывает, что увеличение товарооборота при этом может достигать 10—15%, а в отдельных случаях 20% и более.

Распространение описанных систем автоматизации учета единичных покупок на развесные товары возможно при введении в коды локальных классификаторов показателей, связанных с весом. Разумеется, наилучшие результаты дало бы прямое сочетание цифровых весов (снабженных микропроцессором) с кассовым аппаратом. Однако, поскольку принцип предварительной расфасовки создает многие дополнительные преимущества как для тор-

говли, так и для покупателей, подобные системы вряд ли имеют большие перспективы для развития.

Заслуживают упоминания также системы автоматизации в торговле по предварительным заказам. Если заказ оформляется в виде машинно-читаемого документа, то оформление всех необходимых операций и учет делаются очевидным образом, с минимальной затратой ручного труда. Торговля по заказам при условии автоматизации не только в системе заказов, но и в промышленности, продукция которой поставляется по этим заказам, дает возможность осуществлять новые виды сервиса. Так, многие автомобильные фирмы предоставляют своим заказчикам право индивидуального выбора многих варьируемых особенностей заказываемого автомобиля (цвет, форма руля, внутренняя обивка и др.). В совокупности это приводит к такому числу комбинаций, реализовать которые в нужное время в условиях массового производства без автоматизации управления невозможно.

10.8.6. Учет на пассажирском транспорте. Для городского общественного транспорта и местных автобусных линий учет должен обеспечивать решение задачи определения пассажиро-потоков и точности выполнения расписания движения. Попутно должна решаться и финансовая сторона вопроса: оплата поездок. Весьма проста и вместе с тем достаточно совершенна с этой точки зрения система автоматизации в нашем метрополитене. С одной стороны — автоматическая регистрация и контроль оплаты проезда, с другой — простым взвешиванием массы пятаков, извлеченных из автоматов, оценивать поток пассажиров, входящих на тех или иных станциях.

Однако далеко не столь же просто определить распределение потоков по маршрутам. Для решений этой задачи в автобусах, троллейбусах и трамваях применяют систему автоматического взвешивания пассажиров (с помощью встроенных в рессоры датчиков). Фиксируя этот вес на каждом перегоне, можно оценивать число пассажиров, сошедших и вошедших на каждой остановке *). Закладывая в места стоянок автобусов и троллейбусов под дорожным покрытием специальные индукционные петли, через которые пропускается электрический ток, можно фиксировать автоматически время прихода соответствующих транспортных средств на каждую остановку и тем самым контролировать расписание их движения. Обычно индукционный датчик реагирует просто на величину находящейся над ним металлической массы. Однако, в случае необходимости, нетрудно снабдить каждую транспортную единицу своим индивидуальным ответчиком на

*) Во многих случаях достаточно хорошие исходные данные для учета пассажиро-потоков дает регистрация водителем степени заполнения салона пассажирами по 5-балльной или даже 3-балльной системе нажатием соответствующих кнопок.

сигналы индукционной петли и тем самым сделать учет и контроль движения общественного транспорта еще более совершенным.

Об усовершенствованных автоматических кассах, способных работать с билетами со встроенной магнитной лентой, уже говорилось выше (в § 10.7). Для продажи билетов разной стоимости используются специальные автоматы со встроенными микропроцессорами. Набирая на пульте пункт назначения, пассажир получает на небольшом дешевом специализированном дисплее величину суммы, которую требуется уплатить, и, опустив в кассу соответствующее количество металлических денег, получает требуемый билет и сдачу. Подобные автоматы для работы с бумажными деньгами пока не нашли применения ввиду большой сложности надежного автоматического распознавания банкнот (особенно изношенных).

Для дальних пассажирских перевозок на всех видах транспорта широкое распространение получили автоматизированные системы заказа и оформления билетов. Подобные системы комплектуются специализированными терминалами с наборной клавиатурой и печатающими устройствами, которые впечатывают в пустой бланк билета необходимые реквизиты. Терминалы соединены с ЭВМ, в памяти которой сохраняется информация о заполненных и проданных местах на каждый рейс (поезда, автобуса, самолета, теплохода) в пределах максимальной длительности предварительного заказа. Набирая на клавиатуре терминала требуемые пассажиром условия (прежде всего дату и классность места), кассир через автоматизированную информационно-поисковую систему находит те или иные варианты, и в случае согласия пассажира автоматически оформляет билет, а также производит необходимые изменения в базе данных. Для просмотра различных вариантов желательно, чтобы терминал имел в своем составе алфавитно-цифровой дисплей. Современные автоматизированные системы заказа билетов обеспечивают решение задач оптимизации маршрутов с пересадками (с любыми дополнительными условиями, формулируемыми пассажиром), автоматического резервирования мест в пунктах пересадки и др.

10.8.7. Учет на грузовом транспорте. Для автоматизации учета на грузовом транспорте перевозимые грузы снабжаются машинно-читаемыми сопроводительными документами (накладными). При распределении грузов по отдельным транспортным средствам формируются машинно-читаемая сводная маршрутная карта. В ней указываются наименование, характер и пункты назначения грузов, находящихся в данной транспортной единице, а также дополнительные сведения, необходимые для лучшей организации разгрузки в пункте назначения (схема расположения грузов в трюме судна и т. п.). В пунктах остановок для пере-

компоновки грузов на специальных терминалах происходит переформирование маршрутных карт. При необходимости учета движения грузов по маршруту в любых других запланированных заранее пунктах остановки (в случае железнодорожного транспорта — на сортировочных станциях) маршрутные карты вставляются в специальные регистраторы, которые либо просто фиксируют время регистрации на самой карте, либо передают по линиям связи необходимую информацию в специальный ВЦ, контролирующий движение грузов по маршруту.

В случае железнодорожного транспорта имеются системы (правда, еще не всегда надежно действующие) для автоматического считывания номеров вагонов на замедленном ходу, при прохождении через станции. Поскольку при этом копии маршрутных карт находятся в памяти ЭВМ на центральном пункте учета движения, то получаемая таким образом информация достаточна для ответа на вопрос, где в настоящее время находится любой интересующий нас груз.

Для учета работы грузовых автомобилей, особенно большегрузных машин на дальних перевозках, используется специальное автоматическое регистрирующее устройство, которое в соответствии с заданным режимом фиксирует на магнитной ленте время записи, а также показания спидометра и встроенного в рессоры датчика веса (в моменты остановок или в заданные заранее моменты времени). Возможна, разумеется, фиксация показаний и других приборов. После каждого очередного рейса лента с записями изымается из регистратора и с помощью специального вводного устройства ее содержимое автоматически вводится в ЭВМ. По введенным данным можно судить, насколько выдерживал водитель заданные ему маршрут и режим движения, соотнести эти данные с общим расходом горючего и т. п. Имея столь подробную информацию, можно строить весьма совершенные и эффективные системы стимулирования. Разумеется, необходимым условием действенности подобных систем детализированного учета будет существенно более высокая оплата труда водителей на автомобилях, снабженных такими системами.

10.8.8. Учет в связи и в сетях ЭВМ и ВЦ коллективного пользования. Развитие автоматической междугородной связи заставило разработать и внедрить электронные системы учета длительности телефонных разговоров с автоматическим начислением платы за них. Переход от обычных АТС с электромеханической коммутацией на квазиэлектронные, а затем и чисто электронные АТС (со встроенными в них управляющими ЭВМ) открывает возможность распространить автоматическую систему учета на все виды телефонных разговоров, не только междугородных. Особое значение приобретает развитие таких систем при широком

распространении системы индивидуальных дисплеев и других терминалов, с помощью которых через системы телефонной и телеграфной связи многочисленные потребители будут подключаться к сетям ЭВМ и к автоматизированным банкам данных. Параллельно с развитием государственной сети вычислительных центров будут развиваться и системы автоматического учета услуг, предоставляемых ей системой связи.

Что касается автоматических систем учета работы ЭВМ, то сегодня такие системы встраиваются в математическое обеспечение ЭВМ. С помощью встроенного в ЭВМ таймера учитывается время, на которое каждая задача занимает центральный процессор и другие устройства. Производится также учет объема оперативной и внешней памяти, занимаемой отдельными задачами, объявленных пользователями приоритетов задач, а также объема и вида баз данных и архивов, которые создаются в ВЦ отдельными пользователями. Специальная программа с легко заменяемыми таблицами тарифов на разные виды услуг производит автоматическое начисление платы каждому пользователю за эти услуги.

10.8.9. Учет в системе коммунально-бытового хозяйства. В целях экономии расхода электроэнергии, газа, воды, тепла целесообразно производить учет таких расходов у каждого потребителя системой датчиков (счетчиков и расходомеров), что сегодня делается еще не для всех видов услуг. Главная трудность здесь — необходимость в специальном штате учетчиков, что сильно удорожает систему учета, делая ее в ряде случаев просто нерентабельной (например, при учете расхода воды).

Выход из этого положения заключается прежде всего в объединении систем учета отдельных услуг в единую систему учета. Дальнейшая возможность повышения рентабельности системы учета состоит в возложении на штат учетчиков дополнительных функций (например, функций, связанных с опросами общественного мнения). Еще более радикальное решение вопроса — создание систем автоматического дистанционного опроса датчиков. При условии полной телефонизации подобные системы могут создаваться в процессе нового жилищного строительства. Заметим, что для дистанционного опроса электрических счетчиков может в принципе быть использована обычная электропроводка. Если каждый счетчик снабдить ответчиком, отзывающимся на свою собственную частоту, индивидуальную в пределах данного микрорайона, то такой опрос может делаться из трансформаторных будок без какой-либо дополнительной проводки для передачи сигналов. Организация учета на предприятиях бытового обслуживания (например, в прачечных) в принципе не отличается от организации учета на обычном производстве и сводится к комбинации уже описанных выше компонент учета.

10.8.10. Учет населения и трудовых ресурсов. Сегодня во многих странах уже созданы автоматизированные банки данных для учета населения. Для малых стран такой банк создается в одной центральной ЭВМ с соответствующим объемом внешней памяти (на магнитных дисках). В больших странах он может быть рассредоточен по нескольким регионам с непременным условием возможности автоматизированного обмена информацией между ними (по линиям связи или на машинных носителях). Источником основной информации для подобных банков являются учреждения, регистрирующие рождение, смерть, браки, разводы, а также акты иммиграции и эмиграции. Кроме того, в такие банки обычно направляется дополнительная информация об изменениях постоянного местожительства, изменениях образовательного и имущественного ценза, об отношении к воинской службе и т. п.

Возможна и другая организация дополнительных способов учета населения, а именно — организация специализированных банков данных. В число таких банков включаются уже рассмотренные выше банки с данными о состоянии здоровья населения (§ 10.4). При этом над первичными банками с историями болезней создаются вторичные банки, несущие лишь некоторую обобщенную информацию о состоянии здоровья населения данного региона. Разумеется, при этом должна быть задействована система автоматизированной актуализации подобных вторичных банков. К числу специализированных банков данных, связанных с банком учета населения, относятся обычно также банки, регистрирующие различного рода правонарушения, судимости и т. п. Связь основного банка со специализированными осуществляется через единый учетный код, однозначно идентифицирующий каждого человека данной страны. В одной из наиболее распространенных систем такой код представляет собой десятичное число, первые две цифры которого указывают год рождения (48 — для 1948 г., 01 — для 1901 г.), следующие две цифры — номер месяца рождения. Далее, две цифры отводятся для даты рождения, затем следуют цифры, идентифицирующие место рождения, и наконец, цифры, индивидуализирующие различных людей, у которых все описанные выше старшие разряды кода совпадают.

Среди специализированных банков данных, связанных с учетом населения, особого внимания заслуживают банки учета трудовых ресурсов. Создаваемые в регионах, центрами которых являются крупные города, такие банки фиксируют все трудоспособное население данного района (извлекая необходимые сведения из основного банка учета населения и систем учета труда на предприятиях и организациях), связывая каждого человека прежде всего не с местом его проживания (как в основном банке), а с местом его работы. Основной задачей такого банка является учет динамики соотношения различных категорий трудящихся,

а также перетоков трудовых ресурсов. Особо большое значение подобные банки приобрели бы в том случае, если бы они не просто фиксировали существующие перетоки, а прогнозировали бы будущие и анализировали бы их причины.

С этой целью каждый локальный банк учета трудовых ресурсов должен организовать систему приема и регистрации в памяти ЭВМ запросов трудящихся об изменении места и вида своей работы по возможности с указанием причин такого рода запросов. Постоянный анализ этих причин дал бы ценную информацию для организации систематической работы по их устранению задолго до того, как уже возникшие реально перетоки создадут трудноразрешимые проблемы дефицита работников той или иной специальности. Следует особо подчеркнуть, что меры чисто воспитательного характера по уменьшению перетоков, без устранения причин, их порождающих, чреваты большой опасностью. Сдерживая до поры до времени реальные перетоки, они создают иллюзию мнимого благополучия и тем самым объективно мешают своевременному устранению порождающих причин. Из сказанного отнюдь не следует, однако, что меры воспитательного характера не нужны. Более того, их следует еще более усилить, но при обязательном условии дополнения работой, направленной на своевременную регистрацию пожеланий в банке учета трудовых ресурсов запросов об изменении работы.

Для устранения возможных причин, сдерживающих поступление подобной информации, необходимо гарантировать заявителю тайну его «информационного вклада». Кроме того, банк должен взять на себя обязанность оперативно информировать заявителей о возникновении вакансий, соответствующих высказанным ими пожеланиям. С этой целью все объявления о вакансиях должны своевременно регистрироваться в банке желательно еще до реального возникновения самих вакансий.

10.8.11. Автоматизация подсчета результатов голосования и опросов общественного мнения. В настоящее время разработано и применяется несколько систем автоматизации подсчета результатов голосования. Для небольших коллективов (типа ученых советов) одной из самых удобных является отечественная система Форум. Кресла в зале, в которых производится голосование, снабжаются специальными фигурными четырехполюсными гнездами для подсоединения выдаваемых членам совета *индивидуальных приборов голосования* (ИПГ). На соответствующие позиции (полюсы) каждого гнезда выходят проводники, подсоединяемые к четырем *шинам* — проводам, уложенным под полом зала и выходящим на регистрирующий прибор, расположенный в пульте председателя.

Каждый ИПГ включает в себя калиброванное сопротивление (одинаковое для всех ИПГ), которое с помощью специального

трехпозиционного переключателя может включаться в одну из трех позиций, как это показано на рис. 10.2. Точки, обозначенные 0, +, —, ~, подсоединяются к четырем полюсам фигурного разъема, вставляемого в гнездо, а через это гнездо — к шинам, которым также присвоены обозначения 0, +, —, ~. Таким образом, обладатель ИПГ, манипулируя переключателем, может включать свое сопротивление между шиной 0 и любой из трех других шин.

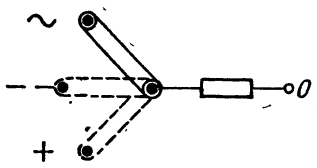

Рис. 10.2

Регистратор на пульте содержит аналого-цифровой преобразователь, который преобразует величину тока от постоянного источника, текущего между шиной 0 и тремя остальными шинами, в количество сопротивления, находящегося в соответствующей позиции. Тем самым возникают три числа: голосующих «за» (+), «против» (—) и воздержавшихся (~). У проводящего голосование на пульте высвечивается общее количество подключенных к системе ИПГ. Убедившись, что все выданные участникам ИПГ подсоединены, он просит членов совета голосовать. Получив подтверждение, что все голосующие установили свои ИПГ в позиции, соответствующие их мнениям, председатель нажимом специальной кнопки фиксирует результат голосования в памяти прибора. Одновременно производится отображение результата на большом экране и впечатывание его в итоговый документ.

Точность аналого-цифрового преобразователя обеспечивает надежное функционирование описанной системы для ста голосующих. При большом числе голосующих систему можно дополнить специальным коммутатором, подключающим последовательно группы мест (числом до 100 мест в каждой группе) к уже описанному регистратору, в состав которого вводится дополнительный (цифровой) групповой сумматор. В случае, когда не требуется точный подсчет голосов, а нужна лишь оценка мнения присутствующих, описанный принцип может применяться в «чистом» виде и в больших залах. При этом с целью упрощения процедуры ИПГ, выдаваемые под расписку, заменяются постоянными трехпозиционными переключателями, встроенными в ручки кресел.

Достаточно проста система автоматизации подсчета результатов голосования с помощью цифрового прибора, регистрирующего суммарное число нажатий каждой из кнопок, имеющихся на его пульте. Голосующий должен зайти в кабину, в которой находится прибор, и нажать кнопку, соответствующую его выбору. При этом с целью избежать возможности многократного нажатия кнопки одним и тем же лицом употребляются различные системы блокировки, включающие прибор для нового голосования

лишь после того, как очередной проголосовавший покинет кабину. Соответствующие меры принимаются также для защиты от повторного прохода кабины одним и тем же лицом. Эти меры упрощаются, если вместо нажатия кнопок голосующим предоставляется право бросить заранее выданные им жетоны в одно из нескольких отверстий. Существуют также системы, автоматически подсчитывающие метки, проставленные голосующими в бюллетенях стандартизованной формы.

Последний способ широко применяется при выборочных опросах общественного мнения. С целью избежать возможных ошибок и упростить процедуру опроса простановка меток в специальном бюллетене-анкете делается опрашивающим лицом на основании устных ответов опрашиваемых.

В системе опроса общественного мнения отметим особо задачу прогноза потребительского спроса, что чрезвычайно важно для плановой экономики. Такая задача может решаться путем выборочных опросов на выставках-ярмарках новых потребительских товаров. Целесообразно также на ранних стадиях проектирования новых потребительских товаров и услуг изготавливать специальные рекламные проспекты, описывающие как сами будущие товары и услуги, так и их ориентировочную цену. Пользуясь такими проспектами, служба опросов общественного мнения могла бы изучать будущий спрос на рекламируемые товары и услуги.

При развитии систем буквенно-цифрового телевидения с возможностью установления двусторонней связи с телезрителями опросы общественного мнения можно производить, формируя вопросы на экранах телевизоров и получая короткие (цифровые) ответы на них, подобно тому, как это делается в системах автоматизированного обучения (§ 10.5). Подобная процедура комплексно автоматизирует не только анализ ответов, но и процедуру самого опроса.

Заметим, что все опросы общественного мнения носят, как правило, выборочный характер. О степени репрезентативности (представительности) выборочных опросов и достоверности выводов, делаемых на их основании, можно судить, используя методы математической теории статистического контроля, основы которой излагаются ниже (п. 10.9.3).

10.8.12. Общегосударственная автоматизированная система учета. Государственная сеть вычислительных центров (ГСВЦ), описанная в гл. VII, позволяет связать все локальные системы автоматизированного учета, рассмотренные в предыдущих пунктах, в общегосударственную автоматизированную систему учета. В такой системе могут реализоваться как регулярные процедуры агрегации учетной информации, так и любые разовые обращения к учетным данным. Благодаря безбумажному методу обмена информацией в ГСВЦ достигается несравненно более быстрая, гиб-

кая и производительная обработка учетной информации, чем в ныне действующей системе ЦСУ. Особенно это касается разовых запросов, которые могут потребовать быстрой агрегации учетной информации для представления ее в любом нетрадиционном разрезе.

Среди регулярных процедур агрегации отметим процедуру вычисления средних текущих зарплат (и тенденций их изменения) по разным категориям работающих в разрезах различных ведомств и регионов. Наличие подобной постоянно действующей системы позволит в значительной мере децентрализовать управление зарплатой, не теряя централизованного контроля над ее средними уровнями, что чрезвычайно важно для предотвращения возможных инфляционных процессов. В случае нежелательных тенденций в изменении средних уровней зарплат меры могут приниматься лишь по отношению к ведомствам и регионам, имеющим наибольшие величины нежелательных отклонений.

Уникальную возможность для резкого улучшения управления экономикой дает система учета (и оперативной агрегации) любых видов материальных ресурсов на складах всех уровней (вплоть до внутрицеховых). Для правильного использования земли, как основного сельскохозяйственного ресурса, огромное значение имеет подробный автоматизированный *земельный кадастр* (доведенный до отдельных полей и участков). В нем наряду с общими сведениями (размеры участков, тип почвенного покрова и др.) должны содержаться данные об истории использования земли, необходимые для выбора правильных севооборотов.

10.9. Автоматизация в системах контроля

Как уже отмечалось выше, функции учета и контроля на практике оказываются зачастую столь тесно переплетенными, что установить границу между ними можно лишь условно. Поэтому ряд вопросов автоматизации контроля (например, контроля за средним уровнем зарплат) уже был нами рассмотрен в § 10.8. Еще ранее (в п. 10.6.5) был рассмотрен вопрос о контроле добросовестности подготовки нормативной информации. Аналогичным образом может быть организован контроль за добросовестностью исполнения любых других, достаточно точно определенных обязанностей. Нам остается сделать несколько дополнительных замечаний об особенностях организации систем автоматизированного контроля в двух важных областях, а именно контроля за исполнением приказов и распоряжений и систем контроля качества продукции.

10.9.1. Автоматизация контроля исполнения. Идея автоматизации контроля исполнения очень проста. Все поручения, данные тем или иным должностным лицам, вместе со сроками исполнения помещаются в память ЭВМ на «счет» соответствующих лиц. По установленной процедуре, за определенное число дней до истечения срока, ЭВМ автоматически формирует напоминание о данном поручении. Если поручение не выполняется, подобные напоминания формируются в момент наступления срока исполнения, а в случае необходимости и после этого срока. Одновременно происходит подсчет дней просрочки исполнения. Напоминания и подсчет просрочек прекращаются после ввода в ЭВМ (лицом, ответственным за контроль исполнения) информации об исполнении (закрытии) данного поручения.

Автоматически для каждого должностного лица ведется суммарный подсчет числа дней просрочек исполнения всех данных ему поручений (возможно, умноженных на коэффициент важности поручений). Для эффективности работы системы весьма важно, чтобы взыскания за неисполнение поручений выносились не случайно, а в соответствии с общим числом дней просрочки. Такая процедура способствует быстрому подъему исполнительской дисциплины, особенно в том случае, когда поручения распределяются между должностными лицами достаточно равномерно. Необходимая информация о степени загруженности сотрудников должна предоставляться системой лицам, дающим поручения, и учитываться ими при определении исполнителя каждого очередного поручения.

10.9.2. Автоматизация контроля качества продукции. При автоматизации контроля качества продукции во всех случаях, когда это возможно, желательно использовать автоматизацию измерений и испытаний, о которых говорилось в предыдущей главе. При отсутствии такой возможности (например, при оценке внешнего вида изделия) оценка, даваемая контролером, должна вводиться в ЭВМ наиболее простым и быстрым способом (например, с помощью нажима на одну из кнопок на специализированном пульте). В случае необходимости одновременного изготовления документа может применяться следующий прием: бланк документа закладывается в металлический переплет с несколькими окошками на его лицевой стороне. Контролер ставит штамп на документ специальным штемпелем через одно из этих окошек. Местоположение штампа на документе (соответствующее выбранному окошку) означает ту или иную оценку качества, сделанную контролером. Одновременно (этим же ударом штемпеля) через датчики, помещенные на задней крышке переплета, посылается соответствующий сигнал в ЭВМ. Возможны и другие системы совмещения операций подготовки документа и ввода информации в ЭВМ.

Хотя во многих современных производствах (особенно в радиоэлектронике) организуется полная система контроля, далеко не утратили своего значения методы выборочного или так называемого статистического контроля (в первую очередь в массовом производстве относительно простых изделий и особенно отдельных деталей). Рассмотрим кратко математические методы, положенные в его основу.

Выборочная дисперсия представляет собой приближение дисперсии (т. е. меры разброса значений) генеральной совокупности (хотя ее математическое ожидание несколько меньше). Она позволяет также оценить дисперсию $\sigma^2_{\bar{x}}$ и среднее квадратичное отклонение $\sigma_{\bar{x}}$ выборочной средней от генеральной средней. Для так называемой *бесповторной выборки* при числе элементов генеральной совокупности, равном N, используется формула

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n-1}} \sqrt{1 - \frac{n}{N}}. \qquad (10.2)$$

При малой величине отношения n/N применяется приближенная формула

$$\sigma_{\bar{x}} \approx \sigma / \sqrt{n-1}. \qquad (10.3)$$

Бесповторная выборка производится следующим образом. Из генеральной совокупности случайным образом выбирается первое изделие, затем из оставшегося множества, также случайным образом,— второе и т. д. Другой тип выборки (называемой *повторной*) получается, если после выборки и измерения очередного изделия его снова возвращают в генеральную совокупность, сохраняя таким образом за ним шанс быть выбранным повторно. Для повторной выборки

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n-1}} \sqrt{1 - \frac{1}{N}} \approx \frac{\sigma}{\sqrt{n-1}}. \qquad (10.4)$$

Приведенные формулы позволяют оценить степень достоверности определения генеральной средней $\bar{x}$. При достаточно больших выборках (на практике для $n > 20$) вероятность p_Δ того, что выборочная средняя будет отличаться (в ту или другую сторону) меньше, чем на Δ $(\Delta > 0)$, определяется по формуле

$$p = \frac{1}{2\pi} \int_{-\Delta/\sigma_{\bar{x}}}^{\Delta/\sigma_{\bar{x}}} e^{-t^2/2} dt. \qquad (10.5)$$

Выражение, стоящее в правой части формулы, являющееся функцией от $\Delta/\sigma_{\bar{x}}$, принято обозначать $\Phi(\Delta/\sigma_{\bar{x}})$. В учебниках по

теории вероятностей обычно приводятся таблицы значений функции $\Phi(X)$, так что для подсчета вероятности p_Δ можно не вычислять интеграл, а воспользоваться этими таблицами. Для малых выборок в курсах по математической статистике приводятся специальные таблицы, показывающие зависимость p_Δ от $\Delta/\sigma_{\bar{x}}$ и от числа n элементов выборки.

Например, при $n = 10$

$$\Delta = 1{,}83\sigma_{\bar{x}}, \quad p_\Delta = 0{,}9;$$

$$\Delta = 3{,}25\sigma_{\bar{x}}, \quad p_\Delta = 0{,}99;$$

$$\Delta = 4{,}78\sigma_{\bar{x}}, \quad p_\Delta = 0{,}999.$$

Используем эти результаты для следующего примера. Пусть в результате бесповторной случайной выборки 10 изделий из 1000 получены значения контролируемого параметра: $x_1 = 1{,}01$; $x_2 = 0{,}98$; $x_3 = 1{,}02$; $x_4 = 1{,}00$; $x_5 = 0{,}99$; $x_6 = 0{,}97$; $x_7 = 1{,}02$; $x_8 = 1{,}01$; $x_9 = 0{,}98$; $x_{10} = 1{,}02$. Требуется на основании произведенной выборки установить интервал, в котором с вероятностью 0,99 заключено среднее значение параметра x для 1000 изделий. Выборочное среднее равно

$$\bar{x} = \frac{1}{10}\sum_{i=1}^{10} x_i = 1;$$

выборочная дисперсия равна $\sigma^2 = 10^{-1}(0{,}01^2 + 0{,}02^2 + 0{,}02^2 + 0^2 + 0{,}01^2 + 0{,}03^2 + 0{,}02^2 + 0{,}01^2 - 0{,}02^2 + 0{,}02^2) = 10^{-5}(1 + 4 + 4 + 1 + 9 + 4 + 1 + 4 + 4) = 3{,}2 \cdot 10^{-4}$, $\sigma = 0{,}018$. Среднее квадратичное отклонение выборочной средней равно

$$\sigma_{\bar{x}} \approx \frac{\sigma}{\sqrt{n-1}} = \frac{0{,}018}{\sqrt{9}} = 0{,}006.$$

На основании приведенного выше табличного значения заключаем, что с вероятностью 0,99 генеральная средняя отличается от единицы (т. е. от выборочной средней) не более чем на $3{,}25 \cdot \sigma_{\bar{x}} = 3{,}25 \cdot 0{,}006 \approx 0{,}02$.

Большой интерес с точки зрения контроля качества представляет оценка ошибки (отклонения от заданного значения) параметра x для генеральной совокупности по результатам выборки. Пусть заданное значение параметра x равно 1. Тогда для приведенной выше выборки ошибки (взятые по абсолютной величине) равны $\delta_1 = 0{,}01$; $\delta_2 = 0{,}02$; $\delta_3 = 0{,}02$; $\delta_4 = 0$; $\delta_5 = 0{,}01$; $\delta_6 = 0{,}03$; $\delta_7 = 0{,}02$; $\delta_8 = 0{,}01$; $\delta_9 = 0{,}02$; $\delta_{10} = 0{,}02$. Выборочное среднее ошибки:

$$\bar{\sigma} = \frac{1}{10}\sum_{i=1}^{10} \delta_i = 0{,}016.$$

Выборочная дисперсия ошибки: $\bar{\sigma}_\delta^2 = 10^{-1}\,(0{,}006^{-2} + 0{,}004^2 + 0{,}004^2 + 0{,}016^2 + 0{,}006^2 + 0{,}014^2 + 0{,}04^2 + 0{,}006^2 + 0{,}004^2 + 0{,}004^2) = 10^{-7}\,(36 + 16 + 16 + 256 + 36 + 196 + 16 + 36 + 16 + 16) = 64 \cdot 10^{-6}$. Выборочное среднее квадратичное отклонение $\bar{\sigma}_\delta = 0{,}008$. Среднее квадратичное отклонение выборочной средней:

$$\sigma_\delta = \sigma_\delta/\sqrt{9} \approx 0{,}003.$$

При $\Delta = 4{,}78$, $\sigma_\delta = 0{,}013$ на основании приведенного выше табличного значения имеем $p_\Delta = 0{,}999$. Таким образом, с вероятностью не менее чем 0,999 можно утверждать, что средняя ошибка в генеральной совокупности не превысит $\bar{\delta} + \Delta = 0{,}016 + 0{,}013 = 0{,}029$.

Третий пример связан с оценкой доли брака p в генеральной совокупности. Предположим, что по техническим условиям изделие бракуется, если отклонение параметра x от 1 больше или равно 0,03. В нашем примере при таком условии бракуется лишь одно (шестое) изделие сделанной выборки. Доля r брака в выборке составит 0,1 (10%). Величину r можно трактовать как среднее величин σ_i, равных 0, если i-е изделие годно, и 1, если i-е изделие бракуется. Дисперсия равна $\sigma^2 = 10^{-1}(r^2 + r^2 + r^2 + r^2 + r^2 + (1-r)^2 + r^2 + r^2 + r^2 + r^2)$. Легко понять, что при любой выборке эта дисперсия выражается формулой $\sigma^2 = (0-r)^2 \times (1-r) + (1-r)^2 r = r(1-r)$. Итак, при подсчете доли брака в выборке

$$\sigma^2 = r(1-r). \tag{10.6}$$

В нашем случае $\sigma^2 = 0{,}1 \cdot 0{,}9 = 0{,}09$, откуда $\sigma = 0{,}3$. Среднее квадратичное отклонение

$$\sigma_p = \frac{\sigma}{\sqrt{n-1}}\,0{,}1.$$

При $\Delta = 3{,}25$, $\sigma_p \approx 0{,}3$ имеем $p_\Delta = 0{,}99$. Таким образом, с вероятностью, не меньшей, чем 0,99, можно утверждать, что доля брака в генеральной совокупности не превысит $r + \Delta = 0{,}1 + 0{,}3 = 0{,}4$. Для более точной оценки доли брака в генеральной совокупности нужно увеличить размер выборки.

В качестве четвертого примера определим размер выборки, при которой с заданной вероятностью 0,99 можно утверждать, что доля брака в выборке отличается от доли брака в генеральной совокупности не более чем на 0,01 (т. е. на 1%).

Трудность задачи состоит в том, что заранее неизвестна величина дисперсии выборки σ^2. Для решения задачи нужно оценить величину σ^2 сверху. Если нет никакой другой оценки, то из формулы (10.2) легко видеть, что максимум равен $\sigma^2 = 0{,}25$ при

$r = 0{,}5$. В ряде случаев на практике эту оценку можно улучшить. Предположим, например, что доля брака r не может превысить 0,1. Тогда из формулы (10.2) имеем $\sigma^2 \leqslant 0{,}1 \cdot 0{,}9 = 0{,}09$ и $\sigma \leqslant 0{,}3$.

При бесповторной выборке n элементов из N среднее квадратичное отклонение σ_p доли брака в выборке от доли брака в генеральной совокупности найдем по формуле

$$\sigma_p = \frac{\sigma}{\sqrt{n-1}} \sqrt{1 - \frac{n}{N}}.$$

Поскольку выборка должна быть, очевидно, большой, для оценки вероятности p_Δ можно воспользоваться формулой (10.1). По таблице значений $\Phi(x)$ находим, что $\Delta = 2{,}6 \cdot \sigma_p$ для $p_\Delta = 0{,}99$. Но нам задано, что $\Delta \leqslant 0{,}01$. Отсюда получаем

$$2{,}6 \frac{\sigma}{\sqrt{n-1}} \sqrt{1 - \frac{n}{N}} \leqslant 0{,}01, \text{ или } \frac{n-1}{1 - n/N} \geqslant 6{,}76 \cdot 10^4 \sigma^2.$$

Так как $\sigma < 0{,}3$, то наше неравенство будет выполнено, если $\frac{n-1}{1-n/N} \geqslant 6{,}76 \cdot 10^4 \cdot 0{,}09 \approx 6 \cdot 10^3$. Тогда $n(1 + 6000/N) \geqslant 6000 + 1$. Единицей в правой части можно пренебречь, так что окончательно имеем

$$n \geqslant \frac{6000}{1 + 6000/N}.$$

При $N = 1000$ имеем $n \geqslant 6000/7 \approx 860$, т. е. выборка должна практически охватить всю генеральную совокупность. Однако, как показывает полученная формула, при очень больших N, например, $N = 600\,000$, для достижения требуемой точности оценки оказывается достаточной выборка 6000 элементов, что может составлять сколь угодно малую долю генеральной совокупности.

При проведении выборок для статистического контроля чрезвычайно важным является соблюдение условия случайности выборки. Дело в том, что любая регулярность выборки, например, выборка каждого десятого изделия, сходящего со станка, может дать совпадение с тем или иным производственным ритмом и исказить результат. Так получится, например, если каждое десятое изделие выпускается в начале смены или после смены режущего инструмента и т. п.

Для достижения полной случайности выборки могут употребляться специальные датчики случайных чисел. В простейшем случае такой датчик генерирует случайную последовательность нулей и единиц, в которой единицы встречаются с заданной частотой (регулируемой в случае надобности). Смена показаний датчика управляется потоком изделий. Если в момент прохода какого-то изделия мимо датчика его показание равнялось еди-

нице, изделие идет на контроль, при нулевом показании датчика изделие в выборку не попадает и не контролируется.

Другой метод — использование специальных таблиц случайных чисел. Если, например, имеется таблица 4-значных случайных чисел и необходимо устроить случайную выборку, скажем 60 изделий из 10 000, то достаточно отобрать изделия, порядковые номера которых будут совпадать с первыми 60 числами таблицы.

В ряде случаев генеральная совокупность естественно делится на части: например, из общего числа N изделий N_1 изделий выпущено на первом станке, N_2 — на втором и т. д. Тогда средние $\tilde{x}_i$ и дисперсии $\tilde{\sigma}_i^2$ берутся по выборкам из каждой части, а общая средняя $\tilde{x}$ и дисперсия средней $\tilde{\sigma}^2$ находятся по формулам

$$\tilde{x} = \frac{\tilde{x}_1 N_1 + \tilde{x}_2 N_2 + \ldots + \tilde{x}_k N_k}{N_1 + N_2 + \ldots + N_k}, \quad \tilde{\sigma}^2 = \frac{\tilde{\sigma}_1^2 N_1^2 + \tilde{\sigma}_2^2 N_2^2 + \ldots + \tilde{\sigma}_k^2 N_k^2}{(N_1 + N_2 + \ldots + N_k)^2}.$$

10.10. Другие информационные технологии

Рассмотренными в настоящей главе случаями далеко не исчерпываются все виды информационных технологий. Достаточно напомнить о таких массовых видах информационных технологий, как библиотечное дело, системы страхования и др. Особое место в списке информационных технологий занимает технология организационного управления. Важность этой технологии заставляет нас посвятить проблемам ее автоматизации и соответствующего математического аппарата особую главу.

Глава XI

АВТОМАТИЗАЦИЯ ОРГАНИЗАЦИОННОГО УПРАВЛЕНИЯ

11.1. Общие сведения об организационном управлении и процессе его совершенствования

Как уже отмечалось выше, *организационное управление* отличается от управления технологическими процессами прежде всего тем, что его объектом являются не столько машины (оборудование), сколько люди, коллективы людей. Разумеется, граница, определяющая это отличие, достаточно условна. Например, управление цехом, оборудованным обычными (не автоматическими) станками, осуществляется через задания работающим на них людям — станочникам. В силу нашего определения управление должно в этом случае рассматриваться как организационное. Заменим в цехе обычные станки на станки с ЧПУ. При этом процедуры оперативного планирования и диспетчерского управления могут остаться в основном практически неизменными. Однако сравнительно небольшое изменение, заключающееся в том, что передача управляющей информации будет теперь производиться прямо на станки, минуя человека, заставляет, согласно принятому определению, отнести соответствующую автоматизированную систему управления к классу технологических систем.

Заметим, что в силу того же определения современные системы управления движением уличного или воздушного транспорта мы также относим к классу организационных систем. Мы специально рассмотрим такие системы несколько ниже. Они занимают промежуточное положение между технологическими и чисто организационными системами. В чистом виде организационные системы, как правило, включают в себя *планирование* на достаточно длительные сроки. В наиболее развитом виде на высоких уровнях (в масштабе отрасли или всего народного хозяйства в целом) возникают разные виды планирования: *долгосрочное*, *среднесрочное* и *краткосрочное* (текущее) планирование. В случае особенно больших сроков (15—20 лет и более) планирование обычно заменяется *прогнозированием*.

При наличии планов задача управления состоит прежде всего в организации их выполнения (а в случае необходимости и корректировки). Обратная связь при таком управлении осуществляется обычно через *систему учета*, проблемы автоматизации которой уже были разобраны в предыдущей главе. Одной из важнейших задач управления является также своевременное пролонгирование планов. При автоматизации организационного управления обычно стремятся перейти к так называемому *непрерывному планированию*. В случае пятилетки переход на непрерывное планирование означает обычно, что по истечении каждого очередного года пятилетний план подвигается на один год вперед *). Годовой план продвигают вперед не реже чем через каждый квартал или даже месяц. Аналогичное положение имеет место и для более краткосрочных планов. Непрерывность планирования ликвидирует неизбежные потери, связанные со стыковкой двух непересекающихся плановых периодов. Обеспечивается гораздо более гибкая адаптация планов к меняющимся условиям, благодаря чему достигается существенное ускорение темпов научно-технического прогресса. Имеются и некоторые иные, более мелкие преимущества (равномерность загрузки плановых органов, большее суммарное время для оптимизации планов и др.).

В отличие от технологического управления, организационное управление не может быть полностью автоматическим, а лишь *автоматизированным*. Это означает, что люди (управленческий аппарат и в первую очередь руководство) являются неотъемлемой составной частью системы управления. Сегодня даже в условиях полной комплексной автоматизации организационного управления за людьми остаются по крайней мере три важные функции: постановка задач управления (в частности, выбор критериев оптимизации), окончательный выбор управленческих решений и придание им юридической силы (утверждение решений) и, наконец, творческая часть труда по выработке планов и решений, выполняемая обычно в диалоге с ЭВМ.

Следует заметить, что граница, отделяющая творческую часть труда от нетворческой, диалектически изменчива. По мере прогресса науки и электронной вычислительной техники все большая часть труда, считающегося сегодня творческим и являющегося поэтому исключительной прерогативой человека, будет формализоваться и передаваться машинам. Что же касается первых двух функций, то в организационных системах при любом уровне развития автоматизации они, по всей видимости, оста-

*) Утверждение пятилетних планов при этом может производиться, как и теперь, один раз в пять лет. Но как расчетная единица пятилетка делается непрерывной.

нутся в своей наиболее существенной части за человеком. И дело здесь не в принципиальной технической невозможности переложить их на плечи машин.

Речь идет скорее о соображениях социально-этического характера, поскольку речь идет об управлении людьми: ведь постановка задач управления в чисто организационной (социально-экономической) сфере в социалистическом обществе связана прежде всего с удовлетворением (личных или общественных) желаний людей. Еще менее приемлемой для человека может оказаться перспектива, когда машина предписывает ему то или иное поведение (т. е. принимает окончательные управленческие решения).

Поскольку люди представляют собой органическую составную часть всякой системы организационного управления, их функции в этой системе должны быть спроектированы с той же степенью тщательности и подробности, как и собственно машинная часть системы (техника, информация и программное обеспечение). Иными словами, *организационные структуры управления* становятся сегодня предметом тщательного *инженерного проектирования.* В отношении структуры и функций персонала в ответственных системах *диспетчерского управления* это в значительной мере имеется уже сегодня. Однако в чисто организационных системах, где просчеты в проектировании организационных структур и функций выявляются не столь быстро и, главное, не столь наглядно, как в случае диспетчерских систем, положение пока значительно хуже. А ведь такие просчеты обходятся обществу в конечном счете гораздо дороже (особенно в крупномасштабных системах).

Необходимо подчеркнуть, что, говоря о проектировании организационных структур, мы имеем в виду весь комплекс вопросов, связанных с функционированием людей (должностных лиц) в составе систем организационного управления. Этот комплекс вопросов подразделяется на два больших класса. Во-первых, это чисто организационные вопросы: структура органов управления, функциональные обязанности органов управления, их подразделений и отдельных должностных лиц, процедуры подготовки и принятия решения, документооборот, контроль исполнения и оценка деятельности всех звеньев системы. Во-вторых, в системах управления народным хозяйством это вопросы хозяйственного механизма: экономические показатели, ценообразование, системы материального и морального стимулирования и т. д.

К этим двум комплексам вопросов, на которых зиждется организационное управление, добавляется еще комплекс вопросов, связанных с технической базой систем управления. Сюда входят оргтехника, техника связи, ЭВМ с соответствующей

периферией, необходимое программное и информационное обеспечение и др.

Необходимо особо подчеркнуть взаимосвязанность всех трех перечисленных комплексов вопросов. Как правило, в современных условиях усовершенствование одного из них не дает желаемых результатов без одновременного согласованного усовершенствования в двух других комплексах. Например, введение описанного в предыдущей главе регистрирующего прибора для учета работы грузового автотранспорта бесполезно без перестройки организационной структуры учета и соответствующей системы стимулирования. Второй пример: ведение строительства в условиях точного согласования сроков строительно-монтажных работ с графиками подачи на стройку необходимых материалов и элементов конструкций в масштабе одного-двух небольших объектов не составляет большого труда и может быть осуществлено за счет мер чисто организационного характера. Одной из таких мер является концентрация управленческих и материальных ресурсов на избранных стройках (как правило, в ущерб всем остальным). Попытка же распространить этот метод на все объекты при ограниченных ресурсах заведомо обречена на неудачу, если не изменить коренным образом техническую базу системы управления и учета и не внести соответствующие коррективы в хозяйственный механизм.

Очень важным вопросом является организация *процесса совершенствования* систем организационного управления. Прежде всего необходимо подчеркнуть, что этот процесс непрерывен в том смысле, что *пока существует организационное управление, будет существовать и постоянная потребность в его усовершенствовании.* Это происходит в силу непрерывного изменения условий, в которых функционируют организационные системы управления. Поэтому даже системы, которые идеально отвечали всем условиям в момент их становления, рано или поздно уменьшают свою эффективность, а иногда просто становятся тормозом дальнейшего развития. Вместе с тем непрерывность процесса совершенствования систем организационного управления *никоим образом нельзя понимать в том смысле, что они должны находиться в процессе непрерывной реорганизации.* Реорганизация (особенно крупная) должна производиться относительно редко, причем она должна (и это самое главное) тщательно и заблаговременно готовиться.

Процесс совершенствования систем организационного управления в общих чертах должен выглядеть следующим образом. Во-первых, организуется постоянное методическое наблюдение за действием существующей системы, тщательно анализируются причины возникновения нежелательных задержек или ошибок в ее работе. Для такого наблюдения максимально исполь-

зуются средства автоматизированного учета и контроля, возможно, специально расширенные и дополненные для решения этой задачи. В случае возможности устранения причин выявленных задержек и ошибок мелкими улучшениями действующей системы такие улучшения делаются незамедлительно. Однако в большинстве случаев подобная практика постепенных мелких улучшений действующей системы управления имеет свои пределы. Рано или поздно наступает такой момент, когда для приведения системы управления в соответствие с изменившимися условиями требуется достаточно радикальная ее перестройка. В ряде случаев это может вылиться в необходимость перехода к принципиально новой технологии организационного управления. Именно в такой *технологической революции* состоит суть перехода на автоматизированное управление (на основе ЭВМ). Непонимание этого момента при разработке и внедрении *автоматизированных систем организационного управления* (АСОУ) является главной причиной малой эффективности таких систем.

Необходимым условием эффективности любой радикальной организационной перестройки (не обязательно связанной с внедрением ЭВМ) является сегодня наличие разработанного во всех деталях *проекта* новой технологии организационного управления и подробного *плана мероприятий* по переходу от действующей технологии к новой. Этот план обязательно должен включать в себя *заблаговременное обучение* всего административно-управленческого персонала работе в новой системе до ее фактического введения. В процессе такого обучения, помимо изучения новых *должностных инструкций* (расписанных во всех деталях), целесообразно включать специальные тренировочные занятия и проверку усвоения особенностей работы в новых условиях. Разумеется, при этом каждый должен знать свое место в будущей системе и, проводя подготовку для работы в ней, продолжать добросовестное исполнение своих обязанностей в старой системе. Лучшим стимулом для этого послужило бы стремление сохранить стаж добросовестной работы в условиях действия описанной в предыдущей главе системы стимулирования за добросовестность. Заметим, что обязательное требование отсутствия безработицы в нашем обществе предусматривает определение в проекте реорганизации места будущей работы для каждого должностного лица действующей системы (исключая возможный уход на пенсию). Не обязательно, конечно, чтобы это место было в рамках именно данной системы.

При описанном способе перехода на новые управленческие технологии (когда, образно выражаясь, «каждый солдат знает свой маневр») не возникает никаких издержек переходного периода, в чем состоит основная беда традиционных реорганизаций. Хорошо известно, что реорганизация, выполняемая по тра-

диционной схеме: сначала реорганизация, потом подбор и расстановка кадров и лишь затем их «притирка» к своим местам, приводит к тому, что реорганизованный (или вновь созданный) орган управления в течение длительного времени (до года и более) работает не с полной отдачей.

Совершенно ясно, что выполнять весь описанный объем работ по совершенствованию систем организационного управления сами эти системы не в состоянии. Необходимо наряду с действующей системой организационного управления иметь также непрерывно действующую *систему совершенствования* этой системы (как это уже делается во многих зарубежных фирмах). По существу, в такой постановке вопроса нет ничего принципиально нового. Ведь именно так организована сегодня система совершенствования технологии материального производства при ее сколько-нибудь радикальных изменениях. Если, например, речь идет о принципиально новой технологии добычи угля, то она разрабатывается отнюдь не теми, кто сегодня сам добывает уголь, а специальными проектно-технологическими институтами. Вообще, в связи с развитием специализации и разделения труда давно сложилось и вошло в практику, что заводы проектируют, строят и реконструируют не те, кто на них потом работает, самолеты — не те, кто на них летает, дома — не те, кто в них живет, и т. д. Иначе было лишь тогда, когда объекты, о которых идет речь, были очень просты, а технология их изготовления — примитивна. Современные системы организационного управления — это, как правило, весьма сложные объекты, *научное проектирование* которых — не менее сложное дело, чем проектирование самолетов и, уж во всяком случае, жилых домов. Однако, к сожалению, продолжает бытовать мнение, что системы организационного управления могут спроектировать и запустить в работу между делом сами управленцы (с помощью комиссий, действующих на общественных началах). Это обстоятельство стало сегодня серьезным тормозом на пути совершенствования системы управления народным хозяйством на действительно научной основе.

В настоящее время существуют и действуют различные системы совершенствования организационного управления. Во всех случаях эффективность таких систем обуславливается так называемым *принципом первого лица.* Суть его состоит в том, что вся работа по совершенствованию управления в том или ином звене должна проводиться по заданиям и под общим руководством первого лица, т. е. главного руководителя соответствующего звена.

Однако, далее, наилучшие результаты получаются в том случае, когда непосредственное повседневное руководство этой работой осуществляется *специальным заместителем первого лица,* освобожденным от всех других обязанностей. При таком руково-

дителе создается необходимый административный аппарат (обычно немногочисленный, но весьма квалифицированный). Кроме того, в его непосредственном распоряжении должны находиться научные и проектно-конструкторские подразделения, специализированные на различных аспектах совершенствования управления. В его же руках должна находиться и система переподготовки административно-управленческих кадров (в рамках соответствующего звена, включая руководящие кадры этого звена). Этому же руководителю обычно подчиняется и вся техническая база систем организационного управления звена (ВЦ с соответствующей периферией и ВЦ, замыкающиеся в данном звене). В иерархических системах управления, состоящих из звеньев различных уровней, описанная организация совершенствования управления повторяет эту иерархию. При этом наряду с непосредственным руководителем — первым в данном звене лицом — ответственность за совершенствование управления имеет и другой прямой (функциональный) руководитель — ответственный за совершенствование управления в следующем по уровню иерархии вышестоящем звене.

В системе совершенствования организационного управления, как и в любых других системах, основанных на творческих началах, помимо обычной системы стимулирования за добросовестную работу должна действовать и другая, не менее мощная система стимулирования — за величину творческого вклада, приносящего реальную пользу. Это могут быть специальные премии или надбавки к зарплате, начисляемые в зависимости от эффективности разработанных систем, продвижения по служебной лестнице и т. п.

11.2. Основные принципы проектирования организационных систем управления

Как правило, проектирование любой организационной системы управления начинается с исследования *объекта*, которым эта система должна управлять (объекта управления). Формируются цели управления и система критериев, позволяющих судить, насколько спроектированная система удовлетворяет этим целям. Цели и критерии согласуются с руководителем (первым лицом) звена, для которого создается система. Далее строится и описывается (в укрупненном неформализованном виде) *база данных об объекте* управления и работающий на этой базе *алгоритм управления*. Укрупненное описание алгоритма достигается за счет использования крупных блоков, например: «решается задача линейного программирования с целевой функцией и ограничениями заданного вида». В случае неформализованных блоков, реализуемых людьми, на этой стадии допускаются описания вида: «по таким-то документам (содержащим предложения или просьбы)

принимается положительное или отрицательное решение». Допускаются ссылки на уже разработанные и хорошо известные управленческие процедуры, например: «оперативно-календарное планирование осуществляется по Новочеркасскому методу» и т. д.

Дальнейшие этапы проектирования заключаются прежде всего в последовательном разукрупнении полученных описаний на отдельные (хотя обычно и взаимосвязанные) *блоки*. С каждым блоком (как и со всей системой в целом) связываются и описываются *потоки входной (выходной) информации*. На определенных стадиях разукрупнения уточняется, какие блоки алгоритма управления будут выполняться людьми, а какие — ЭВМ. Соответствующим образом последовательно уточняется и структура базы данных. На каждом очередном этапе процесса разукрупнения блоков в процесс проектирования вовлекаются, вообще говоря, все новые и новые исполнители. Конечная цель этого процесса — довести описание блоков, исполняемых ЭВМ, до машинных программ, а блоков, исполняемых людьми, до должностных инструкций, написанных столь подробно, чтобы люди с соответствующим (указанным в проекте) образовательным цензом, но, быть может, не имеющие специального опыта работы, могли самостоятельно разобраться в инструкциях, однозначно интерпретировать их и фактически исполнять описываемые в них управленческие действия (сначала, возможно, в замедленном ритме).

Еще ранее (до достижения самой подробной стадии) осуществляется «привязка» отдельных блоков алгоритма к конкретным типам ЭВМ и к конкретным категориям управленческих работников (прежде всего по образованию, а возможно, и по опыту работы). Производятся оценки времени исполнения отдельных блоков и частоты их повторяемости. При этом в одних случаях (*регулярных*) могут возникать детерминированные оценки, а в других (*стохастических*) — вероятностные. Последние даются обычно в виде законов распределения вероятностей величин соответствующих оценок. Ввиду трудности простых расчетов, оценки, о которых идет речь, делаются обычно в результате моделирования на ЭВМ в части, касающейся машинной обработки информации. Для блоков, исполняемых людьми, подобное моделирование может (и в большинстве случаев должно) дополняться прямыми натурными экспериментами (например, для оценки времени, необходимого для ознакомления с теми или иными документами, подготовки проекта решения и т. п.).

После получения оценок становится возможным оценить состав технических комплексов, а также общую численность и квалификацию управленческого аппарата, необходимые для реализации заданного алгоритма управления. Если эти оценки не удовлетворяют заказчика (т. е. первое лицо) или не укладываются

в заданные априори ограничения, то алгоритм управления может быть пересмотрен с целью его упрощения. Поскольку упрощение алгоритма, как правило, уменьшает эффективность управления, при подобных пересмотрах целесообразно применять методы системной оптимизации, описанные в гл. VIII. При этом в случае значительного ухудшения качества управления границы априорных ограничений могут соответствующим образом передвигаться (с разрешения первого лица), находя в конце концов необходимый компромисс между сложностью управляющей системы и обеспечиваемым ею качеством управления.

После нахождения компромисса наряду с процессом дальнейшего разукрупнения блоков и детализации их описания производится проектирование организационной структуры, обеспечивающей своевременное и качественное исполнение всех спроектированных управленческих процедур. При этом разукрупнение блоков должно доводиться до процедур, исполняемых отдельными должностными лицами, а для вопросов, являющихся предметом коллегиального обсуждения,— процедур проведения соответствующих заседаний, рабочих совещаний и т. п. Разумеется, алгоритмом управления должны охватываться не только собственно управленческие процедуры (подготовки и принятия решений), но и все учетно-контрольные, а также различного рода вспомогательные процедуры (связанные с делопроизводством, ведением архивов и т. п.). Таким образом, фактически речь идет о проектировании не одного алгоритма, а целой *системы алгоритмов*.

Следующим этапом проектирования, который также начинается до полного окончания работы по разукрупнению блоков и детализации описаний, является составление *перечня* административно-управленческого персонала, необходимого для выполнения всех спроектированных управленческих процедур. На этом этапе для всех регулярных процедур составляются подробные расписания их выполнения (включая необходимый обмен данными). При этом должны обеспечиваться выявление и исключение в расписаниях возможных н а к л а д о к, при которых одно и то же должностное лицо в один и тот же момент времени оказывается занятым в нескольких несовместимых процедурах. Точные математические методы решения подобной задачи содержат большой элемент перебора и являются поэтому чересчур громоздкими и труднореализуемыми.

На практике предпочитают методику последовательного человеко-машинного улучшения первоначально составленного несовершенного расписания (с накладками). При этом накладки в каждом рассматриваемом варианте выявляются автоматически: по *расписаниям выполнения отдельных процедур и таблице распределения процедур* между отдельными должностными лицами ЭВМ очевидным способом составляет *расписания занятости в регуляр-*

ных процедурах для каждого должностного лица. В расписаниях занятости все накладки сразу выявляются и сообщаются проектанту. По его указаниям (с помощью ЭВМ) вводятся изменения в расписания выполнения регулярных процедур, снова выявляются накладки и т. д., до полного устранения накладок.

Подобный диалоговый режим проектирования позволяет учесть многие дополнительные ограничения и пожелания, направленные на обеспечение большей комфортности для работы в системе (в частности, правильное чередование работы разных видов, обеспечение достаточно больших «окон» в расписании для выполнения нерегулярных процедур и т. д.). Заметим, что отработка расписаний должна предусматривать *подменно-аварийные режимы*, связанные с исполнением управленческих процедур в условиях отсутствия тех или иных сотрудников (отпуска, болезни и т. п.).

После получения достаточно удачного непротиворечивого (без накладок) расписания выполнения регулярных управленческих процедур таблица распределения процедур между сотрудниками распространяется (также в диалоговом режиме) на нерегулярные (случайные) процедуры. При этом учитывается во-первых, характер закрепленных за сотрудниками регулярных процедур (а значит, и их должностное положение), во-вторых, объем их рабочего времени, не занятого регулярными процедурами. Используя методы теории массового обслуживания, излагаемые ниже (в § 11.3), можно рассчитать эти дополнительные нагрузки, чтобы не создавать *информационных заторов* (очередей) при исполнении нерегулярных процедур как в нормальном, так и в подменно-аварийных режимах. Не исключено, что для решения этой задачи придется ввести дополнительное увеличение штата, а в случае необходимости и перераспределение регулярных процедур. Окончательная проверка правильности проведенного распределения обязанностей производится на математической модели проектируемой системы, в результате работы которой должны быть получены графики — гистограммы загрузки каждого сотрудника (и каждой единицы оборудования), а также экспериментальные законы распределения длин очередей на обработку информации и возможных задержек в выработке управленческих решений, ответов на запросы и другой *выходной информации* (как в основном, так и в подменно-аварийных режимах).

Заметим, что если проектируемая система управления действительно нужна для управления объектом (не является лишним звеном), то любые задержки в выдаче информации чреваты, как правило, большими издержками и потому их стремятся по возможности избежать. Однако в этом случае, как показывает теория (см. § 11.3), распределение нерегулярных процедур должно

быть сделано так, чтобы у исполняющих их сотрудников окна в расписаниях занимались с необходимыми *резервами свободного времени*. Эти резервы могут быть использованы для повышения квалификации сотрудников и, в частности, для изучения новых должностных инструкций (прежде всего расписаний и описаний процедур) при подготовке очередной крупной реорганизации (в соответствии с тем порядком, который был описан в предыдущем параграфе).

При распределении управленческих процедур между сотрудниками попутно решается вопрос о распределении сотрудников по соответствующим структурным подразделениям. Процесс такого распределения реализуется в человеко-машинном режиме. Желательно, чтобы структурные подразделения охватывали тех сотрудников, у которых информационные связи, замыкающиеся между ними (в соответствии с закрепленными за ними процедурами), превалируют над внешними информационными связями. В случае, когда подобная возможность отсутствует, при образовании структурных подразделений руководствуются и другими принципами (как правило, плохо формализуемыми). На первое место здесь выступает содержательная родственность процедур, которые предстоит выполнять формируемому подразделению (бухгалтерский учет, материально-техническое снабжение и т. п.).

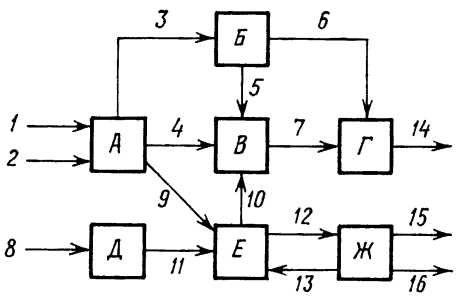

Рис. 11.1

Наглядным инструментом для представления алгоритмов (и систем алгоритмов) в виде совокупности взаимодействующих друг с другом блоков являются так называемые *блок-схемы*. Пример блок-схемы изображен на рис. 11.1. На этой схеме имеются 7 квадратиков — процедур (алгоритмических блоков) и 16 стрелок — информационных потоков. Чтобы использовать рис. 11.1 в качестве *описания алгоритма управления*, необходимо описать процедуры (алгоритмы), выполняемые всеми 7 блоками, а также дать описание всех 16 информационных потоков (состав, форма представления, периодичность и т. п.). Заметим, кроме того, что в состав описаний процедур будут входить, вообще говоря, и описания используемых ими локальных баз данных (принадлежащих отдельным процедурам).

В случае, если все процедуры исполняются людьми, информационные потоки представляются в виде обычных бумажных до-

кументов, так что изображенная на рис. 11.1 схема может рассматриваться как *схема документооборота* в данной системе организационного управления. Чтобы превратить ее в *структурную схему организации*, необходимо закрепить должностных лиц за отдельными процедурами, т. е. построить таблицу распределения процедур. В простейшем случае, когда за каждой процедурой закреплено отдельное должностное лицо, рис. 11.1 может рассматриваться как основа *структурной схемы* проектируемой организации (органа управления). Для дальнейшего уточнения схемы органа в ней выделяют в случае необходимости те или иные структурные подразделения. Исходя из принципа минимизации внешних связей, наиболее целесообразным представляется выделение двух подразделений (подсистем): А, Б, В, Г и Д, Е, Ж. С другой стороны, по принципу родственности выполняемых процедур возможно и другое разбиение на подразделения: А, Д (прием и первичная обработка входящей информации), Б, В, Е (внутренние процедуры подготовки и принятия решений), Г, Ж (оформление и рассылка выходных документов). Часто первое и третье подразделения, осуществляющие внешние связи организации (прием и рассылку документов) объединяют в одно подразделение, называемое *канцелярией* или *секретариатом*.

Необходимо подчеркнуть, что та или иная *структуризация* организации (т. е. выделение в ней тех или иных структурных подразделений) хотя и имеет некоторое значение, но отнюдь не является решающим фактором для эффективного функционирования организации. Главным и решающим является четкое и до конца детализированное описание *алгоритма управления*, разбиение его на отдельные процедуры, закрепление этих процедур за отдельными людьми и обучение людей правильному выполнению этих процедур. Что же касается функций контроля за правильностью исполнения спроектированных процедур, то предполагается, что эти функции являются неотъемлемой составной частью самого алгоритма управления и описываются в виде одной или нескольких составляющих его процедур.

При структуризации (разбиении на блоки) алгоритма управления зачастую удается выделять такие комбинации блоков, которые почти не связаны с остальными блоками. Такие комбинации принято называть *подсистемами*. Алгоритмы в рамках отдельных подсистем можно в принципе проектировать и внедрять до известной степени независимо друг от друга. Разумеется, это не вызовет серьезных ухудшений работы всей системы в целом, при том непременном условии, что подсистемы выделены правильно, т. е. действительно слабо связаны между собой. Такое положение имеет, например, место в подсистемах текущего планирования и управления производством и подсистемах управления капитальным строительством. В то же время планирование

капитального строительства, как правило, неотделимо от долгосрочного планирования производства. Весьма неудачным (приводящим, как правило, к ухудшению эффективности управления) представляется в большинстве случаев все еще широко практикуемое выделение в особые подсистемы (отличные от системы планирования и управления производством) функций материально-технического снабжения и сбыта. Нетрудно понять, что любые изменения в планах производства должны тотчас же находить подкрепление в планах сбыта и материально-технического снабжения, соответствующим образом скорректированных.

При автоматизации организационного управления на основе использования ЭВМ следует помнить, что главным залогом ее успеха является *коренное изменение традиционной технологии организационного управления (основанной на бумажных документах)*. Попытка вставить ЭВМ в традиционную технологию управления эквивалентна попыткам установить реактивный двигатель на телегу или построить мощную тепловую электростанцию, в котлы которой уголь подбрасывался бы лопатами.

Первым (и самым главным) отличием новой технологии организационного управления от традиционной технологии является *форма представления и пути движения информации в системах управления*. Основные потоки информации как внутри отдельных органов управления, так и между ними в традиционной технологии осуществляются в виде бумажных документов. Если при внедрении ЭВМ эти потоки оставить без изменения, как это показано на рис. 11.2, то, как правило, эффект от автоматизации будет сильно снижен или даже станет отрицательным.

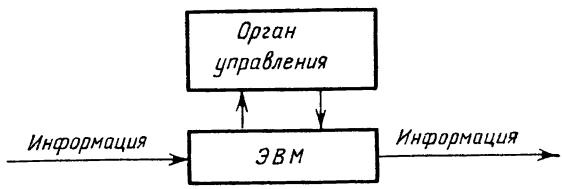

Рис. 11.2

Дело в том, что возникает необходимость иметь большое количество «переводчиков» документов на язык машины и обратно. В качестве таких переводчиков выступает прежде всего персонал, обслуживающий на ВЦ устройства подготовки данных. Расходы на его содержание далеко не всегда в наших условиях могут быть скомпенсированы уменьшением расходов на управленческий персонал, высвободившийся в результате применения ЭВМ. То же самое зачастую происходит и со скоростью решения управленческих задач. Поскольку ручной ввод документа

в ЭВМ — операция существенно более трудоемкая и длительная по сравнению с простым чтением документа человеком, потери времени на вводе далеко не всегда могут быть скомпенсированы экономией, получаемой в результате гораздо более быстрого решения задачи после ее ввода в ЭВМ.

Поэтому главной отличительной особенностью новой технологии организационного управления, называемой *автоматизацией документооборота*, является замыкание всех или по крайней мере подавляющего большинства информационных потоков непосредственно через ЭВМ, как это показано на рис. 11.3. Обмен информацией между отдельными органами управления, а также между ними и объектами управления производится при этом по каналам связи, на машинных носителях или через автоматизированные системы сбора и распределения информации. Многочисленные конкретные примеры реализации подобного безбумажного информационного обмена были рассмотрены в двух предыдущих главах. Обычные бумажные документы (для текущей работы, справки или публикации) в любой момент, по первому требованию лиц, имеющих на то право, могут быть быстро изготовлены на выходных устройствах ЭВМ из огромных информационных запасов, накапливаемых в памяти (в основном внешней).

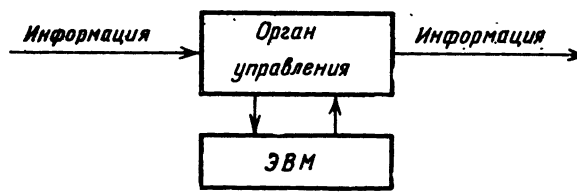

Рис. 11.3

Автоматизация документооборота не только ликвидирует описанные выше источники потерь времени и денег. Гораздо более важно другое: она дает возможность резко увеличить информационные потоки в системах организационного управления, не только не захлестывая людей нескончаемыми потоками бумаг, но даже резко уменьшая эти потоки.

Тем самым создается возможность решать принципиально новые управленческие задачи, недоступные не только для обычных (не автоматизированных) систем управления, но и для ЭВМ без автоматизированного документооборота. В качестве примера такой задачи можно указать рассмотренную в предыдущей главе систему автоматизированного оперативного учета в розничной торговле. *Принцип новых задач* в отдельных случаях может быть реализован и может принести немалый эффект и без автоматизации документооборота. Это имеет место, например, приме-

нительно к некоторым задачам оптимизации долгосрочных планов. Однако, как правило, принцип новых задач и принцип автоматизации документооборота должны реализоваться вместе, поддерживая и дополняя друг друга. Только на этом пути подводится прочная современная база для неограниченного совершенствования организационного управления.

Проблема максимального сокращения потерь времени на ввод даных в ЭВМ получает свое законченное решение, если принцип автоматизации дополняется двумя новыми принципами, а именно принципом *интегральной информационной базы* и принципом *одноразового ввода данных*. Первый принцип состоит в том, что в ЭВМ создается единая для всех управленческих задач *первичная база данных* с минимумом избыточности (необходимым для поддержания сохранности и целостности базы). На эту базу замыкается система учета, осуществляющая ее постоянную оперативную актуализацию и периодически повторяющиеся инвентаризации. Благодаря этому принципу система учета развязывается от собственно системы управления и может проектироваться и внедряться самостоятельно *). Принцип одноразового ввода тесно связан с первым принципом: при наличии единой базы данных любая новая информация помещается в эту базу для последующего, как правило, многократного использования без необходимости повторного ввода. Разумеется, принцип одноразового ввода не исключает дублирования ввода, вызванного соображениями надежности.

Заметим также, что, хотя система учета замыкается на единую информационную базу, управленческие задачи в большинстве случаев могут работать со своими собственными *локальными* базами данных. Однако в правильно построенной системе организация и поддержание таких баз обычно не требуют никакого дополнительного ввода, поскольку они формируются и актуализируются как *вторичные базы данных* непосредственно или последовательно иерархически из исходной (первичной) интегральной базы.

В развитых системах АСОУ локальные базы данных могут выноситься из центральной ЭВМ в сателлитные мини ЭВМ, обслуживающие относительно небольшие группы автоматизированных рабочих мест — *управленческих* АРМ. Такие базы должны снабжаться гибкими системами адаптации языков манипулирования данными к *индивидуальным требованиям* пользователей. В большинстве случаев в качестве управленческих АРМ целесообразно употреблять алфавитно-цифровые дисплеи (желательно

*) В этом одна из причин, заставивших нас выделить проблемы автоматизации учета из проблем автоматизации организационного управления и рассмотреть их заранее в предыдущей главе.

с возможностью изготовления твердых копий выводимой на экран информации).

Еще один важный признак хорошо спроектированного АСОУ — соблюдение *принципа динамической целостности* информационной модели объекта управления. Речь идет, во-первых, о правильном толковании понятия информационной модели объекта. Эта модель должна отражать не только нынешнее состояние объекта, но и его будущее, выраженное в планах развития данного объекта (динамичность модели). Целостность же модели понимается в том смысле, что внесение тех или иных изменений в какую-либо из частей модели должно автоматически инициировать процедуры (обычно человеко-машинные) соответствующих изменений всех остальных, зависящих от нее частей (например, изменения в плане производства должны инициировать изменения в планах сбыта и материально-технического снабжения).

Кроме того, обмен данными между взаимосвязанными процедурами алгоритма управления должен осуществляться внутри ЭВМ через соответствующие промежуточные базы данных, которые также должны рассматриваться как часть информационной модели, но не самого объекта, а системы управления. Тем самым реализуется *принцип системного единства* всех процедур управления.

Очень важным является *принцип типовости* информационного и программного обеспечения АСОУ. Суть его заключается в том, что многие индивидуальные особенности объектов управления, подверженные частым изменениям (число цехов, технологические связи между отдельными участками и т. п.), не «забиваются» жестко в программы, а выносятся в легко заменяемые таблицы, с которыми работают более универсальные программы. Следует заметить, правда, что чрезмерная универсализация программ может вызывать заметное падение их эффективности. Поэтому уровень универсализации программ должен быть разумным компромиссом между уменьшением их эффективности и расширением возможности применений.

К этому принципу непосредственно примыкает принцип *модульности* построения технического, программного и информационного обеспечения АСОУ. Он помогает легко адаптировать АСОУ к меняющимся условиям, производить помодульное развитие и совершенствование системы. Заметим, что принцип модульности позволяет решить одно существенное противоречие в становлении АСОУ. С одной стороны, изложенные выше принципы (особенно принципы единой информационной базы и автоматизации документооборота) требуют достаточно длительного времени для своей реализации. С другой стороны, желательно начать получать отдачу от АСОУ на самых ранних этапах ее развития.

Последнее требование толкает многих разработчиков на антисистемный *позадачный подход*. Суть его состоит в том, что из общего алгоритма управления выхватываются и автоматизируются отдельные процедуры. Для этих процедур организуются собственные базы данных со своим вводом (обычно неавтоматизированным). Основной порок позадачного метода состоит в том, что даже при условии автоматизации всех процедур *единая система управления* не получается и эффект автоматизации оказывается в результате достаточно низким. Пользуясь принципом модульности, можно внедрять отдельные задачи на фоне одновременной работы по созданию системы управления, удовлетворяющей перечисленным принципам. При этом по мере развития общесистемной части подобные локальные очаги автоматизации включаются в качестве готовых модулей в создаваемую систему.

Вместе с принципами первого лица, необходимости отдельной системы непрерывного совершенствования управления и единства организации, социально-экономических механизмов и техники принципы, перечисленные в настоящем параграфе, составляют основу для организации проектирования действительно эффективных АСОУ.

11.3. Теория массового обслуживания

В теории массового обслуживания рассматриваются задачи планирования работ по удовлетворению потока требований, возникающих случайно и требующих различного, заранее точно непредсказуемого времени для их удовлетворения. Такого рода задачи встречаются при проектировании нерегулярных процедур в АСОУ (см. § 11.2). Типичными производственными задачами такого рода являются работы по ремонту и наладке оборудования. Другие примеры задач: обслуживание покупателей в магазине, медицинское обслуживание, продажа билетов, организация информационно-справочной службы, телефонная и телеграфная связь и т. п.

Хотя каждое индивидуальное требование в системе массового обслуживания возникает случайно, потоки таких требований удовлетворяют тем или иным закономерностям статистического характера. Для простейшего, *пуассоновского*, потока вероятность $d\rho$ появления требования в любой бесконечно малый промежуток времени dt пропорциональна (с точностью до бесконечно малых высших порядков) длине этого промежутка: $d\rho = \alpha\, dt$ и не зависит от того, возникли или нет требования в предшествующие моменты времени.

Величина α, называемая *интенсивностью* (или плотностью) потока, может быть постоянной или меняться во времени по определенному закону: $\alpha = \alpha(t)$. В первом случае поток назы-

вается *стационарным*, во втором — *нестационарным*. Термин *простейший поток* обычно применяется к стационарному пуассоновскому потоку.

Пуассоновские потоки описывают реально возникающие потоки требований во всех перечисленных выше примерах. В ряде случаев, однако, приходится рассматривать потоки более общего вида, например, часто встречающиеся на практике *неординарные* (*групповые*) пуассоновские потоки. Моменты возникновения требований в таких потоках распределены как и в обычном, ординарном, потоке, но в каждый такой момент возникает не одно, а целая группа требований. Величина этой группы n также случайна и описывается некоторым законом распределения, задающим тем или иным способом вероятности p_n того, что в группе содержится ровно n требований ($n = 1, 2, \ldots$). Характерными примерами такого потока являются потоки коллективных заявок на билеты, обслуживание организованных групп туристов и т. п.

Имеются и другие обобщения, связанные с тем, что возникновение требования может изменять (обычно уменьшать) вероятность возникновения новых требований в последующие промежутки времени.

Важной характеристикой любого потока требований является время T между моментами возникновения одного требования и непосредственно следующего за ним. Эта величина случайна и характеризуется некоторым законом распределения. Для пуассоновского потока с интенсивностью α вероятность того, что время T заключено в пределах от x до $x + dx$, с точностью до бесконечно малых высших порядков равна $\alpha e^{-\alpha x} dx$. Таким образом, вероятность $P\{T \leqslant A\}$ того, что $T \leqslant A$, равна

$$\int_0^A \alpha e^{-\alpha x} dx = 1 - e^{-\alpha A}.$$

Следовательно, вероятность того, что $T > A$, равна $e^{-\alpha A}$. Это так называемый *показательный закон распределения*. В непуассоновских потоках это распределение отлично от экспоненциального.

Другой важной характеристикой потока является количество N требований, возникающих в заданный интервал времени длины t. Эта величина также случайна и для простейшего (стационарного пуассоновского) потока задается распределением *)

$$P\{N = k\} = -\frac{(\alpha t)^k}{k} e^{-\alpha t}.$$

*) $P\{N = k\}$ есть вероятность того, что в течение времени t будет получено k требований.

Это — так называемое *распределение Пуассона* (по имени которого и назван соответствующий поток).

Кроме перечисленных характеристик, описывающих моменты возникновения требований, важное значение для расчета систем массового обслуживания имеет также рассмотрение закономерностей распределения длительности обслуживания каждого требования. Обычно считается, что эта закономерность является общей для всех требований. Наиболее часто рассматриваются два частных случая: когда длительность обслуживания постоянна и когда она имеет показательное распределение. Во втором случае вероятность $P\{\tau > B\}$ того, что длительность обслуживания τ превышает B, равна $e^{-\lambda B}$, где λ — некоторый постоянный коэффициент. Величина σ, обратная этому коэффициенту ($\sigma = 1/\lambda$), равна, как нетрудно подсчитать, средней длительности обслуживания.

Показательное распределение достаточно хорошо описывает распределение длительности телефонных разговоров и ряд других задач. Для описания многих случаев требуются различные обобщения этого распределения, одним из них является *распределение Эрланга*, предельными случаями которого являются показательное распределение и постоянная длительность обслуживания.

Все законы распределения, характеризующие поток требований, могут задаваться либо на основе априорных соображений общего характера, либо в результате специального экспериментального исследования соответствующего потока. Примером первого задания может служить телефонная станция, обслуживающая большое количество индивидуальных абонентов. Априорным соображением здесь является естественное предположение о независимости телефонных звонков различных абонентов и полной случайности момента, когда такой звонок последует. Из этих предположений следует, как доказывается в теории, что рассматриваемый поток будет пуассоновским. На долю экспериментального исследования остается лишь определение интенсивности α этого потока. Для стационарного потока эта величина может быть получена в результате определения числа N звонков в течение достаточно большого момента времени T и деления первой величины на вторую: $\alpha \approx N/T$. Если априорные соображения, позволяющие определить вид закона распределения, отсутствуют или имеются основания сомневаться в их справедливости, необходимо более детальное экспериментальное исследование рассматриваемого потока. Для этой цели в течение достаточно длительного промежутка времени фиксируются моменты поступления требований и потребовавшиеся фактически длительности обслуживания. На основании полученных данных строятся экспериментальные кривые распределения, которые затем обычно ап-

проксимируются формулой, желательно из числа законов распределения, хорошо изученных в теории.

Кроме закономерностей, характеризующих поток требований, для расчета систем массового обслуживания необходимо задаться той или иной *дисциплиной обслуживания*. Дисциплина обслуживания определяет порядок, в котором данная система обрабатывает поступающие требования. В системе *с потерями* всякое требование, поступившее тогда, когда все средства системы заняты обслуживанием ранее поступивших требований, теряется. Так обстоит дело, например, для обычных АТС. В системе *с очередью* все вновь поступающие требования ждут своей очереди для обслуживания. Встречается и смешанный случай, когда при достижении некоторой длины очереди вновь поступающие требования теряются. Возможен случай *отпугивающей очереди*, когда при увеличении длины очереди возрастает вероятность того, что вновь поступившее требование не станет в очередь и потеряется.

В понятие дисциплины обслуживания входит также порядок обработки поступивших требований. На практике чаще всего встречается случай обработки в порядке поступления. Возможны также случаи, когда требования разделяются по определенным *приоритетам* и обслуживание производится в порядке *приоритетности требований* *). Встречаются также системы, в которых обработка производится по принципу «последнее требование обслуживается первым».

Основным элементом дисциплины обслуживания является порядок, в котором система выделяет средства обслуживания. Этот вопрос возникает в кратных системах — когда любое из поступивших требований может обслуживаться одним из нескольких приборов (рабочих мест), имеющих общее назначение (например, парикмахерская). В этом случае возможны три основные дисциплины обслуживания. Во-первых — циклическое обслуживание, когда первое требование поступает на первый прибор, второе — на второй и т. д. Этот вариант наиболее прост для математического анализа, но относительно редко встречается на практике. Более употребительные два других варианта. В первом из них образуются различные очереди к разным приборам и в момент поступления нового требования принимается решение, к какой из очередей его присоединить. Во втором варианте имеется одна общая очередь, из которой требования поступают на обслуживание к первому из освободившихся приборов.

*) Обычно при этом не происходит прерывания уже начатой обработки. Однако в так называемых системах *с абсолютными приоритетами* любое требование высшего приоритета прерывает обслуживание требований более низкого приоритета.

Задача расчета системы массового обслуживания состоит в том, чтобы определить законы распределения и средние значения различных (случайных) величин, связанных с этой системой. К числу таких величин относятся длина очереди, время ожидания обслуживания, время занятости приборов и т. п.

Ниже излагаются два основных метода решения указанной задачи.

11.3.1. Метод математического моделирования. Сущность этого метода состоит в том, что специальная программа, поставленная на ЭВМ, повторяет шаг за шагом все действия рассматриваемой системы массового обслуживания: определяет случайным образом, в соответствии с заданным законом распределения, моменты появления требований и длительности их обработки, в соответствии с заданной дисциплиной обслуживания производит постановку на очередь, выделяет обслуживающие приборы и т. п. При этом специальная часть программы все время ведет подсчет интересующих величин (длин очередей, простоев обслуживающих приборов и т. п.) и строит экспериментальные кривые распределения вероятностей (частот) — *гистограммы*. По полученным гистограммам вычисляются средние значения, дисперсии, среднеквадратичные отклонения и другие характеристики экспериментальных распределений. Например, пусть было проделано сто вычислений длины очереди. Откладывая по горизонтали различные значения длины, а по вертикали — частоту, т. е. количество раз, в течение которых длина очереди принимала заданные значения, получаем некоторую гистограмму (рис. 11.4), из которой видно, что длина очереди, равная нулю, встречалась 10 раз, а длина очереди, равная трем, 25 раз, что соответствует экспериментальным вероятностям $P_0 = 10/100 = 0{,}1$, $P_3 = 25/100 = 0{,}25$. Среднее значение длины очереди равно

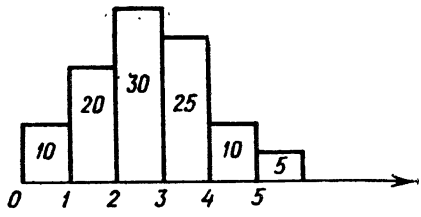

Рис. 11.4

$$n_0 = 0P_0 + 1P_1 + 2P_2 + 3P_3 + 4P_4 + 5P_5 =$$
$$= 0 + 0{,}2 + 0{,}6 + 0{,}75 + 0{,}4 + 0{,}25 = 2{,}2.$$

Наиболее вероятная длина очереди равна, очевидно, 2. Дисперсия равна

$$d = \sum_{i=0}^{5} (i - n_0)^2 P_i = 2{,}2^2 \cdot 0{,}1 + 1{,}2^2 \cdot 0{,}2 +$$
$$+ 0{,}2^2 \cdot 0{,}3 + 0{,}8^2 \cdot 0{,}25 + 1{,}8^2 \cdot 0{,}1 + 2{,}8^2 \cdot 0{,}05 = 1{,}307.$$

Среднее квадратичное отклонение равно

$$\sigma = \sqrt{d} = 1{,}14.$$

Для облегчения построения моделирующих программ используются алгоритмические языки, ориентированные специально на задачи моделирования (симула, симскрипт, слэнг и др.). ЭВМ снабжается специальным устройством или программой, генерирующей случайные числа, обычно числа заданного отрезка (например, [0, 1]), с равной вероятностью выбора любого такого числа. Специальные операторы языка моделирования могут переводить получаемое таким образом равномерное (равновероятное) распределение в распределения по любым другим, употребляющимся на практике законам, например, в экспоненциальное распределение, распределение Пуассона и т. п. Языки моделирования обычно располагают также операторами для регистрации очередей, построения гистограмм и т. п.

Методом математического моделирования в принципе могут быть решены любые задачи по расчету систем массового обслуживания, в том числе и сложных систем с переменной дисциплиной обслуживания (зависящей от состояния системы). Отрицательной чертой этого метода являются большие затраты машинного времени, поскольку для получения устойчивых статистических характеристик исследуемой системы приходится производить большое число (многие тысячи) экспериментов.

11.3.2. Аналитические методы. Эти методы позволяют представить интересующие нас характеристики в виде функций от исходных характеристик. К сожалению, простые аналитические (формальные) выражения для такого рода функций удается получать лишь в простейших случаях.

Рассмотрим пример аналитического расчета одной из простейших систем массового обслуживания. Пусть поток исходных требований является стационарным пуассоновским потоком с интенсивностью α. Длительность обслуживания задана экспоненциальным законом

$$P\{T > B\} = e^{-\alpha B}.$$

Дисциплина обслуживания предусматривает наличие очереди, обрабатываемой в порядке поступления требований. Искомые величины: длина очереди n (случайная величина) и коэффициент полезного действия η, т. е. среднее относительное время, в течение которого система занята обслуживанием.

Строго говоря, для ответа на указанные вопросы необходимо, кроме уже приведенных данных, знать состояние системы в начальный момент времени (т. е. длину очереди и признак, занята она обслуживанием или нет). Однако в теории для широкого класса систем массового обслуживания, заведомо включающего

рассматриваемую систему, доказывается наличие у них так называемого свойства *эргодичности*. Сущность этого свойства состоит в том, что для моментов времени, достаточно удаленных от начального, влиянием начального состояния на состояние системы в рассматриваемый момент можно пренебречь. Для любой эргодичной системы обычно ищется устойчивое (случайное) решение, к которому она стремится при $t \to \infty$. Расчет переходного процесса, в течение которого система переходит в устойчивое состояние (с некоторой заданной точностью), представляет собой самостоятельную задачу (как правило, гораздо более трудную).

Переходя к решению поставленной задачи, заметим прежде всего, что состояние системы можно охарактеризовать лишь одним числовым показателем — длиной очереди n, если считать, что случай $n = 0$ соответствует нулевой очереди при занятости системы, а случай $n = -1$ — нулевой очереди при свободной (не занятой обработкой) системе. Заметим далее, что если система в течение времени t_0 была занята обработкой некоторого требования, то вероятность окончания обработки в промежутке $[t_0,\ t_0 + \Delta t]$ есть условная вероятность

$$\frac{P\{t_0 \leqslant t \leqslant t_0 + \Delta t\}}{P\{t \geqslant t_0\}}.$$

Для показательного распределения e^{-2t} величина в числителе равна $\lambda e^{-2t_0}\Delta t$ (с точностью до бесконечно малых более высокого порядка, чем Δt), а в знаменателе e^{-2t_0}. Таким образом, вероятность окончания уже начатого обслуживания в любой бесконечно малый промежуток времени равна $2\Delta t$ и не зависит от времени начала обслуживания.

Обозначим через $p_i(t)$ вероятность того, что рассматриваемая система в момент времени t находится в i-м состоянии ($i = -1, 0, 1, 2, \ldots$). Вычислим те же вероятности в близкий момент времени $t + \Delta t$. Для того чтобы система в момент времени $t + \Delta t$ оказалась в состоянии -1, имеются две возможности. Первая состоит в том, что система была в этом состоянии (-1) в момент времени t и в промежутке от t до $t + \Delta t$ не приняла ни одного нового требования. Вероятность такой возможности равна*) $p_{-1}(t)(1 - \alpha\Delta t)$, поскольку вероятность возникновения нового требования за элементарное время Δt для рассматриваемого потока равна $\alpha\Delta t$. Вторая возможность состоит в том, что в момент времени t система находилась в состоянии 0, а за последующий отрезок времени Δt закончилось обслуживание обрабатывавшегося требования и не поступило нового требования. Вероятность

*) Все вычисления производятся с точностью до бесконечно малых величин порядка, более высокого, чем Δt.

такой возможности равна $p_0(t)\lambda\Delta t(1-\alpha\Delta t)\approx p_0(t)\lambda\Delta t$. Объединяя обе эти (независимые) возможности, получаем соотношение

$$p_{-1}(t+\Delta t)=p_{-1}(t)\,(1-\alpha\Delta t)+p_0(t)\lambda\Delta t,$$

или

$$\frac{p_{-1}(t+\Delta t)-p_{-1}(t)}{\Delta t}=\lambda p_0(t)-\alpha p_{-1}(t).$$

Переходя к пределу при $\Delta t\to 0$, получаем дифференциальное уравнение

$$\frac{dp_{-1}(t)}{dt}=\lambda p_0(t)-\alpha p_{-1}(t). \qquad (11.1)$$

Для состояния $i\neq -1$ рассмотрим три случая.

1. В момент времени t состояние системы было $i-1$; за время Δt пришло новое требование и не закончилась обработка старого. Вероятность этой возможности равна

$$p_{i-1}(t)\alpha\Delta t(1-\lambda\Delta t)\approx p_{i-1}(t)\alpha\Delta t.$$

2. В момент времени t система находилась в состоянии $i+1$; за время Δt закончилась обработка старого требования, а новое требование не пришло. Вероятность равна

$$p_{i+1}(t)\lambda\Delta t(1-\alpha\Delta t)\approx p_{i+1}(t)\lambda\Delta t.$$

3. В момент времени t система была в состоянии i; за время Δt не закончилась обработка старого требования и не пришло ни одного требования. Вероятность равна

$$p_i(t)\,(1-\alpha\Delta t)\,(1-\lambda\Delta t)\approx p_i(t)-(\alpha+\lambda)p_i(t)\Delta t.$$

Существует и четвертый случай, когда система сохраняет состояние i, закончив за время Δt обработку старого требования и получив за это время новое требование. Однако вероятность такой возможности, равная $p_i(t)\alpha\Delta t\lambda\Delta t$, есть бесконечно малая величина порядка, более высокого, чем Δt, и поэтому должна быть отброшена.

Объединяя три случая, получаем дифференциальное уравнение

$$\frac{dp_i(t)}{dt}=\alpha p_{i-1}(t)-(\alpha+\lambda)\,p_i(t)+\lambda p_{i+1}(t). \qquad (11.2)$$

Решение полученной (бесконечной) системы дифференциальных уравнений (11.1) и (11.2) полностью определяет (при задании начального состояния) дальнейшее поведение системы. Пользуясь свойством эргодичности системы (заведомо справедливым при $\alpha<\lambda$), будем искать лишь установившееся решение, для которого, как нетрудно понять, $p_i=\text{const}$ и производные $dp_i(t)/dt$ $(i=-1,\ 0,\ 1,\ \ldots)$ должны обращаться в нуль. В результате

имеем систему обычных алгебраических уравнений:

$$\begin{gathered}\lambda p_0 - \alpha p_{-1} = 0,\\ \alpha p_{i-1} - (\alpha + \lambda) p_i + \lambda p_{i+1} = 0, \quad i = 0, 1, \ldots\end{gathered} \tag{11.3}$$

Эта система позволяет по p_{-1} найти p_0, по этим двум величинам — p_1, по величинам p_0 и p_1 — величину p_2 и т. д. Если обозначить через r величину α/λ, то решение системы (11.3) запишется в виде

$$p_i = r^{i+1} p_{-1}, \quad i = 0, 1, 2, \ldots$$

Для определения значения p_{-1} воспользуемся тем очевидным соображением, что сумма вероятностей всех состояний, равная

$$\sum_{i=-1}^{\infty} p_i = p_{-1} \sum_{i=-1}^{\infty} r^{i+1} = p_{-1} \frac{1}{1-r},$$

должна равняться 1. Отсюда

$$p_{-1} = 1 - r, \tag{11.4}$$

$$p_i = r^{i+1}(1-r). \tag{11.5}$$

При выводе соотношений (11.4), (11.5) предполагаем, что $r = \alpha/\lambda < 1$. В противном случае полученные для вероятностей значения приводят к абсурду. Этот результат легко понять, если вспомнить, что $1/\lambda$ есть средняя длительность обслуживания, а $1/\alpha$ — средняя величина интервалов времени между моментами поступления очередных требований. При $r > 1$ первая величина больше второй, и, как легко понять, система не будет справляться с обработкой поступающих требований, очередь будет неограниченно расти, и стремление к какому-либо устойчивому состоянию не будет иметь места.

Полученное распределение вероятностей различных состояний (11.4), (11.5) представляет собой *геометрическое распределение*. Величина $1 - p_{-1} = r$ есть искомый коэффициент полезного действия системы:

$$\eta = r = \alpha/\lambda.$$

Вероятность того, что длина очереди будет не меньше k, равна

$$\sum_{i=k}^{\infty} r^{i+1}(1-r) = \frac{r^k}{1-r}(1-r) = r^k.$$

Отсюда следует, что при стремлении «выжать» из системы большой к. п. д. неизбежно будет расти очередь. Так, при $\eta = 0{,}9$ вероятность того, что в очереди находится не менее пяти требований, равна $0{,}9^5 = 0{,}59$.

Средняя длина очереди l в геометрическом распределении равна

$$\sum_{i=0}^{\infty} ir^{i+1}(1-r) = \frac{r}{1-r}.$$

При $r = 0{,}9$ приходим к значению $l = 9$. Таким образом, стремление к максимальной загрузке обслуживающих приборов неизбежно приводит к росту очередей. При расчете систем массового обслуживания необходимо поэтому находить разумный компромисс между этими противоречивыми тенденциями.

Необходимость поиска такого компромисса — одна из особенностей систем со случайными потоками требований, отличающихся от регулярных систем, где в принципе достижим сколь угодно высокий к. п. д. использования имеющихся ресурсов.

Для решения задачи на оптимум в случае систем массового обслуживания необходимо прежде всего задаться удельной (за единицу времени) ценой a потерь в результате пребывания требования в системе. Как показывается в теории, в рассмотренной выше простейшей задаче массового обслуживания вероятность того, что время t пребывания требования в системе заключено в бесконечно малом интервале $[t,\ t+dt]$, равна $(\lambda-\alpha)e^{-(\lambda-\alpha)t}dt$. Отсюда среднее время пребывания требования в системе равно

$$\int_0^{\infty} t(\lambda-\alpha)e^{-(\lambda-\alpha)t}dt = \frac{1}{\lambda-\alpha}.$$

За единицу времени в среднем возникает α требований. Тогда общие потери в результате задержек в системе будут в среднем равны

$$\frac{\alpha}{\lambda-\alpha}a = \frac{ra}{1-r}.$$

Уменьшить эти потери можно только за счет наращивания обслуживающих мощностей, суммарная пропускная способность Q которых обратно пропорциональна $r = \alpha/\lambda$, т. е. к. п. д. использования. Удельные расходы, отнесенные на единицу времени работы системы, которые необходимы для создания и эксплуатации этих мощностей, равны c/r, где c — некоторый постоянный коэффициент.

Суммарная цена обслуживания и происходящих в результате его ожидания потерь есть

$$f = \frac{ar}{1-r} + \frac{c}{r}.$$

Нетрудно проверить, что минимум этой функции достигается при $r = 1/(1+\sqrt{a/c})$. Это есть оптимальный к. п. д. рассматриваемой системы массового обслуживания.

11.4. Автоматизированные системы диспетчерского управления

Системы диспетчерского управления предназначены для управления сложными человеко-машинными системами в реальном масштабе времени. Они представляют собой как бы переходный этап от систем технологического управления к чисто организационным системам. Подобные системы нередко автоматизировались еще до появления ЭВМ (как это имело, например, место при диспетчерском управлении в энергосистемах или на железнодорожном транспорте). Применение ЭВМ позволяет оперативно прогнозировать изменения состояния объекта управления и тем самым (предвидя заранее многие назревающие ситуации) существенно улучшить управление. С этой целью в ЭВМ закладывается динамическая модель управляемого объекта, используемая различными способами (и в различных целях) в различных системах. Имея в виду большое разнообразие существующих систем диспетчерского управления, рассмотрим лишь некоторые примеры, наиболее характерные теми или иными особенностями.

11.4.1. Управление воздушным движением. Диспетчерское управление воздушным транспортом охватывает все стадии процесса его использования, т. е. погрузку и разгрузку самолетов, руление, взлеты и посадки, движение в зоне круга аэродрома и, наконец, управление движением самолетов непосредственно на маршрутах. Остановимся подробнее именно на последнем этапе, поскольку возникающие здесь проблемы и методы их решения характерны для многих других систем диспетчерского управления.

На большинстве маршрутов регулярного движения самолетов их положение фиксируется с интервалами порядка десяти секунд наземным радиолокатором кругового обзора. Высота и скорость (вектор скорости) полета также могут быть определены с помощью наземной аппаратуры, хотя чаще всего эти величины измеряются бортовой аппаратурой и передаются диспетчеру зоны, в которой находится самолет в данный момент. При диспетчировании с помощью ЭВМ все эти сведения, равно как и сведения о типе самолета, номере рейса, пункте прибытия и др., помещаются в память ЭВМ и высвечиваются в виде изображений самолетов с соответствующими формулярами на специальных графических дисплеях — АРМ диспетчеров зон. Хотя пересекающиеся или встречные маршруты обычно разносятся по высоте (эшелонируются), однако не исключено, что в случае ошибок в определении высот возможны нежелательные сближения самолетов в воздухе и даже столкновения. С целью избежать возникновения подобных ситуаций ЭВМ, «зная» маршруты движения, скорости и высоты самолетов, все время экстраполирует их движение на определенное время вперед. В случае, когда такая

экстраполяция укажет на возможность опасного сближения, изображения сближающихся самолетов на экране дисплея начинают мигать, приковывая тем самым к себе внимание диспетчера. В режиме диалога с диспетчером вырабатывается решение, которое затем передается на борт самолетов. Оно может состоять в изменениях скоростей самолетов или их эшелонирования по высоте. Возможно также, если есть, гарантированное разнесение самолетов по высоте (и притом достаточное для безопасного расхождения), разрешить им следовать далее в прежнем режиме.

ЭВМ также решает задачу определения расчетного времени вхождения каждого самолета в зону круга аэропорта назначения. С целью избежать очередей самолетов, ожидающих разрешения на взлет и посадку, рассчитываются изменения путевых скоростей самолетов (с учетом ветра), направляющихся к любому данному аэропорту. Эти изменения (передаваемые на борт через систему диспетчерской связи) обеспечивают равномерность прибытия самолетов в аэропорт даже при различного рода нарушениях расписания их движения.

11.4.2. Управление уличным движением. Для получения информации о движении автотранспорта все контролируемые перекрестки снабжаются датчиками, реагирующими на проезд перекрестка отдельными транспортными средствами. Сегодня в большинстве случаев для этой цели используются индукционные датчики, о которых уже упоминалось выше (в п. 10.8.6). Петля провода с током, уложенная под дорожным перекрытием, способна не только регистрировать прохождение над ним отдельных транспортных средств, но и определять их тип (легковые, грузовые автомобили, автобусы и т. д.), а если надо, то и скорость их движения. Для различения спецавтомобилей (скорая помощь, пожарные машины и др.) они снабжаются специальными ответчиками, которые могут не только передать в петлю информацию о типе спецмашины, но и в случае необходимости полностью идентифицировать ее. Все данные от датчиков передаются в управляющую ЭВМ.

Главное отличие рассматриваемой системы от системы управления воздушным движением состоит в том, что (за исключением отдельных спецмашин) наблюдения за движением каждой индивидуальной машины не ведется и маршруты их движений диспетчерской системе неизвестны. Поэтому вместо прогнозирования движения каждой транспортной единицы прогнозируется *величина их потоков*. В простейшем случае используется оперативное прогнозирование на один шаг, т. е. на (среднее) время перемещения наблюдаемых потоков к следующим по направлениям их движения контролируемым перекресткам. Исходя из такого прогноза, вычисляются моменты переключения сигналов управляемых от ЭВМ светофоров с целью минимизировать сум-

марные задержки транспорта на регулируемых перекрестках (с учетом необходимости внеочередного пропуска различного рода спецмашин).

Кроме того, могут делаться более долгосрочные прогнозы, основанные на статистическом материале или специальных исследованиях. Они учитывают регулярные (или почти регулярные) изменения потоков, возникающие в течение суток (начало и конец работы), недели (массовые выезды за город в конце недели) или более длительных периодов. То же касается массовых мероприятий разового характера (демонстраций, спортивных соревнований и т. п.). Для быстрого рассасывания подобных, заранее планируемых потоков вырабатываются специальные алгоритмы управления.

Заметим, наконец, что широкое распространение дешевых микрокомпьютеров делает возможным создание своеобразных «автоматических штурманов» на индивидуальных транспортных средствах, включая личные автомобили. Установленная на автомобиле микро-ЭВМ должна при этом иметь возможность устанавливать автоматическую двустороннюю радиосвязь с центром управления уличным движением. Водитель вводит в память своего штурманского микрокомпьютера начальный и конечный пункты планируемого маршрута, после чего «штурман», общаясь с центром управления, вырабатывает полный маршрут движения, наиболее целесообразный в данных конкретных условиях. При широком развитии подобных систем (начало которым положено в Японии) центр управления может располагать предполагаемыми маршрутами значительной части транспортных средств и перейти на более совершенные методы диспетчирования (аналогичные тем, которые были изложены в предыдущем пункте). Разумеется, при этом потребуется значительное увеличение мощности управляющих компьютеров.

11.4.3. Диспетчерское управление на производстве. Рассмотрим два характерных примера такого управления. Первый пример относится к диспетчированию отдельных производственных участков, работающих друг на друга через *буферные емкости* (склады, резервуары, газгольдеры и т. п.). Смысл диспетчерского управления здесь состоит в следующем.

Отслеживая и прогнозируя потоки *выходной продукции* (всего предприятия в целом) и учитывая при этом возможные остановки тех или иных производственных участков (для переналадки, профилактики или в результате аварий), а также ограниченность буферных емкостей,— находить *оптимальные режимы* работы отдельных производственных участков. При этом обычно требуется безусловное поддержание планируемого потока выходной продукции (с высокой, устанавливаемой заранее степенью вероятности). В качестве критериев оптимизации чаще всего выступают

различного рода технико-экономические критерии (наиболее щадящие режимы эксплуатации оборудования, минимальная себестоимость продукции, минимальный расход энергии, лучшее качество продукции и т. п.). Если режимы работы участков устанавливаются локально (без учета темпа выпуска конечной продукции), то их оптимизация (например, по показателю наивысшей производительности) может принести не пользу, а вред. Работая что называется «на всех парах», участок может внезапно остановиться в результате переполнения буферной емкости. В результате производившаяся «гонка» (связанная, как правило, с дополнительными затратами, усиленным износом оборудования и т. п.) окажется в конце концов напрасной, ибо тех же самых конечных результатов можно было добиться ценой меньших усилий.

Второй пример связан с оперативным управлением взаимосвязанным множеством работ, между которыми отсутствуют буферные емкости. Наиболее характерный пример такого рода — сборочный конвейер. В своем классическом виде сборочный конвейер имеет лишь один управляемый параметр — *темп* движения конвейера. Управлять таким процессом (ввиду его однопараметричности) просто. Однако при этом возникают неудобства для работающих. Для одних заданный темп может оказаться чрезмерно высоким, а для других — низким. Поскольку степень подготовленности рабочих и их физическое состояние — величины переменные, технологам, планирующим разбиение операций на конвейере, практически невозможно создать условия *равнонагруженности* каждого рабочего места. Этого можно достичь лишь в результате *оперативного управления* числом рабочих мест для каждой операции, а также, возможно, и характером самих операций.

Наиболее простой способ решения этой задачи — создание бригад, за которыми закрепляются крупные операции на конвейере. Разукрупнение этих операций между членами бригады производится оперативно самой бригадой в зависимости от степени их работоспособности в каждый данный момент. Расчеты перераспределения работ могут выполняться методами теории массового обслуживания (в сложных случаях — с помощью ЭВМ). Еще более радикальное решение задачи — построение конвейера не в виде одной линии, а в виде направленного графа с продольными и поперечными связями вида, изображенного на рис. 11.5. Горизонтальные линии (последовательности стрелок) на этом графе могут рассматриваться как обычные (линейные) конвейеры, которые настраи-

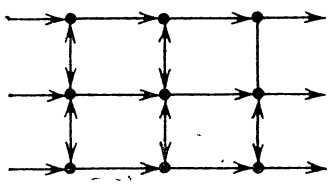

Рис. 11.5

ваются на одну и ту же работу. На каждую операцию при этом имеется несколько рабочих мест (на рис. 11.5 — три). Если на каком-то рабочем месте имеет место отставание от графика, то с помощью поперечных связей (на рис. 11.5 — вертикальных) можно перераспределять работу между этими местами.

При практической реализации этой идеи обычно используют одну транспортную линию, вдоль которой на специальных тележках перемещаются собираемые объекты (например, телевизоры). Рабочие места расположены не прямо на этой линии, а рядом с ней (рис. 11.6). Отцепляя тележку от транспортера, рабочий перемещает ее на свое рабочее место, выполняет требуемую операцию и снова возвращает тележку на транспортер для перемещения к следующей группе рабочих мест (ответственных за выполнение следующей технологической операции). Управление подобным конвейером осуществляется с помощью изменения не только общего темпа работы, но и числа задействованных рабочих мест на каждой операции (на рис. 11.6 эти числа равны соответственно 3, 2, 3, 2, 4). При условии овладения каждым рабочим несколькими операциями подобное управление может производиться без изменения общего числа людей, занятых на конвейере. Алгоритм управления настраивается на обеспечение наиболее высокой суммарной производительности с учетом реальной возможности и желания каждого работающего. За счет этого, как правило, удается добиться одновременного решения двух противоречивых задач — увеличения производительности и повышения уровня комфортности работы на конвейере.

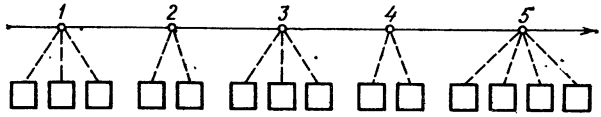

Рис. 11.6

В случае, когда детали и узлы, монтируемые на каждом рабочем месте, невелики по размерам (как это имеет место для большинства операций по сборке телевизоров), они могут запасаться впрок на рабочих местах, пополняясь лишь время от времени. При тяжелых деталях (например, при сборке автомобилей), помимо основного (сборочного) конвейера, возникает необходимость иметь транспортные конвейеры для доставки в нужные моменты времени на нужные рабочие места нужных деталей и блоков. Для сборки отдельных блоков (например, автомобильного мотора) организуются дополнительные сборочные конвейеры. Процессы взаимодействия всех этих конвейеров должны быть строго *синхронизированы* по времени в соответствии с заданным темпом

выпуска конечного изделия. Хотя расчеты по синхронизации сами по себе тривиально просты и вряд ли требуют дополнительных пояснений, при переходе к управлению числом задействованных рабочих мест по всем взаимосвязанным линиям они становятся практически неосуществимыми вручную и требуют наличия центральной диспетчерской ЭВМ.

При автоматизации управления системой конвейеров (сборочных, отладочных и транспортных) возникает возможность *индивидуализации* сходящих с конвейера изделий. Например, многие автомобильные фирмы предоставляют возможность заказывать автомобили с различными индивидуальными особенностями (форма руля, тип радиоприемника, цвет и материал отделки салона и т. п.). ЭВМ, осуществляющая диспетчеризацию сборки, обеспечивает при этом управление навеской деталей (заготовок) на транспортные конвейеры таким образом, чтобы к моменту начала очередной сборочной операции на данное рабочее место попала нужная деталь. Разумеется, технологи обеспечивают при этом максимально возможную идентичность самих сборочных операций на каждом рабочем месте, независимо от вида подаваемых для этой цели деталей.

В современных машиностроительных и приборостроительных предприятиях, выпускающих массовую или крупносерийную продукцию, *синхронизация* распространяется не только на сборочно-транспортные, но и на все остальные операции (включая работу заготовительных цехов). Однако, благодаря возможности создания *буферов* в виде запасов деталей и заготовок на разных стадиях обработки, требования к точности решения задачи синхронизации снижаются. Необходимо помнить вместе с тем, что даже при отсутствии априорных ограничений на величину буферных емкостей (складов) этими емкостями нельзя злоупотреблять, создавая слишком большие запасы. Помимо расходов на хранение, возможности потерь от порчи запаса и замораживания оборотных средств нужно еще считаться с тем обстоятельством, что излишние запасы являются тормозом на пути научно-технического прогресса. При переходе на новый вид продукции они ставят предприятие перед дилеммой: либо задержать внедрение новшества до полной выработки сделанных запасов, либо выбросить запасы, не нужные для производства новой продукции, понеся соответствующие материальные потери.

Более подробно встающие здесь вопросы разобраны в следующем параграфе, посвященном управлению запасами. В заключение же настоящего параграфа заметим, что существуют случаи, когда создание буферных запасов невозможно или нежелательно. В наиболее полной форме с подобным обстоятельством приходится встречаться в электроэнергетике. Здесь синхронизация процессов выработки и потребления энергии осуществляется

за счет наличия резерва *мощностей* по выработке электроэнергии и использования этого резерва при увеличениях потребления. При этом в задачу диспетчерского управления входит, как и обычно, оперативное прогнозирование изменения уровня потребления электроэнергии и своевременное подключение и отключение резервных мощностей (в качестве которых удобно использовать агрегаты гидроэлектростанций). При решении задачи управления мощностями учитываются необходимые перетоки электроэнергии по линиям электропередач и связанные с ними потери. Кроме того, принимаются в расчет различная себестоимость электроэнергии, производимой разными агрегатами, график их предполагаемого отключения на ремонты и профилактику, уровни запасов топлива на тепловых электростанциях и запасов воды в водохранилищах (и гидроаккумуляторах) на гидроэлектростанциях. В более изощренных алгоритмах управления учитываются также прогнозы изменения запасов, в частности запасов воды в результате ожидаемых ее поступлений от выпавших дождей, перетоков с вышерасположенных каскадов и т. п. При этом возникают сложные оптимизационные задачи, для решения которых в реальном масштабе времени для крупных энергосистем требуются весьма мощные ЭВМ и многомашинные комплексы.

11.5. Управление запасами и оперативно-календарное планирование

Оперативно-календарное планирование представляет собой следующий (за диспетчерским управлением) уровень организационного управления экономическими объектами. Как и при диспетчировании, главная задача здесь состоит в обеспечении синхронизации работы взаимодействующих участков для надежного функционирования всего объекта в целом. Хорошо известно, что любая задержка в одном из мест цепочки взаимосвязанных участков может привести к лавинообразному нарастанию задержек в других местах и в результате — к большим потерям. Для того чтобы получить высокую степень надежности всего процесса в целом при относительно небольшой надежности отдельных его частей, прибегают к использованию резервов. Резервы могут быть либо в виде незадействованного оборудования, либо в виде запасных частей, при помощи которых осуществляется быстрый ремонт оборудования, либо, наконец, в виде запасов материалов, полуфабрикатов и готовых изделий для снабжения цепи при временных выходах из строя отдельных звеньев.

Установление правильных уровней запасов различных видов, их постоянный контроль, пополнение и использование — центральный вопрос комплекса задач оперативного планирования и

управления. Многие системы оперативно-календарного планирования могут строиться как системы *управления запасами.*

Задачи управления запасами делятся на *статические* и *динамические.* В статических задачах вопрос о создании того или иного запаса выступает как единичный акт; для задач динамических расходование и периодическое пополнение запасов рассматривается как процесс, развертывающийся во времени.

Характерным примером статической задачи является проблема закупки запасных частей вместе с каким-либо уникальным дорогостоящим оборудованием (например, с прокатным станом, турбогенератором большой мощности и т. п.). При изготовлении заказанного оборудования поставщик может изготовить запасные части по относительно низкой цене. Если же закупать их по прошествии определенного времени, когда в них возникает потребность, то их цена может оказаться значительно большей (в силу необходимости заново осуществить подготовку производства у поставщика). Кроме того, могут иметь место значительные потери от простоя оборудования в ожидании изготовления необходимой запасной части (вместо вышедшей из строя). Вопрос состоит в том, какое количество запасных частей покупать вместе с оборудованием.

Чтобы ответить на этот вопрос, предположим, что цена некоторой запасной части в момент поставки оборудования равна 1000 р., убытки же от ее отсутствия в момент аварии равны 20 000 р. (включая закупку ее по более дорогой цене). Пусть, далее, вероятность p_0 того, что за время эксплуатации данная запасная часть не потребуется, равна 0,90; вероятность того, что она потребуется один раз, равна $p_1 = 0{,}08$, а два раза — $p_2 = 0{,}02$. Если не покупать с оборудованием ни одной запасной части, то математическое ожидание убытка равно $S_0 = 0{,}08 \cdot 20\,000 + 0{,}02 \cdot 2 \cdot 20\,000 = 2400$ р. При покупке одного экземпляра $S_1 = 1000 + 0{,}02 \cdot 20\,000 = 1400$ р., а при покупке двух экземпляров $S_2 = 2 \cdot 1000 = 2000$ р. Таким образом, минимальные потери будут во втором случае и, следовательно, правильная политика состоит в закупке запасной части в одном экземпляре.

В качестве второго примера рассмотрим случай разового заказа магазином какого-либо скоропортящегося продукта. Пусть $p(n)$ есть вероятность того, что спрос за время реализации будет не менее чем n единиц; A — прибыль, получаемая при продаже одной единицы продукта, B — убыток на каждую единицу непроданного продукта. Тогда математическое ожидание прибыли от n-й единицы продукта выразится формулой

$$E(n) = Ap(n) - B[1 - p(n)].$$

Легко понять, что закупка n-й единицы будет целесообразной,

если $E(n)>0$. Из неравенства $Ap(n)-B[1-p(n)]>0$ получаем

$$p(n)>\frac{B}{A+B}.$$

Следовательно, правильной политикой будет закупка наибольшего числа n единиц продуктов, при котором еще имеет место это неравенство.

В динамических моделях управления запасами заказы на пополнение запасов даются многократно. Важнейшим здесь является определение *точки заказа*, т. е. момента времени, в которой должен быть произведен заказ, и размеров партии, т.е. количества одновременно заказываемого продукта.

Предположим, что скорость v расходования запаса постоянна (простейший, так называемый линейный случай), а в момент его исчерпания поступает новая партия, пополняющая запас; снова происходит его расходование и т. д. Мы будем называть такую политику заказов политикой *нулевых страховых запасов*. Пусть A — расход на подготовку выполнения заказа у поставщика. В первом приближении можно считать, что эта величина не зависит от размера N заказываемой партии. В таком случае прибавка стоимости на каждую единицу заказываемого продукта выразится величиной A/N. При скорости расходования запаса, равной v, запас величиной N исчерпывается за время $T=N/v$. Среднее время $T_{\text{ср}}$ хранения одной единицы продукта равно

$$\frac{1}{2}T=\frac{N}{2v}.$$

При затратах на хранение одной единицы продукта в течение одной единицы времени, равных B, общие затраты на хранение, отнесенные на единицу продукта, представляются в виде $BN/(2v)$. Общая сумма S дополнительных расходов на единицу заказываемого продукта равна

$$S=\frac{A}{N}+\frac{BN}{2v}. \tag{11.6}$$

При сделанных предположениях (постоянство величин A, B, v) имеем

$$\frac{dS}{dN}=-\frac{A}{N^2}+\frac{B}{2v}.$$

Оптимальный размер партии N_0 (дающий минимум дополнительных расходов) достигается при $dS/dN=0$, откуда

$$N_0=\sqrt{2vA/B}. \tag{11.7}$$

Величина дополнительных расходов S_0, приходящаяся на единицу заказываемого продукта, при оптимальной партии выразится

формулой, получающейся подстановкой значения N_0 в выражение (11.6):

$$S_0 = \sqrt{2AB/v}.$$

Время T_0, через которое должно проходить периодическое обновление запаса (поступление нового заказа), равно N_0/v, т. е.

$$T_0 = \sqrt{2A/(vB)}. \qquad (11.8)$$

Назовем это время периодом *оптимального обновления запаса.* Если скорость v потребления запаса переменна, но меняется относительно медленно, то для расчета объемов оптимальных партий и периодов оптимального обновления можно пользоваться теми же формулами (11.7) и (11.8), принимая в качестве соответствующего значения среднюю скорость $v_{ср}$ расходования запаса в течение времени T_0.

Величины $v_{ср}$ и T_0 находятся последовательным приближением. Пусть $v = ce^{kt}$, где k — достаточно малая величина. Принимая сначала $v_{ср} = v(0) = c$, найдем

$$T_0' = \sqrt{\frac{2A}{cB}}, \quad v_1 = v(T_0) = ve^k \sqrt{2A/(cB)},$$

$$v_{ср}' = v\left(1 + \frac{1}{2} e^{k\sqrt{2A/(cB)}}\right).$$

По этой новой средней скорости найдем новое значение

$$T_0'' = \sqrt{\frac{2A}{v_{ср}' B}}.$$

Если величина $k\sqrt{2A/(cB)}$ мала по сравнению с единицей, то указанный процесс быстро приведет к нахождению удовлетворительных значений для объема оптимальных партий и периода оптимального обновления запаса. Разумеется, указанные значения будут меняться с течением времени.

Малость величины $k\sqrt{2A/(cB)}$ означает, что скорость расходования запаса за период его оптимального пополнения меняется мало. При быстрых изменениях скорости расходования запаса выведенные формулы уже неприменимы и должны быть заменены более сложными приемами, основанными, как и прежде, на минимизации величины добавочных расходов, но уже без предположения о постоянстве величины v.

Может быть, например, применен следующий метод последовательной приближенной оптимизации. Вначале, на основании какой-то средней скорости (выбираемой без особых претензий на точность), находят первое приближение для очередной точки заказа T_0. Затем начинают изменять T_0 в направлении, в котором сумма дополнительных расходов уменьшается до тех пор, пока

она не станет увеличиваться. Иными словами, применяется общий (градиентный) метод оптимизации, описанный в § 8.1. Таким образом, последовательно, одну за другой, определяют очередные точки заказа и соответствующие размеры оптимальных партий.

При использовании подобных общих методов могут применяться более сложные алгоритмы для определения величины дополнительных расходов, связанных с выполнением заказа и хранением заказанного продукта. Например, можно учесть непостоянство расходов на подготовку выполнения заказа, возникающих в том случае, когда при больших размерах заказа нужно вести параллельную подготовку нескольких параллельных технологических линий или менять оснастку в процессе выполнения заказа. Появляется возможность учета нелинейного характера расходов на хранение, если при больших размерах запасов возникает необходимость в оборудовании дополнительных складских помещений. При хранении жидких и газообразных продуктов могут возникнуть ограничения, связанные с наличным объемом буферных емкостей. Такие же ограничения на размер запасов могут быть наложены в результате тех или иных директив вышестоящих органов.

Нелинейность зависимости расходов на хранение от времени хранения имеет место также в том случае, когда при хранении возникают потери. Эти потери могут иметь как физическую природу (например, уменьшение сахаристости свеклы при ее длительном хранении), так и технико-экономическую (например, опасность морального устаревания хранимого продукта в результате возможного изменения технологии, при которой этот продукт сделается ненужным). В стоимость хранения должны быть включены также те дополнительные проценты роста омертвляемых в запасах денежных средств, которые могли бы быть получены при включении этих средств в более активные формы оборота *).

Как было указано выше, при формировании заказов желательно указывать размеры потерь при изменении сроков их выполнения. При досрочном выполнении заказа возникают дополнительные расходы на хранение (включая проценты с омертвленных денежных сумм при досрочной оплате заказов). В простейшем (линейном) случае, описанном выше, подобные расчеты не вызывают особых трудностей: прямые дополнительные расходы на хранение в расчете на одну единицу продукта составят $B\Delta t$, где Δt — время, на которое данный заказ опередил назначенный срок. Дополнительные потери от замораживания денежных

*) Эти потери обычно учитываются лишь при превышении оптимальных размеров запасов, рассчитанных описанными методами.

средств на одну единицу продукта равны $p(1+r)\Delta t/\tau$, где p — стоимость единицы продукта; r — процент роста средств в обороте; τ — период времени, за который начисляется этот процент.

В случае задержки поставки заказа на время Δt при политике нулевого страхового запаса это время представляет собой чистый простой для рассматриваемого объекта. Потери будут исчисляться суммой затрат и амортизационными отчислениями за это время плюс потери от сдвига на соответствующий период заказа, который выполняет данный объект. Размер этих последних потерь предполагается определенным у заказчика в момент формирования заказа.

Политика нулевого страхового запаса приводит к тому, что задержки на самом объекте и в системе его материально-технического снабжения оборачиваются значительными потерями и распространением задержек по всем последующим цепочкам технологических связей. Поэтому на практике стремятся планировать снабжение таким образом, чтобы поступление очередной партии заказанного продукта происходило не в момент полного исчерпания имевшегося запаса, а при сохранении некоторой страховой величины запаса $N_{\text{стр}}$.

В линейном случае (при постоянной скорости v расходования запаса) обычно величину $N_{\text{стр}}$ выбирают постоянной. При этом, как легко проверить, ни размер N_0 оптимальной партии, ни величина T_0 периода оптимального обновления запаса не меняются, т. е. могут по-прежнему вычисляться по формулам (11.7) и (11.8). Дополнительные же расходы увеличатся на величину $BN_{\text{стр}}/v$ и составят

$$S_0 = \sqrt{2AB/v} + BN_{\text{стр}}/v.$$

Средний объем запаса в течение периода обновления T_0 равен

$$\frac{1}{2}(N_{\text{стр}} + N_{\text{стр}} + N_0) = N_{\text{стр}} + \frac{N_0}{2}.$$

Расходы на хранение в расчете на одну единицу заказа равны

$$\frac{B(N_{\text{стр}} + N_0/2)T_0}{N_0} = \frac{B(N_{\text{стр}} + N_0/2)\cdot N_0/v}{N_0} = \frac{BN_{\text{стр}}}{v} + \frac{BN_0}{2v},$$

что лишь постоянным слагаемым $BN_{\text{стр}}/v$ отличается от расходов на хранение в схеме с нулевым страховым запасом.

При расчетах объемов страховых запасов также возможны постановки задач на оптимум. Для этой цели необходимы те или иные гипотезы о вероятностях задержек поставок заказанных партий продуктов и о потерях заказчика от этих задержек. Рассмотрим один из простейших примеров. Предположим, что для каждой заказанной партии возможна задержка, не зависящая от

задержек, имевших место при поставках других заказанных партий, причем вероятность $p(t)$ того, что эта задержка превзойдет время t, выражается экспоненциальной зависимостью $p(t)=e^{-kt}$. Пусть, далее, потери у поставщика за каждую единицу времени простоя постоянны и равны P. Обозначим время $N_{\text{стр}}/v$, на которое хватит страхового запаса при работе с постоянным расходом v, через $t_{\text{ср}}$. Вероятность того, что задержка выполнения заказа лежит в пределах t, $t+dt$, равна $-dp(t)=ke^{-kt}dt$. Если $t>t_{\text{стр}}$, то потери при такой задержке будут равны $P(t-t_{\text{стр}})$. Математическое ожидание этих потерь представится формулой

$$E=\int_{t_{\text{стр}}}^{\infty} P(t-t_{\text{стр}})\,ke^{-kt}dt=\frac{P}{k}e^{-kt_{\text{стр}}}=\frac{P}{k}e^{-kN_{\text{стр}}/v},$$

или, в расчете на единицу продукта в заказываемой партии,

$$E=\frac{P}{kN_0}e^{-kN_{\text{стр}}/v}.$$

Дополнительные же расходы от хранения страхового запаса в расчете на единицу заказываемого продукта равны, как уже было отмечено выше, $BN_{\text{стр}}/v$. Минимизация суммы этих расходов (приравниванием нулю производной по $N_{\text{стр}}$) приводит к определению оптимального размера страхового запаса:

$$N_{\text{стр}}=\frac{v}{k}\ln\left(\frac{P}{BN_0}\right). \qquad (11.8а)$$

Разумеется, размер потерь от простоя объекта в единицу времени должен превышать расходы на хранение заказываемой партии продукта в единицу времени (BN_0), иначе эксплуатация объекта стала бы невыгодной, а для величины страхового запаса по формуле (11.8а) было бы получено отрицательное значение.

При более сложных законах изменения вероятности $p(t)$, характеризующих степень надежности поставщика, равно как и при других отступлениях от принятых выше простейших предположений, задача минимизации дополнительных расходов (точнее, их математических ожиданий) за счет выбора оптимальных размеров страховых запасов усложняется и требует применения общих приемов нахождения экстремумов (см. гл. VIII).

Представляет значительный интерес вопрос о месте, в котором следует иметь страховые запасы. Далеко не всегда выгодно создавать запасы у потребителей. Действительно, нетрудно понять, что в случае, когда число поставщиков гораздо меньше, чем число потребителей (что чаще всего и имеет место), а работа транспорта абсолютно надежна (что бывает далеко не всегда), значительно выгоднее создавать страховые запасы у поставщи-

ков. Разумеется, при этом необходимо обеспечить соответствующие формы ответственности поставщиков за безусловное соблюдение сроков поставок. Если принимать во внимание ненадежность транспорта, то страховые запасы необходимы и у потребителей, однако их размер будет меньшим, поскольку ненадежность одного транспорта меньше суммарной ненадежности транспорта и поставщика.

Рассмотренные примеры иллюстрируют лишь простейшие случаи, возникающие при решении проблем управления запасами. Обобщения и усовершенствования могут идти в различных направлениях. Одно из обобщений возникает в том случае, когда доставка заказа представляет собой не разовый акт, а равномерное поступление продукции в течение некоторого времени t, составляющего часть периода T_0 оптимального обновления запаса.

$$t = rT_0, \quad 0 \leqslant r \leqslant 1.$$

При этом скорость пополнения запаса и скорость его расходования связаны соотношением $v = ru$. Случай $r = 0$ соответствует рассмотренному выше случаю разовой поставки всего запаса. Размер оптимальных партий, величина дополнительных расходов (на единицу продукта) и период оптимального обновления запаса выразятся в общем случае формулами

$$N_0 = \sqrt{\frac{2vA}{B(1-r)}}, \quad S_0 = \sqrt{\frac{2AB(1-r)}{v}}, \quad T_0 = \sqrt{\frac{2A}{Bv(1-r)}}.$$

Другие обобщения связаны с введением случайных колебаний в скорость v расходования запаса, что приводит к необходимости создания страховых запасов.

Большой интерес представляет собой многопродуктовая модель, когда запасы создаются по нескольким различным видам продуктов. В этом случае помимо задачи определения оптимальных размеров заказов по различным продуктам возникает задача о взаимном расположении точек заказа. Дело в том, что одновременный заказ различной продукции может привести к необходимости иметь значительный запас денежных средств для оплаты всех этих заказов. Более равномерное распределение точек заказа во времени при сохранении средней суммы оборотных средств, вложенных в запасы, приводит к уменьшению максимальных стоимостей всех имеющихся на данный момент запасов. Иными словами, в многопродуктовой модели обычно стремятся к более равномерному использованию денежных средств, предназначенных для оплаты заказов. Решение этой задачи может быть получено общими методами поиска экстремума для оценочной функции $S(t_1, t_2, \ldots, t_k)$, где переменные $t_1, t_2, \ldots, t_k$ представляют собой искомые точки заказов соответствующих продуктов. Функ-

ция задается алгоритмом, описывающим скорости пополнения и расходования запасов.

В реальных системах управления запасами учитывается также задержка между временем заказа и временем получения заказанного продукта. Если это время обозначить через τ, размер страхового запаса — через $N_{стр}$, постоянную скорость расходования запаса — через v, то точка заказа определяется моментом, когда уровень запаса S станет равным

$$S = N_{стр} + v\tau.$$

Этой формулой обычно и пользуются в АСУ, ведущих непрерывный учет состояния запасов для определения точек заказов. Если учет запасов ведется с интервалом $t_{уч}$, то размер запаса, определяющий точку заказа, вычисляется по формуле

$$S = N_{стр} + v\left(\tau + \frac{1}{2} t_{уч}\right).$$

В противоположность описанной системе, в которой определяется оптимальный размер заказа и которую при $v = \text{const}$ можно назвать системой с *постоянным размером заказа*, иногда употребляются и другие системы управления запасами. Хотя они и не минимизируют дополнительных расходов, но могут быть удобными в других отношениях, прежде всего для упрощения вычислений.

Так, в системе с *постоянным уровнем запасов* фиксируется некоторый максимальный уровень запасов и через равные промежутки времени проверяется уровень запасов. В результате проверки формируется заказ размером

$$N = S_{max} - S_{тек} - S_{непол},$$

где S_{max} — принятый максимальный объем запаса; $S_{тек}$ — фактический запас на момент проверки; $S_{непол}$ — сделанные, но еще не полученные заказы (при $\tau > t_{уч}$). Максимальный уровень запаса определяется по формуле

$$S_{max} = N_{стр} + v(\tau + t_{уч}).$$

При $v = \text{const}$ средний уровень запасов равен

$$S_{сред} = N_{стр} + \frac{1}{2} v t_{уч}.$$

Иногда применяется система с *двумя уровнями*. В ней, как и в предыдущей системе, размер заказа определяется формулой

$$N = S_{max} - S_{тек} - S_{непол}.$$

Однако решение о производстве заказа принимается не при каждой

проверке (через время $t_{уч} = \text{const}$), а лишь в том случае, когда

$$S_{тек} + S_{непол} < S_{зак} = N_{стр} + v(\tau + t_{уч}/2),$$

где $S_{зак} < S_{max}$ и представляет как раз второй уровень запаса, по которому производится регулирование и который дал название данной системе.

При переменной скорости v расходования запаса вместо членов $v(\tau + t_{уч})$, $v(\tau + t_{уч}/2)$ применяются прогнозные оценки для величины расхода за соответствующие промежутки времени.

Имеются вариации методов управления запасами, использующие несколько точек заказа. Например, кроме максимального размера запаса S_{max} может существовать не одна точка заказа, а две: S_1 и S_2 $(S_1 > S_2)$. При снижении уровня запаса S ниже S_1 (но выше S_2) при очередной проверке дается заказ размером $N = S_{max} - S_{тех} - S_{непол}$, а при $S < S_2$ размер заказа определяется без учета неполученных заказов: $N = S_{max} - S_{тек}$. Такой способ применяется в том случае, когда между очередными проверками возможны быстрые изменения скорости расхода запаса, так что ожидание неполученных заказов может привести к возникновению дефицита.

Применение теории управления запасами к оперативно-календарному планированию основано на расчленении технологических маршрутов на отдельные участки, выступающие по отношению друг к другу как поставщики и потребители. Для возможности отделения одного участка от другого необходимо наличие буферных емкостей для хранения запасов продуктов, выпускаемых одним участком и потребляемых на другом участке. При отсутствии таких буферов отдельные участки соединяются в один участок, называемый *поточной линией*, на котором производительности отдельных составляющих его участков должны быть строго согласованы между собой. Примером поточной линии является обычный конвейер, хотя, разумеется, поточные линии могут строиться и без него.

Для каждого участка формируется поток заказов. В заключительных участках (например, сборочных цехах) такие задания формируются из заданий для всего рассматриваемого экономического объекта для других участков — из заказов участков, расположенных впереди по соответствующим технологическим маршрутам. Результатом расчета календарного плана должны быть сменные задания на выполнение необходимых производственных, подготовительных и ремонтных операций, а также формирование и привязка к точным календарным срокам заказов другим участкам и заданий на материально-техническое снабжение всего объекта.

Для иллюстрации задач, возникающих при оперативно-календарном планировании, рассмотрим пример. Рассчитаем календар-

ный план работы производственного участка, работающего в две смены по 8 ч с двумя выходными днями в неделю (на июнь); выходные дни приходятся на числа 1, 7, 8, 14, 15, 21, 22, 28, 29. Участок выпускает два вида деталей: I и II. На изготовление детали типа I тратится 10 мин, а на изготовление детали типа II — 50 мин. Подготовительные операции (переналадка оборудования) для выпуска партии деталей типа I занимают 1 ч, а деталей типа II — 2 ч.

Производственное задание участку формируется в следующем виде: 9 июня к началу первой смены поставить партию из 400 шт. деталей типа I, 12 июня к началу второй смены — 60 шт. деталей типа II, 18 июня к началу второй смены — 400 шт. деталей типа I, 26 июня к началу второй смены 120 шт. деталей типа II и 1 июля к началу первой смены — 200 шт. деталей типа I. Заданные сроки поставок будем называть *контрольными сроками*. Предполагается, что описанное задание сформировано в результате предварительного решения двух типов оптимизационных задач. Во-первых, при объемно-календарных расчетах была произведена оптимизация распределения месячного задания между отдельными участками. Во-вторых, при определении точек заказа и размеров партий использовались описанные выше методы теории управления запасами. К сожалению, на практике при согласовании работы большого числа участков не удается полностью выдержать оптимальные размеры партий, удается лишь приближаться к ним, как к желательному ориентиру. Это обстоятельство нашло место и в приведенном выше задании.

Отправным пунктом календарных расчетов является общее количество ресурсо-часов (или ресурсо-минут), которыми мы располагаем до моментов, когда необходимо осуществлять поставку заказанных партий продукции. В соответствии с приведенным заданием имеются пять таких моментов (9, 12, 18, 26 июня и 1 июля). Поскольку мы не расчленяем участок на более мелкие составные части*), то все его ресурсы могут рассматриваться как единый ресурс и исчисляться просто в минутах работы всего участка. Нетрудно подсчитать, что до первого момента располагаем 10 сменами, или 4800 мин, между первым и вторым моментами — 7 сменами (3360 мин) и т. д. Все данные расчета приведены в табл. 11.1.

Определяем общий резерв времени, т. е. разность между имеющимся и необходимым для выполнения всех заданий временем: $20\,160 - 19\,000 = 1160$ мин $= 19$ ч 20 мин. Допустим, что в предстоящем месяце необходимо осуществить плановый текущий ремонт оборудования, что должно занять по нормам 12 ч. Остальные

*) Весь участок может сводиться к одному станку и одному рабочему месту.

ное время (7 ч 20 мин) может быть использовано на переналадки и создание оперативного резерва рассматриваемого участка. Пусть к началу планового периода (т. е. 2 июня) участок был настроен на выпуск детали I. Тогда для подготовки выпуска всех пяти заказанных партий потребуются четыре переналадки общей продолжительностью $2+1+2+1=6$ ч. Оперативный резерв времени составит лишь 1 ч 20 мин, что означает очень плотную загрузку участка.

Таблица 11.1

Контрольные сроки	Тип деталей	Размер партии, шт.	Время для выполнения задания, мин	
			необходимое	имеющееся
9 июня начало 1-й смены	I	400	4 000	4 800
12 июня начало 2-й смены	II	60	3 000	3 360
18 июня начало 2-й смены	I	400	4 000	3 840
26 июня начало 2-й смены	II	120	6 000	5 760
1 июля начало 1-й смены	I	200	2 000	2 400
Итого			19 000	20 160

В практике планирования может встретиться и такой случай, когда этот резерв отрицателен. Тогда уменьшают общее число переналадок, объединяя несколько партий, или плановые ремонты осуществляют не в рабочие смены и т. п. Разумеется, объединение заказов означает отступление от оптимальных размеров партий и, как следствие этого, дополнительные финансовые потери. Может возникнуть необходимость и в сдвиге сроков поставок, что, как уже отмечалось выше, должно производиться лишь на основании расчета объемов потерь, возникающих при этом у заказчика.

При наличии резервов времени возникает вопрос об их распределении между различными подынтервалами планового периода. Для этой цели составляется баланс времени (и ресурсов) на весь плановый период нарастающим итогом с учетом необходимых переналадок (табл. 11.2).

Поскольку необходимый суммарный расход времени на 26 июня лишь на 360 мин (6 ч) меньше наличного запаса времени, то 12-часовой плановый ремонт может быть отнесен лишь на последний интервал (между 26 июня и 1 июля). Остающийся резерв времени (1 ч 20 мин) настолько мал, что вопрос о его распределении по плановому периоду не имеет существенного значения. При большей величине этого резерва обычно стремятся распределить его равномерно вдоль всего планового периода, что-

бы обеспечить равномерную загрузку участка и возможность маневрирования при возникновении непредвиденных обстоятельств. Наличие резервов позволяет производить загрузку оборудования в соответствии с экономическими критериями, сохранять оптимальные размеры партий и минимально необходимые размеры запасов.

Таблица 11.2

Контрольные сроки	Номер партии	Время переналадки, мин	Рабочее время, мин	Время от начала планового периода, мин	
				расход	запас
9 июня	1	0	4000	4 000	4 800
12 июня	2	120	3000	7 120	8 160
18 июня	3	60	4000	11 180	12 000
26 июня	5	120	6000	17 300	17 760
1 июля	5	60	2000	19 360	20 160

Когда имеется значительная вероятность появления новых заданий, более целесообразно придерживаться политики сохранения резервов до момента появления этих новых заданий, с тем чтобы облегчить возможность быстрой коррекции плана.

В рассматриваемом случае естественно принять стратегию, основанную на заданной последовательности заданий, поскольку сделанная в табл. 11.2 раскладка времени ни к одному из контрольных сроков не обнаружила дефицита. Плановый ремонт, как уже было отмечено выше, относится на конец периода. Сменные задания можно планировать непосредственно с учетом данных табл. 11.2:

8 смен 2, 3, 4, 5 июня: выпуск 384 деталей типа I (по 48 деталей в смену);

1-я смена 6 июня: $0—2^{40}$ — выпуск 16 деталей типа I (поставка 400 деталей типа I), $2^{40}—4^{40}$ — переналадка, $4^{40}—8^{00}$ — выпуск 4 деталей типа II;

2-я смена 6 июня

4 смены 9 и 10 июня: выпуск 48 деталей типа II;

1-я смена 11 июня: $0—6^{40}$ — выпуск 8 деталей типа II (поставка 60 деталей типа II досрочно), $6^{40}—7^{40}$ — переналадка, $7^{40}—8^{00}$ резерв;

2-я смена 11 июня: выпуск 48 деталей типа I;

6 смен 12, 13, 16 июня: выпуск 288 деталей типа I;

1-я смена 16 июня: выпуск 48 деталей типа I;

2-я смена 16 июня: $0—2^{40}$ — выпуск 16 деталей типа I (поставка 400 деталей типа I досрочно), $2^{40}—4^{40}$ — переналадка, $4^{40}—8^{00}$ — выпуск 4 деталей типа II;

10 смен 17, 18, 19, 20, 23 июня: выпуск 96 деталей типа II;
2 смены 24 июня: выпуск 19 деталей типа II, 10 мин — резерв;
1-я смена 25 июня: $0—0^{50}$ — выпуск одной детали типа II (поставка 120 деталей типа II досрочно), $0^{50}—8^{00}$ — ремонт;
2-я смена 26 июня: $0—4^{50}$ — ремонт, $4^{50}—5^{50}$ — переналадка, $5^{50}—8^{00}$ — выпуск 13 деталей типа I;
2 смены 27 июня и одна смена 30 июня: выпуск 144 деталей типа I;
2-я смена 30 июня: $0—7^{10}$ — выпуск 43 деталей типа I (поставка деталей типа I), $7^{10}—8^{00}$ — резерв.

После составления сменных заданий проводится расчет материально-технического обеспечения участка. Описанными выше методами определяются объемы страховых запасов, размеры партий и точки заказа для заготовок, материалов, инструмента и т. п. Аналогичным образом заказываются вагоны и другие транспортные средства для вывоза готовой продукции. Определяется ожидаемое время поступления платежей за поставленную продукцию и на каждый день принятого планового периода рассчитывается величина сумм по различным статьям, которая будет находиться на текущем счете предприятия.

Составленный план непрерывно пополняется и уточняется. Для этой цели он постоянно держится на машинных носителях (лучше всего — на дисках) в состоянии готовности. Для облегчения работы с планом создается специальное математическое обеспечение (специализированная операционная система), позволяющее пользователю быстро вносить необходимые изменения, сохраняя полную сбалансированность плана. Например, при изменении срока выполнения задания должны автоматически изменяться сроки поставок материалов, заготовок инструмента и т. п. В систему должны быть включены специальные программы-макрооператоры, позволяющие быстро подсчитывать различные временны́е и технико-экономические характеристики плана.

Иными словами, нужно иметь возможность непрерывной работы с планом и принятия решений по его улучшению на основе машинного моделирования возникающих ситуаций и предлагаемых путей устранения возникающих трудностей. Динамичность плана должна проявляться, в частности, в том, что в нем постоянно корректируются запасы времени до соответствующих контрольных точек. Например, в табл. 11.1 на начало рабочего дня 9 июня в первой строке последнего столбца указано 4800 рабочих минут до первой контрольной точки. На начало следующего рабочего дня эта цифра должна быть заменена на 3840 мин и т. д.

Существенную роль играет связь системы планово-календарных расчетов с другими системами. Например, если размер стра-

ховых запасов по заготовкам деталей типа I был определен в 100 шт., а из системы технической подготовки производства поступает информация о том, что к концу месяца ожидается изменение технологии, устраняющей необходимость использования детали типа I, то система должна обеспечить постепенную выработку накопленного страхового запаса к моменту выхода приказа об изменении технологии.

Описанные задачи и приемы их решения являются общими для всех производств дискретного характера. Особенность производств, имеющих дело со сложными изделиями с длительными циклами изготовления (корабли, прокатные станы и т. п.), состоит в том, что основу оперативно-календарного планирования составляют сетевые графики (хотя и здесь управление заказами на сравнительно большие партии деталей может строиться на описанных принципах).

Свои особенности имеет также массовое производство (производство холодильников, телевизоров и т. п.). Одна из важных задач, которую приходится решать в этом случае,— задача нахождения оптимального ритма производства. Смысл ее состоит в том, чтобы через промежуток времени T, называемый *циклом* или *ритмом* работы объекта, повторялись все сменные задания, которые имели место на предшествующем цикле. Обычно стремятся найти минимальный цикл $T_{\min}$, выражаемый целым числом смен или дней.

Для примера рассмотрим работу заготовительного участка, обеспечивающего сборочный цех деталями n различных типов. Пусть p_i — время, затрачиваемое на изготовление одной детали i-го типа; τ_i — время на подготовку выпуска деталей i-го типа; S_i — скорость расходования этих деталей на сборке ($i = 1, 2, \ldots, n$). Чтобы обеспечить сборку, заготовительный участок за время T должен произвести S_iT деталей i-го типа. Время, необходимое для изготовления S_iT изделий, равно $\tau_i + S_iT/p_i$, а суммарное время не должно превосходить T:

$$T \geqslant \sum_{i=1}^{n} \left(\tau_i + \frac{S_i T}{p_i} \right),$$

откуда

$$T \geqslant \sum_{i=1}^{n} \tau_i \left(1 - \sum_{i=1}^{n} \frac{S_i}{p_i} \right)^{-1}. \qquad (11.9)$$

Минимальное (целое) значение T, удовлетворяющее неравенству (11.9), и представляет собой искомый минимальный ритм работы рассматриваемого объекта.

При $\tau_1 = \tau_2 = \tau_3 = 0{,}5$, $S_1 = 10$, $S_2 = 15$, $S_3 = 20$, $p_1 = 40$, $p_2 = 30$, $p_3 = 100$

$$T \geqslant \frac{1{,}5}{1 - (1/4 + 1/2 + 2/10)} = \frac{1{,}5}{1 - 0{,}95} = 30.$$

Это означает, что возможен минимальный ритм работы с периодом в 30 смен.

11.6. Программно-целевое управление

Программно-целевым управлением называется разновидность организационного управления, направленная на достижение относительно небольшого числа точно очерченных целей. Первый этап программно-целевого управления — *составление программы*. Суть его состоит в том, что помимо первоначально заданных *конечных* целей программы формулируются *промежуточные* (вспомогательные) цели (называемые также *подцелями*), которых необходимо достичь прежде, чем будут достигнуты основные конечные цели. Очень важно, чтобы при этом были по возможности выделены и о б ъ е д и н е н ы все промежуточные подцели, необходимые для достижения нескольких целей (основных или промежуточных). После подобной *структуризации* системы целей описываются *работы*, которые необходимо выполнить для их достижения. Производится *оценка времени* и *ресурсов* для выполнения описанных работ (трудовых, материальных и финансовых).

На этом этапе отдельные работы еще не привязаны ни к каким конкретным срокам. Математическим аппаратом, с помощью которого описываются составленные программы, являются *сетевые графики*, описываемые ниже (§ 11.7). Специальными методами определяются минимальное время и суммарные ресурсы, необходимые для выполнения программы. Если эти величины не удовлетворяют *руководителя программы*, производится дальнейшая работа по улучшению программы (в большинстве своем неформализуемая). Окончательные (улучшенные) варианты программ вводятся в ЭВМ.

С помощью специального программного обеспечения составляющие программы работы привязываются к конкретным временны́м интервалам. В результате такой привязки, которую мы будем также называть проекцией, появляется возможность рассчитать, *временны́е графики* расходов (помесячные, поквартальные и т. п.) всех видов ресурсов, необходимых для выполнения программ. Эти графики сравниваются с наличными запасами ресурсов на каждый интервал времени. В случае превышения требований к ресурсам по сравнению с их наличием производится изменение проекций *отдельных работ* (обычно в человеко-машинном режиме), составляющих программы на конкретные временны́е интервалы. Применяя методы системной (многокритериаль-

ной) оптимизации, описанные в § 8.12, осуществляют направленное управление как проекциями работ, так и ограничениями, пока не получат *реальный* (обеспеченный ресурсами) план выполнения всех программ с соблюдением окончательно формируемых в процессе системной оптимизации ограничений по времени, ресурсам, равномерности использования ресурсов (уменьшающей их потери) и по любым другим возможным критериям, характеризующим качество составленного *плана реализации программ.*

При управлении выполнением программ в соответствии с составленными планами сохраняется динамическое представление этих планов в памяти ЭВМ. Это означает, что в ЭВМ хранится не только составленный *календарный план* работ, но и программы в их первоначальном виде (не привязанном ко времени) вместе с соответствующими отображениями программ на планы. В случае изменения условий в процессе выполнения планов процесс системной оптимизации может быть продолжен (с соответствующей коррекцией планов). При этом динамическое представление планов обеспечивает при любых их коррекциях сохранение увязки всех целей и работ, предусмотренной в исходных программах, т. е. *целостность программ.*

Обеспечение целостности программ представляет собой фактически ту же самую синхронизацию работ отдельных участков, которая уже была разобрана выше (в §§ 11.4, 11.5), но уже применительно не к массовому и крупносерийному, а к единичному (и, возможно, к мелкосерийному) производству, а также к строительству, реконструкциям, проектно-конструкторским, научно-исследовательским работам и т. п.

Как и раньше, для обеспечения надежного выполнения планов, в случае программно-целевого управления, в руках руководителей программ должны находиться определенные резервы, которые могут использоваться для компенсации тех или иных отклонений в процессе *оперативного управления* выполнением программ. При значительных отклонениях, а также при возникновении дополнительных возможностей по улучшению планов может производиться уже упоминавшаяся выше коррекция планов на основе продолжения процесса системной оптимизации.

Заметим, что описанная здесь общая схема программно-целевого планирования и управления несколько отличается от классической американской системы ПЕРТ, в которой этап планирования предшествует этапу программирования (система ППБ: планирование, программирование, бюджет). Впрочем, в значительной мере это различие носит не принципиальный, а чисто терминологический характер *).

*) Под планированием ПЕРТ понимается не планирование в принятом у нас смысле слова, а лишь предварительное располагание.

Второе замечание связано с тем, что для долгосрочных проектов большой сложности, в первую очередь для проектов создания и внедрения принципиально новых образцов техники и технологии, этапу программирования должен предшествовать этап *научно-технического прогнозирования*. С одним из методов такого прогнозирования (принадлежащим автору), особенно удобным для программно-целевого планирования, мы познакомимся ниже (в § 11.8).

Еще одно замечание связано с тем, что для удобства сопряжения систем программно-целевого и обычного (объемно-календарного) планирования и управления вводится четкое разделение ресурспой части управления на собственно ресурсную часть, в которую включаются рабочая сила и основные фонды (оборудование), и *материально-техническое снабжение*, в которое включаются все поставки материалов и комплектующих изделий из систем, внешних по отношению к рассматриваемым программно-целевым комплексам. В программное обеспечение автоматизированных систем программно-целевого управления вводятся средства быстрого вычисления временны́х графиков поставок материально-технического снабжения после осуществления любой проекции программы на временну́ю ось.

В математическое обеспечение автоматизированных систем программно-целевого управления вводятся специальные средства, позволяющие объединять цели и работы (обычно в диалоговом человеко-машинном режиме), осуществляя тем самым укрупнение форм представления программ (сетевых графиков). Поскольку исходные представления программ при этом сохраняются в памяти ЭВМ (на соответствующих уровнях системы управления), то в случае необходимости можно автоматически осуществить разукрупнение (дезагрегацию) любого участка программы или всей программы целиком.

11.7. Сетевые графики

Сетевые графики представляют собой математический аппарат для представления, изучения и управления сложными комплексами взаимосвязанных работ, направленных к достижению относительно небольшого числа четко определенных целей. Наиболее простыми являются *одноцелевые* сетевые графики, с которых мы и начнем наше рассмотрение. Целью здесь может являться окончание строительства того или иного объекта, создание уникального изделия, выполнение научно-исследовательской или опытно-конструкторской работы, завершение реконструкции или ремонта и т. п.

Простейший пример одноцелевого сетевого графика изображен на рис. 11.7. Сетевой график состоит из объектов двух ви-

дов: узлов, обозначенных кружками, и соединяющих их направленных ребер, обозначенных стрелками. Каждому узлу соответствует некоторое событие, заключающееся в окончании того или иного этапа работ, например «закладка фундамента окончена», «технический проект принят комиссией» и т. д. Каждой стрелке соответствует та или иная работа, понимаемая как процесс, а не как результат, например процесс сооружения стен, процесс оформления эскизного проекта и т. д. Для каждой работы задается ее продолжительность, измеряемая в единицах, фиксированных для данного графика (часы, дни, недели, месяцы).

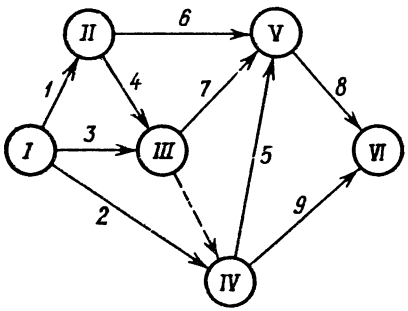

Рис. 11.7

Смысл графика состоит прежде всего в том, чтобы указать все технологические связи, определяющие возможные последовательности работ. Из рис. 11.7 видно, что работа 6 может быть начата лишь после окончания работы 1, а работа 7 — лишь после окончания работ 3 и 4. В ряде случаев для задания связей приходится пользоваться так называемыми *фиктивными работами*, имеющими нулевую продолжительность и обозначаемыми на графике пунктирными стрелками. Роль фиктивных работ легко уяснить из рис. 11.7. Работа 7 может выполняться после окончания работ 3 и 4, а работы 5 и 9 — после окончания работ 2, 3 и 4. Если бы мы объединили узлы III и IV, то работе 7 пришлось бы ждать окончания не только работ 3 и 4, но и работы 2. При наличии фиктивной работы необходимость такого ожидания исключается.

На любом одноцелевом сетевом графике выделяются два события — начальное и конечное. *Начальное событие* (событие I) характеризуется тем, что в него не входит ни одна стрелка, а *конечное* (событие VI) — тем, что из него не выходит ни одна стрелка. Все остальные события носят название *промежуточных*. Начальное событие соответствует началу работ (нулевой момент времени), конечное — завершению всех работ (достижению поставленной цели).

Первой задачей, возникающей при анализе сетевых графиков, является определение времени наступления каждого события и возможностей варьирования времени начала и окончания каждой работы. Алгоритм решения этой задачи весьма прост. Предположим, что для графика, изображенного на рис. 11.7, продолжительность работ задается $t_1 = 3$; $t_2 = 7$; $t_3 = 5$; $t_4 = 3$; $t_5 = 5$; $t_6 = 6$; $t_7 = 3$; $t_8 = 2$; $t_9 = 6$.

Событию I соответствует момент времени $T_1 = 0$. Чтобы найти минимальный срок T_i наступления любого другого события ($i \geqslant 2$), необходимо, очевидно, просчитать суммарные затраты времени по всем путям, ведущим из начального узла в узел, соответствующий этому событию, и выбрать из них максимальное значение. Всякий раз, когда для какого-то события срок его наступления установлен, можно строить дальнейшие пути, отправляясь от этого события как от начального. Для события II, к которому ведет только один путь (работа 1), получим $T_2 = T_1 + t_1 = 0 + 3 = 3$.

Для события III имеем два пути. Учитывая лишь работу 3, получим первое (не окончательное) значение для времени t_3 наступления этого события: $T_3^{(3)} = T_1 + t_3 = 0 + 5 = 5$. Соответствующее вычисление по работе 4 приводит к значению $T_3^{(4)} = T_2 + t_4 = 3 + 3 = 6$. Наибольшее из этих двух значений и дает искомый срок наступления события III: $T_3 = 6$.

Для события IV $T_4^{(2)} = T_1 + t_2 = 0 + 7 = 7$; $T_4^{(\text{фикт})} = T_3 + 0 = 6 + 0 = 6$. Отсюда $T_4 = 7$. Для события V $T_5^{(6)} = T_2 + t_6 = 3 + 6 = 9$; $T_5^{(7)} = T_3 + t_7 = 6 + 3 = 9$; $T_5^{(5)} = T_4 + t_5 = 7 + 5 = 12$. Таким образом, $T_5 = 12$. Наконец, $T_6^{(8)} = T_5 + t_8 = 12 + 2 = 14$; $T_6^{(9)} = T_4 + t_9 = 7 + 6 = 13$, откуда $T_6 = 14$.

Получив минимальный срок T_6 наступления конечного события, а значит, и время окончания всех работ, мы можем, отправляясь от него и выполняя описанную процедуру в обратном порядке, вычислить максимально возможные сроки $\overline{T}_i$ наступления всех событий графика, исходя из необходимости закончить все работы в вычисленный срок. Для конечного события полагаем $\overline{T}_6 = T_6 = 14$. Для события V имеем $\overline{T}_5 = T_6 - t_8 = 14 - 2 = 12$. Для события IV имеем две возможности (двигаясь от событий V и VI в направлении, обратном стрелкам): $\overline{T}_4^{(5)} = \overline{T}_5 - t_5 = 7$; $\overline{T}_4^{(9)} = \overline{T}_6 - t_9 = 14 - 6 = 8$. Из полученных чисел необходимо, очевидно, выбрать минимальное, т. е. $\overline{T}_4 = 7$. Для события III получаем $\overline{T}_3^{(7)} = \overline{T}_5 - t_7 = 12 - 3 = 9$; $\overline{T}_3^{(\text{фикт})} = 7$. Таким образом, $\overline{T}_3 = 7$. Для события II $\overline{T}_2^{(6)} = \overline{T}_5 - t_6 = 12 - 6 = 6$; $T_2^{(4)} = \overline{T}_3 - t_4 = 7 - 3 = 4$, откуда $T_2 = 4$.

При правильности вычисления время наступления начального события всегда должно быть равно 0. Проверяем: $\overline{T}_1^{(1)} = \overline{T}_2 - t_1 = 1$; $\overline{T}_1^{(2)} = \overline{T}_4 - t_2 = 7 - 7 = 0$; $\overline{T}_1^{(3)} = \overline{T}_3 - t_3 = 2$. Минимальное значение равно 0. Таким образом, $\overline{T}_1 = 0$, что подтверждает (хотя и не с абсолютной гарантией) правильность проведенных нами вычислений.

Сведем полученные значения в табл. 11.3. Из этой таблицы видно, что для событий II и III допустимо опоздание на едини-

цу времени без изменения общего срока выполнения работ. Для событий I, IV, V, VI никакие опоздания недопустимы: любая задержка в их наступлении влечет за собой увеличение общего срока всех работ. Про события, обладающие таким свойством, принято говорить, что они находятся на *критическом пути*.

Таблица 11.3

Сроки	События					
	I	II	III	IV	V	VI
$\underline{T}_i$	0	3	6	7	12	14
$\overline{T}_i$	0	4	7	7	12	14

Чтобы найти критический путь для каждой j-й работы, вычислим минимально и максимально возможные сроки начала и окончания работы:

$$T^{\min}_{j\text{нач}} = T_p, \qquad T^{\min}_{j\text{кон}} = T_p + t_j,$$
$$T^{\max}_{j\text{нач}} = \overline{T}_q - t_j, \quad T^{\max}_{j\text{кон}} = \overline{T}_q,$$

где p — событие, стоящее в начале работы j; q — в ее конце. Результаты расчетов сведены в табл. 11.4.

Таблица 11.4

Сроки	Виды работ								
	1	2	3	4	5	6	7	8	9
$T_{j\,\text{нач}}^{\min}$	0	0	0	3	7	3	6	12	7
$T_{j\,\text{нач}}^{\max}$	1	0	2	4	7	6	9	12	8
$T_{j\,\text{кон}}^{\min}$	3	7	5	6	12	9	9	14	13
$T_{j\,\text{кон}}^{\max}$	4	7	7	7	12	12	12	14	14
t_j	3	7	5	3	5	6	3	2	6

Разность $\tau_j = T^{\max}_{j\text{нач}} - T^{\min}_{j\text{нач}} = T^{\max}_{j\text{кон}} - T^{\min}_{j\text{кон}} = \overline{T}_q - T_p - t_j$ будем называть *резервом времени* для j-й работы. Работы, у которых резерв времени равен нулю, составляют критический путь. В нашем случае критический путь составляют работы 2, 5 и 8. Остальные работы имеют меньшие или большие ненулевые резервы времени. Разумеется, при использовании сетевого графика в качестве инструмента управления, а не только как средства для начального анализа, необходимо осуществлять регулярное обновление информации (как об ожидаемой продолжительности работ, так и о фактическом их состоянии) и каждый раз заново

вычислять все показатели. При этом может меняться не только ожидаемое время наступления событий (в том числе и конечного), но и критический путь. Основное преимущество управления на основе сетевого графика состоит в том, что внимание руководства может сосредоточиваться на тех работах, которые являются решающими с точки зрения сроков окончания всех работ.

На этапе предварительного планирования определение критического пути сопровождается анализом возможностей сокращения продолжительностей работ, находящихся на этом пути. Учет этих возможностей с соответствующим пересчетом всего графика определяет новый критический путь, для которого вновь ищутся возможности сокращения сроков, и так далее, пока не будут исчерпаны все возможности дальнейшего улучшения графика. Эта работа должна продолжаться постоянно и в процессе фактического выполнения работ, поскольку здесь могут открываться новые возможности улучшения графика, недоступные на этапе предварительного планирования, либо могут возникать новые задачи по улучшению графика в связи с изменениями критического пути.

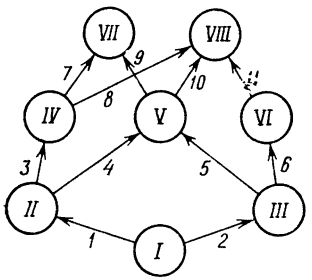

Рис. 11.8

При переходе от одноцелевых сетевых графиков к многоцелевым описанная выше методика временно́го анализа по существу не меняется. Разница состоит лишь в том, что максимально возможные сроки $\overline{T}_i$ наступления событий должны вычисляться для каждой из поставленных целей. Самый поздний из этих сроков определяет время решения задачи, т. е. достижение всех конечных целей.

Рассмотрим двухцелевой сетевой график, изображенный на рис. 11.8, для которого продолжительность работ задана табл. 11.5.

Таблица 11.5

Виды работ	1	2	3	4	5	6	7	8	9	10	11
Срок (t_i)	4	3	4	2	3	4	4	6	5	2	4

Действуя точно так же, как и в предыдущем примере, определим минимальные сроки наступления всех событий (табл. 11.6). Для определения максимально возможных сроков $\overline{T}_i$ наступления событий полагаем $\overline{T}_7 = T_7 = 12$; $\overline{T}_8 = T_8 = 14$. Обозначим через $\overline{T}_i^{(\mathrm{VII})}$ и $\overline{T}_i^{(\mathrm{VIII})}$ максимально возможные сроки наступления собы-

тия i с точки зрения задач достижения отдельно взятых целей VII и VIII соответственно. Тогда $\overline{T}_6^{(VII)} = \infty$, поскольку достижение цели VII не требует наступления события VI. С другой стороны, $\overline{T}_6^{(VIII)} = \overline{T}_8 - t_{11} = 10$. Максимально возможный срок наступления события VI с точки зрения достижения обеих целей равен $\overline{T}_6 = \min(\infty, 10) = 10$.

Таблица 11.6

События	I	II	III	IV	V	VI	VII	VIII
Срок ($\overline{T}_i$)	0	4	3	8	6	7	12	14

Для события V получаем $\overline{T}_5^{(VII)} = \overline{T}_7 - t_9 = 12 - 5 = 7$; $\overline{T}_5^{(VIII)} = \overline{T}_8 - t_{10} = 12$. Тогда $T_5 = \min(7, 12) = 7$. Продолжая таким же образом, находим сроки $\overline{T}_i$ для всех событий (табл. 11.7).

Таблица 11.7

События	I	II	III	IV	V	VI	VII	VIII
Срок ($\overline{T}_i$)	0	4	4	8	7	10	12	14

Для вычисления минимально и максимально возможных сроков начала и окончания работ можно пользоваться теми же формулами, что и раньше, если речь идет о достижении обеих целей. Заменяя в формулах $\overline{T}_i$ на $\overline{T}_i^{(VII)}$ и $\overline{T}_i^{(VIII)}$, находим соответствующие сроки с точки зрения задач достижения каждой из целей VII и VIII в отдельности. Например,

$$T_{4нач}^{min} = T_2 = 4;\ T_{4нач}^{max} = \overline{T}_5 - t_4 = 5;$$
$$T_{4нач}^{(VII)max} = T_5^{(VII)} - t_4 = 7 - 2 = 5;$$
$$T_{4нач}^{(VIII)max} = \overline{T}_5^{(VIII)} - t_4 = 12 - 2 = 10.$$

Соответственно,

$$T_{4кон}^{min} = T_2 + t_4 = 3 + 2 = 5;\ T_{4кон}^{max} = \overline{T}_5 = 7;$$
$$T_{4кон}^{(VII)max} = \overline{T}_5^{(VII)} = 7;\ T_{4кон}^{(VII)max} = \overline{T}_5^{(VIII)} = 12.$$

Аналогичным образом вычисляем сроки для всех остальных работ.

До сих пор при рассмотрении сетевых графиков мы имели дело с проблемами, касающимися лишь сроков выполнения и

продолжительностей работ. Гораздо более трудными являются проблемы, связанные с распределением ресурсов при планировании и управлении на основе сетевых графиков. Такие проблемы сводятся к задачам нелинейного целочисленного программирования весьма большого объема, решение которых представляет большие трудности даже при использовании самых мощных ЭВМ. Поэтому на практике ограничиваются упрощенными постановками задачи и частичной оптимизацией, без гарантии достижения абсолютного оптимума.

Для лучшего уяснения сущности проблем, встающих в связи с оптимизацией распределения ресурсов при планировании и управлении на основе сетевых графиков, рассмотрим следующий пример. Пусть совокупность подлежащих выполнению работ описывается сетевым графиком, изображенным на рис. 11.9. Предположим, что имеются всего два ресурса: R_1 и R_2, которые подлежат распределению (рабочие двух специальностей или оборудование двух различных видов).

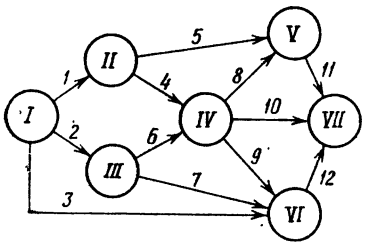

Рис. 11.9

На всем протяжении работ располагаем двумя единицами первого ресурса (например, двумя бригадами рабочих) и тремя единицами второго ресурса. Предположим, что время работ исчисляется месяцами, а затраты ресурсов (в ресурсо-месяцах) для работ 1, 2, ..., 12 приведены в табл. 11.8. В последнем столбце указаны суммарные затраты, необходимые для выполнения всех работ. Деля эти затраты на имеющиеся в наличии

Таблица 11.8

Ресурсы	Виды работ												
	1	2	3	4	5	6	7	8	9	10	11	12	Σ
R_1	2	4	6	2	0	2	2	2	0	2	2	4	28
R_2	3	3	0	3	3	0	3	3	3	6	3	6	36

ресурсы, получим минимальное время для выполнения всех работ: $T_1 = 28/2 = 14$ мес.; $T_2 = 36/3 = 12$ мес. Критическим ресурсом (ограничивающим снизу срок выполнения работ) в данном случае будет R_1. Минимально возможное время выполнения всех работ $T \geqslant 14$ мес. Это время представляет собой теоретический предел для оптимизации плана выполнения работы (не всегда достижимый).

Составляем начальный план, при котором соблюдается технологическая последовательность выполнения работ, заданная рассматриваемым сетевым графиком. В нашем случае в качестве такого плана может быть принято, например, последовательное выполнение всех работ в порядке роста их номеров. При этом каждый раз все имеющиеся ресурсы полностью направляются на выполнение одной работы и лишь после ее окончания перебрасываются на следующую.

Для первой работы как по первому, так и по второму ресурсу время t_1 ее выполнения одинаково и равно 1 мес. Потерь ресурсов (простоев) при этом нет. При выполнении второй работы критическим является первый ресурс. Вычисленное по нему время исполнения работы t_2 равно $4/2=2$ мес. По второму ресурсу при работе в течение этого срока располагаем $3\times 2=6$ ресурсо-месяцами. Поскольку фактически для выполнения второй работы требуются лишь 3 ресурсо-месяца, то по ресурсу R_2 образуется потеря, равная $6-3=3$ ресурсо-месяцам. Продолжая аналогичным образом, сведем время, а также суммарные затраты и потери ресурсов при выбранном порядке выполнения работ нарастающим итогом в табл. 11.9.

Таблица 11.9

Показатели	Виды работ											
	1	2	3	4	5	6	7	8	9	10	11	12
Затраты R_1	2	6	12	14	14	16	18	20	20	22	24	28
R_2	3	6	6	9	12	12	15	18	21	27	30	36
Потери R_1	0	0	0	0	2	2	2	2	4	6	6	6
R_2	0	3	12	12	12	15	15	15	15	15	15	15
Время	1	3	6	7	8	9	10	11	12	14	15	17

Очевидный недостаток построенного плана состоит в том, что происходят потери критического (первого) ресурса. Именно в результате этих потерь полученный срок выполнения работ (17 мес.) на 3 мес. больше, чем теоретически допустимый предел.

Предположим, что ресурсы R_1 и R_2 допускают независимое использование и при выполнении любой работы могут расходоваться в любом порядке *). Основная идея по улучшению начального плана при таком предположении состоит в том, чтобы

*) Взаимозависимые ресурсы, например каменщики и подъемные механизмы, обычно объединяются в единый комплексный ресурс. Если работа допускает использование ресурсов только в одном порядке, то ее следует свести к нескольким работам, введя новые события на исходном сетевом графике.

начинать использование освободившегося ресурса для выполнения последующих работ, не дожидаясь полного высвобождения всех ресурсов. Очевидно, что такая тактика приведет к успеху лишь в том случае, когда работа, в которой начинается использование освободившегося ресурса, является критической в отношении именно этого ресурса. Это достигается изменением порядка выполнения работ, разумеется, при строгом соблюдении условий очередности работ, вытекающих из исходного сетевого графика.

Для облегчения выбора надлежащей очередности выполнения работ разделим их на три группы: критичные по первому, по второму и по обоим ресурсам. В первую группу попадают работы 2, 3, 6, во вторую — работы 5, 9, 10 и в третью — работы 1, 4, 7, 8, 11, 12. Работы последней группы могут выполняться в любое время, допускаемое исходным сетевым графиком, не создавая нежелательных потерь ресурсов. Проблема состоит в определении правильной последовательности работ первой и второй групп.

Таблица 11.10

Показатели	Виды работ											
	1	2	5	4	7	6	9	3	10	9	11	12
Время начала использования ресурса R_1	0	1	3	3	4	5	6	6	9	10	11	12
Время окончания использования ресурса R_1	1	3	3	4	5	6	6	9	10	11	12	14
Время начала использования ресурса R_2	0	1	2	3	4	5	5	6	6	10	11	12
Время окончания использования ресурса R_2	1	2	3	4	5	5	6	6	8	11	12	14
Время начала работы	0	1	2	3	4	5	5	6	6	10	11	12
Время окончания работы	1	3	3	4	5	6	6	9	10	11	12	14
Использование R_1	2	6	6	8	10	12	12	18	20	22	24	28
R_2	3	6	9	12	15	15	18	18	24	27	30	36
Потери R_1	0	0	0	0	0	0	0	0	0	0	0	0
R_2	0	0	0	0	0	0	0	3	6	6	6	6

Поскольку при выполнении второй работы образуется резерв 3 ресурсо-месяца по ресурсу R_2, то желательно, не дожидаясь окончания этой работы, начать критичную по R_2 работу, в которой этот ресурс желательно полностью использовать. Такой является, очевидно, работа 5. Для работы 3 такими могут быть работы 5, 9, 10, а для работы 6—9, 10. Одним из возможных

решений, дающих требуемую очередность, служит последовательность выполнения работ 1, 2, 5, 4, 7, 6, 9, 3, 10, 8, 11, 12, для которой все показатели приведены в табл. 11.10.

В рассматриваемом случае удалось достичь теоретического предела для времени выполнения работ и полностью избежать потерь критического ресурса. Нетрудно понять, что поскольку речь идет о нахождении надлежащей перестановки исходного порядка выполнения работ, то вместо примененного приема угадывания нужного порядка можно применить общий метод динамического программирования, аналогичный тому, который был применен в задаче о коммивояжере. Отличие состоит лишь в том, что рассмотрению подлежат не все возможные подстановки, а только те, в которых последовательность работ удовлетворяет требованиям исходного сетевого графика.

В общем случае при оптимизации использования ресурсов приходится рассматривать возможности параллельного выполнения работ, что еще более усложняет задачу.

11.8. Целевое прогнозирование

Мы рассмотрим лишь одну, принадлежащую автору, модель целевого прогнозирования. Чтобы проще уяснить себе сущность модели, рассмотрим ее в конкретной интерпретации — как задачи прогнозирования научно-технического прогресса. Предположим, что требуется оценить вероятное время и пути решения некоторого числа нерешенных научно-технических проблем $s_1, s_2, \ldots, s_m$. Среди них могут быть как проблемы прикладного характера (например, проблема повышения производительности труда в угольной промышленности в 10 раз), так и абстрактные проблемы (например, построение единой теории поля). Предположим далее, что каждой из указанных проблем s_i приписан некоторый весовой коэффициент λ_i, определяющий относительную важность этой проблемы по сравнению с остальными ($i = 1, 2, \ldots, m$). Поставим в качестве цели решение всех указанных проблем, а в качестве критерия качества управления — минимизацию суммы $\sum \lambda_i \tau_i$, где τ_i — срок, потребовавшийся для решения проблемы s_i при заданных суммарных ресурсах R, отпущенных на решение всех этих проблем.

Первоначальная структуризация проблемы состоит в том, чтобы дополнить список поставленных проблем $s_1, s_2, \ldots, s_m$, называемых далее *конечными* или *основными целями*, новыми проблемами — *промежуточными целями* $s_{m+1}, \ldots, s_{m+n}$, решение которых может оказаться необходимым или полезным для достижения конечных целей. *Параметризация* состоит в приписывании каждому из выделенных элементов s_i двух булевых параметров $\alpha_i(t)$, $\beta_i(t)$ $(i = 1, 2, \ldots, m+n)$, из которых первый ха-

рактеризует состояние элемента (проблема s_i решена или не решена), второй — управляющее воздействие (проблема s_i поставлена в план и финансируется или нет). Введем также оценку вероятности $p_i(t)$ того, что к моменту времени t проблема s_i окажется решенной, т. е. вероятность того, что $\alpha_i(t)=1$ $(i=1, 2, \ldots, m+n)$.

С целью установления зависимости между параметрами для каждой проблемы s_i привлекается группа экспертов, формулирующих условия, состоящие в том, что некоторые из целей $s_{i_1}, \ldots, s_{i_k}$ считаются уже достигнутыми, и дающих оценки времени t_i достижения цели s_i после выполнения поставленного условия. Эта оценка может производиться для нескольких различных величин ассигнований v_i, которые могут быть отведены для решения проблемы s_i, так что t_i являются в общем случае функциями от v_i. В результате возникают зависимости между параметрами α_i:

$$\alpha_i(t)=\alpha_{i1}(t-t_i)\,\alpha_{i2}(t-t_i)\ldots\alpha_{ik}(t-t_i). \qquad (11.10)$$

Для каждого i будет получено несколько таких зависимостей, в соответствии с числом привлеченных экспертов (некоторые зависимости могут, разумеется, совпадать, но нам все равно удобно считать их различными). Каждая из зависимостей получает весовой коэффициент r_{ij}, где i — номер цели, j — номер эксперта в группе, оценивавшей эту цель *). Из зависимостей вида (11.10), установленных различными экспертами, получаем следующую очевидную формулу для вычисления вероятности:

$$p_i(t)=\frac{\sum\limits_{j=1}^{lj} r_{ij}p_{ij1}(t-t_{ij})\,p_{i/2}(t-t_{ij})\cdots p_{ijkj}(t-t_{ij})}{\sum\limits_{j=1}^{lj} r_{ij}}, \qquad (11.11)$$

где $ij1$, $ij2$, ..., $ijkj$ — номера промежуточных целей, выдвинутых j-м экспертом в качестве условия достижения цели s_i, а через t_{ij} обозначена его оценка времени достижения цели s_i после выполнения поставленного условия.

Для того чтобы по уравнениям вида (11.11) можно было последовательно найти функции $p_i(t)$ для всех целей s_i $(i=1, 2, \ldots, m+n)$, нужно в процессе построения модели произвести дополнительные уточнения. Один из возможных путей заключается в том, чтобы, вводя новые вспомогательные цели и дополнительно работая с экспертами, добиваться расслоения всех целей (как конечных, так и вспомогательных) на пересекающиеся множества $M_0, M_1, \ldots, M_p$. Множество M_0 составляется из целей, имеющих лишь безусловные оценки времени своего достижения.

*) Процедура подсчета r_{ij} включает в себя самооценку эксперта, его оценку другими экспертами и ряд дополнительных приемов.

Для целей в любом из множеств M_i в качестве условий могут выступать лишь цели из множеств $M_0, M_1, \ldots, M_{i-1}$ ($i=1, 2, \ldots, p$). При выполнении этого требования формулы (11.11) оказываются достаточными для вычисления вероятностей $p_i(t)$ для всех целей при всех значениях t (с некоторым интервалом дискретности).

Чтобы лучше уяснить себе сущность таких вычислений, рассмотрим простейший пример. Пусть нам заданы две конечные цели s_1 и s_2 с весами 2 и 1 соответственно и две вспомогательные цели s_3 и s_4, каждая из которых оценивалась двумя экспертами. Данные оценок сведены в табл. 11.11 (для простоты пренебрегаем здесь зависимостью оценок времени t_{ij} от размера выделяемых ассигнований).

Таблица 11.11

Цели	s_1	s_2	s_3	s_4
Эксперты	1 2	1 2	1 2	1 2
Условные	$(s_2, s_4)(s_3, s_4)$	$(s_3, s_4)(s_3)$	(s_4) —	—
Оценки времени, t_{ij}	2 3	2 4	1 2	2 3
Вес r_{ij}	3 2	2 3	1 3	3 2

Из табл. 11.11 непосредственно видно, что цели распадаются на слои M_0 (цель s_4), M_1 (цель s_3), M_2 (цель s_2), M_3 (цель s_1). Функции $p_i(t)$ будем задавать векторами $(p_i(0), p_i(1), p_i(2), p_i(3), \ldots)$. Начиная с некоторого t, все $p_i(t)$ в этих векторах равняются 1 и в этих местах векторы могут быть оборваны (ибо недостающие компоненты при необходимости легко могут быть выписаны). Если при оценке цели эксперт не выдвинул никаких условий, то произведение $r_{ij}p_{ij1}(t-t_{ij})\ldots p_{ijkj}(t-t_{ij})$ в уравнениях (11.11), очевидно, должно быть заменено на $r_{ij}Q(t-t_{ij})$, где $Q(t)=0$ при $t<0$ и $Q(t)=1$ при $t\geqslant 0$. В таком случае имеем

$$p_1(t) = \frac{3}{5}Q(t-2) + \frac{2}{5}Q(t-3) =$$

$$= \frac{3}{5}(0;\ 0;\ 1;\ 1) + \frac{2}{5}(0;\ 0;\ 0;\ 1) = (0;\ 0;\ 0{,}6;\ 1);$$

$$p_3(t) = \frac{3}{4}Q(t-2) + \frac{1}{4}p_4(t-1) =$$

$$= \frac{3}{4}(0;\ 0;\ 1;\ 1;\ 1) + \frac{1}{4}(0;\ 0;\ 0;\ 0{,}6;\ 1) = (0;\ 0;\ 0{,}75;\ 0{,}9;\ 1);$$

$$p_2(t) = \frac{2}{5}p_3(t-2)\,p_4(t-2) + \frac{3}{5}p_3(t-4) =$$

$$= (0;\ 0;\ 0;\ 0;\ 0{,}18;\ 0{,}36;\ 0{,}85;\ 0{,}94;\ 1);$$

$$p_1(t) = \frac{3}{5}p_2(t-2)\,p_4(t-2) + \frac{2}{5}p_3(t-2)p_4(t-3) =$$

$$= (0;\ 0;\ 0;\ 0;\ 0;\ 0{,}18;\ 0{,}47;\ 0{,}61;\ 0{,}91;\ 0{,}97;\ 1).$$

Полученные данные могут быть использованы для прогноза наиболее вероятного времени достижения каждой из рассматриваемых целей. Обычно в качестве такого времени принято считать *медиану распределения*, т. е. время t, для которого вероятность $p_i(t)$ равна 0,5. Если считать, что между найденными точками вероятность $p_i(t)$ меняется по линейному закону, как указано для случая $p_4(t)$ на рис. 11.10, то медианы N_i распределений $p_i(t)$ будут равны

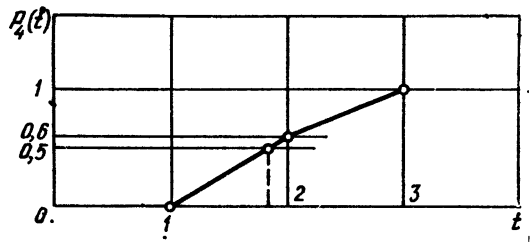

Рис. 11.10.

$$N_1 = 6\frac{3}{14} \approx 6{,}21;$$

$$N_2 = 5\frac{2}{7} \approx 5{,}29;$$

$$N_3 = 1\frac{2}{3} \approx 1{,}67;$$

$$N_4 = 1\frac{5}{6} \approx 1{,}83.$$

Степень неопределенности прогноза характеризуется обычно так называемыми *квартилями*: нижний квартиль Q_i' есть значение t, при котором $p_i(t) = 0{,}25$, а для верхнего квартиля Q_i'' имеем $p(Q_i'') = 0{,}75$. Нетрудно подсчитать, например, что $Q_4' = 1\frac{5}{24} \approx 1{,}42$, $Q_4'' = 2\frac{3}{8} \approx 2{,}38$. В качестве единой меры неопределенности ΔQ_i может быть выбрана разность $Q_i'' - Q_i'$. Для четвертой цели эта разность равна $\Delta Q_4 = 23/24 \approx 0{,}96$.

Для уточнения прогноза экспертиза делается непрерывной: всякий раз, когда тот или иной эксперт изменяет свое мнение, он посылает соответствующее сообщение в систему, где каждый раз осуществляется пересчет функций распределения $p_i(t)$. Чтобы ускорить процесс улучшения прогноза, производится ранжировка целей (как основных, так и промежуточных) в соответствии с их *информационной значимостью*. Наибольшей информационной значимостью обладают цели, для которых уточнение прогноза вызывает наибольшее уточнение прогноза для основных целей.

Количественной мерой *информационной значимости* i-й цели может служить специальный коэффициент I_i, вычисляемый следующим образом. Прежде всего, распределение $p_i(t)$ заменяется дельта-распределением $p_i(t)$ с той же самой медианой $N_i\,(p_i'(t)) = 0$ при $t < N_i$ и $p_i'(t) = 1$ при $t \geqslant N_i$. Для основных целей $s_1, \ldots, s_m$ вычисляются уменьшения d_j разностей ΔQ_i между соответствующими квартилями ($j = 1, 2, \ldots, m$). Тогда величину

$$\sum_{j=1}^{m} \lambda_j d_j \Big/ \sum_{j=1}^{m} \lambda_j$$

естественно принять за меру информационной значимости i-й цели. Понятие информационной значимости помогает концентрировать внимание на тех целях, по которым уточнение прогноза является наиболее важным. Сам процесс уточнения может использовать различные процедуры дополнительного обращения к экспертам, целенаправленное снабжение их информацией, проведение совещаний с соответствующими группами экспертов и т. п.

Близким к понятию информационной значимости является понятие *значимости цели* (по срокам). По определению, *коэффициентом значимости* i-й цели называется величина

$$z_i = \sum_{j=1}^{m} \lambda_j \Delta_i N_j \Big/ \sum_{j=1}^{m} \lambda_{j_i}$$

где $\Delta_i N_j$ — приращение медианы распределения $p_j(t)$ при условии сдвига вправо на один временно́й интервал распределения $p_i(t)$, т. е. при замене функции $p_i(t)$ функцией $p_i(t-1)$.

В рассмотренном выше примере для вычисления коэффициента значимости z_3 необходимо, сохранив распределение $p_4(t)$, заменить распределение $p_3(t)=(0;\ 0;\ 075;\ 09;\ 1)$ на $p_3(t)=(0;\ 0;\ 0;\ 0{,}75;\ 0{,}9;\ 1)$. Тогда

$$p_2'(t) = \frac{2}{5} p_3'(t-2)\, p_4(t-2) + \frac{3}{5} p_3'(t-4) =$$

$$= \frac{2}{5}(0;\ 0;\ 0;\ 0;\ 0;\ 0{,}75;\ 0{,}9;\ 1)(0;\ 0;\ 0;\ 0;\ 0{,}6;\ 1) +$$

$$+ \frac{3}{5}(0;\ 0;\ 0;\ 0;\ 0:\ 0;\ 0{,}75;\ 0{,}9;\ 1) =$$

$$= (0;\ 0;\ 0;\ 0;\ 0;\ \ 0{,}3;\ 0{,}36;\ 0,\ 85;\ 0{,}94;\ 1);$$

$$p_1'(t) = \frac{3}{5} p_2'(t-2)\, p_4(t-2) + \frac{2}{5} p_3'(t-3)\, p_4(t-3) =$$

$$= \frac{3}{5}(0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0{,}3;\ 0{,}36;\ 0{,}85;\ 0{,}94;\ 1) \times$$

$$\times (0;\ 0;\ 0;\ 0;\ 0{,}6;\ 1) + \frac{2}{5}(0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0{,}75;\ 0{,}9;\ 1) \times$$

$$\times (0;\ 0;\ 0;\ 0;\ 0;\ 0{,}6;\ 1) =$$

$$= (0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0{,}3;\ 0{,}54;\ 0{,}62;\ 0{,}91;\ 0{,}96;\ 1).$$

Отсюда $N_1' = 6\frac{5}{9} \approx 6{,}83$; $N_2' = 6\frac{2}{7} \approx 6{,}29$. Поскольку $\lambda_1 = 2$, $\lambda_2 = 1$, то

$$z_3 = \frac{2(N_1' - N_1) + 1(N_2' - N_2)}{2+1} = \frac{2(6{,}83 - 6{,}21) + (6{,}29 - 5{,}29)}{3} \approx 0{,}75.$$

Коэффициенты значимости промежуточных целей могут употребляться для выбора начальных участков путей достижения конечных целей, когда прогноз еще не доведен до такой степени точности, чтобы был возможен однозначный выбор всего оптимального пути. В самом деле, нетрудно понять, что цели с большими коэффициентами значимости являются наиболее полезными (а возможно, даже и необходимыми) для достижения конечных целей. Поэтому, когда неопределенность прогноза еще не позволяет выбрать оптимальные пути для достижения конечных целей, задача управления переформулируется и временной целью управления может стать достижение наиболее важных промежуточных целей.

Еще в более непосредственном виде такая временная подмена целей может быть достигнута введением весов для всех промежуточных целей. Эти веса мы будем обозначать через $\mu(s_k)$. По определению,

$$\mu(s_k) = \sum_{i=1}^{m} \lambda_i p_{ik},$$

где λ_i — вес i-й основной цели; p_{ik} — вероятность того, что при ее достижении окажется необходимым предварительно достигнуть k-ю промежуточную цель. Поскольку основные цели также могут выступать в качестве промежуточных (как, например, цель в рассмотренном примере), им могут быть приписаны, кроме исходных базовых весов λ_i, также относительные веса $\mu(s_i)$. Для примера, рассмотренного выше, легко получаем $\mu(s_1) = \lambda_1$, $\mu(s_2) = 0{,}6\lambda_1 + \lambda_2$, $\mu(s_3) = \mu(s_4) = \lambda_1 + \lambda_2$.

Управление в описанных моделях состоит в переводе тех или иных целей в реальный план. Планом можно, разумеется, считать и саму исходную модель, однако, благодаря наличию в ней альтернативных путей, при этом пришлось бы достигать многих промежуточных целей, которые впоследствии могут оказаться ненужными. Таким образом, сущность управления состоит в выборе альтернатив (одной или нескольких) для достижения каждой конечной цели. Эти альтернативы характеризуются множествами всех целей, которые надо достичь в процессе достижения данной цели.

Основанием для выбора той или иной альтернативы служат три основных момента. Главным является степень уверенности, что данная альтернатива приведет к рассматриваемой конечной цели. Основанием для такой уверенности могут служить прежде всего достаточно высокая вероятность данной альтернативы (зависящая, в частности, от числа высказавшихся за нее экспертов) и степень согласованности мнений экспертов, высказавшихся за эту альтернативу. Мерой указанной согласованности может слу-

жить разность ΔQ_i между квартилями распределения времени достижения (конечной) цели s_i при учете только рассматриваемой альтернативы (с отбрасыванием всех остальных).

В рассмотренном выше примере для достижения конечной цели имеем две альтернативы: (s_3) и (s_3, s_4). Вероятность первой из них равняется, очевидно, произведению вероятностей достижения s_2 без s_4 и s_3 без s_4, т. е. $\frac{3}{5}\cdot\frac{3}{4}=\frac{9}{20}=0{,}45$, а вероятность второй — ее дополнению до 1, т. е. 0,55. Распределение по первой альтернативе выразится функциями $p_3'(t)=(0{,}01), p_2'(t)=$ $=p_3'(t-4)=(0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 1;\ 1)$. Хотя степень неопределенности здесь равна нулю, это, разумеется, не может служить основанием для выбора первой альтернативы ввиду ее невысокой вероятности.

Вторым основанием для выбора альтернативы (после степени уверенности) служит обычно ожидаемое время достижения заданной конечной цели s_i при отбрасывании всех прочих альтернатив (медиана распределения $p_i(t)$). Третьим основанием служат оценка затрат для достижения цели s_i при данной альтернативе и мера ее неопределенности (среднее квадратичное уклонение или разность между квартилями). Оценки затрат $R(s_i)$ производятся экспертами для каждой цели s_i (как основной, так и промежуточной) без включения затрат на достижение промежуточных целей, выдвинутых в качестве условий. В этом случае $R(s_i)$ можно считать независимыми случайными величинами и оценивать общие затраты по каждой альтернативе суммой затрат на все входящие в нее цели.

Выделение и оценка альтернатив могут производиться также для любых групп конечных или промежуточных целей. Так, нетрудно видеть, что в рассмотренном выше примере для группы (s_1, s_2) существует единственный вариант (s_1, s_2, s_3, s_4), требующий достижения всех рассматриваемых целей.

В случае, если не имеется достаточных оснований для выбора единственной альтернативы, для осуществления управления можно применять два основных приема. Один из них уже был описан выше — он заключается во включении в план ближайших целей с наибольшими коэффициентами значимости (или с наибольшими относительными весами) без предопределения последующего развития плана. Второй прием заключается в объединении близких альтернатив (отличающихся небольшим числом элементов). Например, альтернативы (s_1, s_3, s_5, s_6) и (s_1, s_4, s_5, s_6) могут быть объединены в один вариант $(s_1, s_3, s_4, s_5, s_6)$. Для возможности такого объединения существенно, чтобы цели, которыми различаются объединяемые варианты (в данном случае s_3 и s_4), имели сравнительно невысокие затраты для их реализации.

11.9. Объемно-календарное планирование

При описании методов *объемно-календарного планирования* (равно как и оперативно-календарного планирования) обычно специфицируют это описание в зависимости от уровня планирования (предприятие, объединение, главк, министерство) и от вида производства (непрерывное, дискретное, мелкосерийное, массовое и т. п.). Объем книги, однако, не позволяет провести подобное описание во всей его полноте. Остановимся поэтому лишь на основных принципах объемно-календарного планирования применительно к некоторой абстрагированной модели *экономического объекта* (безотносительно его конкретной специфики).

Прежде всего, всякий *экономический объект* для организации материального производства располагает некоторыми *ресурсами*, которые мы будем подразделять на два класса: *трудовые ресурсы* (рабочие разных специальностей) и *основные фонды* (рабочие помещения, оборудование, земля, продуктивный скот и др.). В соответствии с принятым *режимом работы* (сменность, выходные дни, отпуска), а также *планами изменения ресурсов* (набор и подготовка рабочих, ввод в действие новых фондов, замена и демонтаж устаревшего оборудования) на заданный плановый период T вычисляется общее количество ресурсо-часов, которым будет располагать предприятие (по каждому виду ресурса). Вектор располагаемого числа ресурсо-часов по всем видам ресурсов обозначим через $\bar{b}$.

При построении систем автоматизации планирования необходимо иметь программное обеспечение, которое в диалоговом режиме управляет возможными изменениями вектора $\bar{b}$. С этой целью планы (целевые программы) работ, направленные на изменение ресурсов (капитального строительства, реконструкций, подготовки кадров и т. п.), представляются (в памяти ЭВМ) в виде сетевых графиков. В руки же лиц, осуществляющих диалог, даются средства, позволяющие изменять *проекции этих графиков в заданный плановый период* T. Под проекцией здесь, как и выше (§ 11.6), понимается задание конкретных значений для моментов времени начала и конца каждой работы сетевого графика. Разумеется, при этом предполагается сохранение взаимозависимостей сроков окончания одних работ и начала других (технологической последовательности работ), определенных исходным сетевым графиком. В равной мере предполагается, что сроки выполнения работ в их реальных проекциях в плановый период не выходят за рамки первоначальных оценок их минимальных длительностей.

После задания проекции программное обеспечение должно автоматически рассчитывать графики и суммарные расходы ресурсов и материалов всех видов на выполнение всех работ гра-

фика, попавших в плановый период T. Вычисляются также, в пределах планового периода, определяемые сделанными проекциями сроки полного окончания работ, приводящих к изменениям тех или иных ресурсов основного производства и количества часов, которые эти ресурсы должны отработать до конца планового периода. Тем самым вектор $\bar{b}$ располагаемого числа ресурсо-часов оказывается заданным полностью и притом динамически (со всеми возможными вариациями).

Далее, в соответствии со спецификой рассматриваемого экономического объекта определяется и запоминается в ЭВМ *номенклатура продукции*, которую он может (но не обязательно будет) производить. Для каждого вида продукции фиксируются предполагаемые единицы ее измерения (штуки, тонны, рубли и т. д.), а также *таблицы* и алгоритмы (программы, пользующиеся этими таблицами) для перевода количества продукции разных видов из одних единиц измерения в другие.

Следующий вид информации, необходимый для планирования,— *технологические карты*, описывающие технологию производства различных видов продукции на данном экономическом объекте. В карте указывается прежде всего последовательность технологических операций на рассматриваемом объекте для производства соответствующего вида продукции. Удобным средством для представления таких последовательностей являются сетевые графики. Для каждой операции указываются (в расчете на одну единицу выпускаемой продукции) *нормы расхода ресурсо-часов* всех видов, используемых в данной операции (обычно агрегированные), а также *нормы расхода материалов* и *комплектующих изделий* в данной операции. Указываются также *минимальные технологические сроки* выполнения каждой операции (которые нельзя уменьшить за счет любой возможной концентрации ресурсов), а также (в случае серийного и массового производства) рациональные размеры партий поставляемой и выпускаемой продукции и размеры страховых запасов.

Для *структурированных* экономических объектов, которые разбиваются на отдельные *подразделения* (предприятия, цехи, участки), имеющие *свои собственные планы*, технологические карты выделяют группы операций, выполняемые отдельными подразделениями. Заметим, что в соответствии с общей концепцией ведения нормативного хозяйства, описанной в предыдущей главе, *прогнозные* (ориентировочные) технологические карты должны составляться на самых ранних стадиях проектирования новой продукции и будущей технологии ее производства. Все они должны находиться в памяти ЭВМ и уточняться по мере продвижения разработок новых технологий и соответствующей *подготовки производства* (создания необходимых инструментов, обучения рабочих и т. п.). При изменениях структуризации

объекта соответствующие изменения в группировках операций также должны немедленно вноситься в технологические карты.

В программное обеспечение описанной *технологической базы данных* вводятся описанные в предыдущей главе процедуры агрегации и дезагрегации нормативов. Заметим, что при агрегации в один (агрегированный) вид продукции может объединяться несколько видов продукции *специфицированной* номенклатуры, например легковые автомобили разных марок, костюмы различных размеров и фасонов и т. п. Другой вид агрегации предполагает нахождение средних нормативов на производство одного и того же вида продукции, выполняемого с помощью *различных технологий*. Как известно, средний (агрегированный) норматив q получается из частных (специфицированных) нормативов $q_1, \ldots, q_n$ по формуле

$$v = \sum_{i=1}^{n} \alpha_i q_i, \quad \sum_{i=1}^{n} \alpha_i = 1. \tag{11.12}$$

Здесь через α_i обозначены относительные доли производства отдельных специфицированных продуктов (или специфицированных технологий) в производстве агрегированного продукта за данный плановый период T. При оптимизации планов коэффициенты α_i могут служить *управляемыми параметрами*, разумеется, лишь в том случае, если состав оперативного продукта не задан жестко вышестоящими плановыми органами.

Особенно важно подчеркнуть значение коэффициентов α_i и формулы (11.12) при агрегации технологий производства одного и того же вида продукции или последовательно улучшаемого при переходе к новым технологиям. В таком случае *управление агрегацией* (т. е. коэффициентами α_i) представляет собой для рассматриваемого объекта по существу *управление научно-техническим прогрессом*. В случае, когда речь идет о выборе между технологиями, уже с у щ е с т в у ю щ и м и на объекте, мы имеем дело просто с расчетной задачей (выбора значений коэффициентов α_i). Будем называть такой выбор *прямым управлением агрегацией*.

Еще более важен случай, когда речь идет о *вновь внедряемых технологиях*. Подобный процесс неизбежно связан с дополнительными затратами (инвестициями) на разработки, подготовку производства, закупку нового оборудования, реконструкцию и даже на новое строительство. В этом случае имеет место процесс *косвенного управления агрегацией*, поскольку выбор коэффициента, определяющего относительную долю работы, выполняемой по новой технологии, зависит от *длительности функционирования* этой технологии на рассматриваемом объекте. В свою очередь эта длительность определяется сроками окончания работ по внедрению данной (i-й) технологии и следующей, более

совершенной ($(i+1)$-й) технологии. Тем самым управление агрегированным нормативом идет не непосредственно, а через *систему управления инвестициями* (разработками, подготовкой производства, капитальным строительством и т. п.). В результате выбор направления инвестиций, планирование и управление этим процессом на рассматриваемом объекте оказываются подчиненными в конце концов *задаче улучшения нормативов основного производства* на этом объекте.

Тем самым вместо обычных (последовательно развертываемых во времени) динамических моделей производства для учета влияний на него инвестиций предлагается необычный с точки зрения эконометрики подход *проекции динамики в статику*. Смысл его состоит в том, что через агрегированные по периоду планирования нормативы *динамические* (развернутые во времени) планы инвестиций оказывают влияние (вычисляемое по формуле (11.12)) на средние технико-экономические показатели основного производства по всему плановому периоду T. Разумеется, этот плановый период можно произвольным образом варьировать, получая, в частности, любые разбивки планов основного производства по подпериодам. В результате такой разбивки *окончательное представление* планов основного производства может быть выполнено в динамическом виде. Иными словами, чисто *объемные планы* (по всему периоду T) могут быть доведены до уровня *объемно-календарных планов* (пятилетний план разбит на годовые, годовые на квартальные и месячные и т. п.).

Как известно, в *эконометрике* (математической экономике) при работе с инвестициями используются динамические модели, развертывающие процесс функционирования рассматриваемого экономического объекта во времени в соответствии с динамикой инвестиций. При этом нормативы q_i являются не постоянными величинами, а функциями времени (зависящими от динамики инвестиционных процессов), что делает задачи оптимизации в подобных моделях чрезвычайно сложными. По существу, на практике для сколько-нибудь сложных объектов речь идет о задачах динамического программирования в пространствах столь большой размерности, что точные методы оптимизации даже при условии применения самых мощных ЭВМ оказываются попросту неприменимыми. Попытки упрощать задачу за счет сильной агрегации моделей превращают эти модели из реального инструмента планирования в лучшем случае в средство для *предплановых ориентировок*, которые могут служить, как правило, лишь для чисто качественных, а не точных количественных решений.

Предлагаемый метод проекции динамики в статику представляет собой по существу математическое оформление традиционно

сложившегося на практике метода, когда *этап объемного планирования предшествует этапу календарной разбивки планов*. При этом оптимизационные расчеты резко упрощаются, поскольку на каждом этапе планирования приходится иметь дело с постоянными нормативами и в основном с линейными (или близкими к ним) моделями. Что же касается *управления нормативами*, в чем как раз и состоит *основная задача оптимизации*, то оно выполняется *целенаправленным* (после выяснения степени важности возникающих ограничений) *изменением границ областей оптимизации*, как это предусматривается в общей методологии *системной оптимизации* (§ 8.12). В данном случае соответствующие границы (зависящие от средних значений нормативов) определяются направлениями инвестиций и сроками их реализации. Хотя при этом, как и вообще при использовании методов системной оптимизации, вообще говоря, не получается абсолютного оптимума в строгом математическом смысле этого слова, однако, благодаря работе в реальных (неупрощенных) пространствах (как планов, так и критериев), в результате получаются не наметки оптимального плана (которые еще требуется расшифровать и детализировать), а *реальный план производства и развития объекта*, причем в разумной степени (определяемой реальными возможностями) оптимизированный. Заметим, что в процессе расшифровки и детализации точного оптимального плана, полученного на высокоагрегированной модели, происходит, как правило, значительно большее отклонение от действительно оптимального реального плана, чем при предлагаемом методе системной оптимизации.

Целью объемного планирования основного производства любого экономического объекта является прежде всего нахождение вектора $x = \begin{Vmatrix} x_1 \\ \cdots \\ x_n \end{Vmatrix}$ объемов выпуска продукции различных видов за плановый период T, а также вектора $a = \begin{Vmatrix} a_1 \\ \cdots \\ a_n \end{Vmatrix}$ *материально-технического снабжения*, т. е. объемов *поставок извне* продукции разных видов (материалов и комплектующих изделий), необходимых для выполнения данного плана. При этом план должен быть *сбалансирован по ресурсам*. Иными словами, запасы ресурсо-часов по различным видам ресурсов, которыми располагает объект на период T, должны быть достаточны для выполнения плана. Оптимальность плана означает, что соответствующая ему точка $r = (r_1, \ldots, r_k)$ в *пространстве критериев* r_i должна быть с точки зрения ЛПР (лица, принимающего решения) лучше точек, соответствующих другим возможным планам.

Необходимо отметить, что априорное задание функции $r = f(r_1, \ldots, r_n)$, сводящей многокритериальную задачу оптимиза-

ции к однокритериальной, на практике оказывается чаще всего невыполнимой задачей. В большинстве случаев эта функция задается столь сложным алгоритмом, что вместо его задания априори предпочитают оценивать «качество» различных точек в пространстве критериев в процессе прямого диалога с ЛПР. При этом роль предварительного (априорного) анализа пространства критериев сводится обычно к максимально возможному «сжатию» (уменьшению размерности) пространства критериев. Поскольку для различных объектов в разное время и в разных ситуациях действуют разные системы критериев (прибыль, производительность труда, фондоемкость, различного рода качественные критерии и т. п.), при построении автоматизированных систем объемно-календарного планирования целесообразно «критериальную» часть программного обеспечения строить как *открытую систему*. Иными словами, наряду с законченными программами для быстрого вычисления общеупотребительных критериев, в системе необходимо иметь средства для быстрого включения в нее программ вычисления любых других критериев, которые могут быть придуманы и законодательно утверждены.

Предполагая, что планы инвестиций (т. е. проекции в заданный плановый период T соответствующих целевых программ) вместе с их материальным и ресурсным обеспечением заданы, мы должны иметь возможность быстрого вычисления вектора b общих запасов ресурсо-часов не только на заданный плановый период T, но и на любой его подпериод. Заметим, что при одновременном существовании на объекте ресурсов одинакового назначения (предназначенных для выполнения одних и тех же операций), но различной производительности, целесообразно предусматривать возможность их агрегации в один ресурс, вводя соответствующие коэффициенты для перевода в ресурс одной и той же производительности.

Далее, из технологической базы данных (массива технологических карт и списка полных технологий) и планов инвестиций (определяющих сроки готовности тех или иных технологий) с помощью специальных программ вычисляются *матрицы коэффициентов полных затрат* $A = \|a_{ij}\|$, $B = \|b_{ij}\|$ по всем позициям материально-технического снабжения и соответственно по всем видам ресурсов (точнее, ресурсо-часов). Коэффициент a_{ij} означает полные затраты i-й позиции материально-технического снабжения (материалов или комплектующих изделий) на производство одной единицы продукции по j-й технологии, а коэффициент b_{ij} — полные затраты на то же самое производство ресурсо-часов по i-му ресурсу. Тогда с помощью формул

$$a = Ax, \quad b = Bx, \quad x = \begin{Vmatrix} x_1 \\ \cdots \\ x_s \end{Vmatrix}, \tag{11.13}$$

определяются в *первом приближении* вектор материально-технического снабжения a и вектор b полных затрат ресурсо-часов для плана производства, выражаемого вектором x, где через x_j обозначено общее количество продукции, выпущенное за период T по j-й технологии ($j = 1, 2, \ldots, s$).

Слова «в первом приближении» отмечают прежде всего тот факт, что производственный процесс не замыкается рассматриваемым плановым периодом. Вступая в этот период и выходя из него, рассматриваемый объект обладает некоторыми *оборотными фондами*, которые материализованы в так называемом *незавершенном производстве*. В незавершенное производство $P(t)$ на любой данный период времени t включаются запасы продукции, получаемой предприятием по материально-техническому снабжению, а также вся продукция (выпускаемая на объекте), находящаяся на той или иной стадии обработки и еще не поставленная потребителям. Поэтому истинные размеры материально-технического снабжения и затрат ресурсов за период T будут, вообще говоря, отличаться от векторов a и b, находимых по формулам (11.13). Кроме того, для уменьшения размерности задачи обычно не все технологические изменения, меняющие технологические коэффициенты a_{ij} и b_{ij}, выделяются в новые технологии. Чаще всего прибегают к агрегированным (смешанным) технологиям. Но в таком случае при изменении пропорций, в которых смешиваются агрегируемые технологии, будут происходить изменения коэффициентов a_{ij}, b_{ij}, а следовательно, и векторов a, b.

Однако, несмотря на приближенный характер формул (11.13), ими обычно можно воспользоваться для предплановых оптимизационных расчетов, особенно в случаях, когда технологический цикл производства мал по сравнению с периодом планирования T, а рост производства в течение периода T не очень велик. С этой целью выделяют какой-либо критерий γ_i или какую-либо функцию $\varphi(\gamma_1, \ldots, \gamma_k)$ от этих критериев и находят *оптимальный план* производства $x = \begin{Vmatrix} x_1 \\ \cdots \\ x_s \end{Vmatrix}$ (по выделенному критерию γ_i или φ) за период T в области, задаваемой системой неравенств

$$Bx \leqslant b, \tag{11.14}$$

где b — вектор запаса ресурсо-часов, которым будет располагать рассматриваемый объект в течение планового периода T. К этим неравенствам, в случае наличия априорных ограничений по материально-техническому снабжению, могут добавляться все или часть неравенств вида

$$Ax \leqslant a. \tag{11.15}$$

Наконец, вышестоящим плановым органом могут быть заданы априорные ограничения на все или на некоторые плановые показатели x_i, которые в общем случае будут иметь вид

$$x' \leqslant x \leqslant x'', \quad x' \geqslant 0. \tag{11.16}$$

При отсутствии ограничений по j-й компоненте плана соответствующее неравенство сводится к тривиальному неравенству $0 \leqslant x_j < \infty$, которое предполагается выполненным во всех случаях.

Если критерий оптимизации представляет собой линейную функцию от компонент x_j плана x, то мы приходим к задаче линейного программирования, методы решения которой были рассмотрены в гл. VIII. Это будет, например, при использовании критерия максимума валовой продукции: $\max px$, где $p = \|p_1, \ldots, p_s\|$ — вектор оптовых цен на продукцию, выпускаемую различными технологиями на рассматриваемом объекте. Другие возможные критерии могут оказаться нелинейными. Таким будет, например, прибыль при нелинейной зависимости зарплаты от величины выработки (что чаще всего имеет место) или максимальный относительный уровень потерь использования основных фондов различных видов. В подобных случаях решаются задачи нелинейного (чаще всего — выпуклого) программирования, также описанные в гл. VIII.

Наконец, в случае мелкосерийного производства, когда выпуск продукции измеряется штуками, возможно использование методов целочисленного программирования.

После нескольких туров подобных *оптимизационных прикидок* с разными критериями оптимизации ЛПР могут сформулировать один из вариантов реального плана, использующих эти прикидки (т. е. вариантов с упором на наиболее выгодную продукцию в условиях действий заданных априорных ограничений (11.16)). При этом целесообразно в выделенном для дальнейшей работы варианте плана несколько выйти за рамки ограничений по ресурсам и материально-техническому снабжению, задаваемых неравенствами (11.14) и (11.15). При этом в векторах $b - Bx$ и $a - Ax$ возникают некоторые отрицательные компоненты, свидетельствующие о дефицитах по соответствующим позициям ресурсов и материально-технического снабжения. Программное обеспечение ранжирует эти дефициты по их относительным величинам (в процентном выражении) и выдает их плановикам и ЛПР для отработки *неформализованных мер* по их уменьшению или даже полной ликвидации. Главными из таких мер являются изменения планов инвестиций, управление будущими нормативами на ранних стадиях проектирования, изменения структуры производства (за счет увеличения относительных долей продукции, некритичной к возникшим дефици-

там), возможные изменения режимов работы (например, переход от двухсменной работы к трехсменной) и т. п.

Очень важно, что в возникающем таким образом процессе системной оптимизации плана основного производства осуществляется целенаправленное управление инвестиционным процессом и проектно-конструкторскими работами, обеспечивающее оптимизирующее влияние на основное производство не стихийного, а *сознательно управляемого научно-технического прогресса.* Поэтому в строящейся по этому принципу АСОУ нет возможности выделить из общего процесса управления такие традиционно обосабливаемые подсистемы, как системы управления капитальным строительством, НИОКР, а тем более подсистемы сбыта и материально-технического снабжения. Все они оказываются объединенными в общей (интегрированной) динамической модели рассматриваемого объекта. Поэтому строящиеся по этому принципу АСОУ принято называть *интегрированными.* Разумеется, интеграция на уровне организационного управления в рамках одного объекта не исключает и другие виды интеграции. В первую очередь это интеграция организационного и технологического управления в рамках одного объекта, а также интеграция в *Общегосударственную автоматизированную систему* (ОГАС) систем управления отдельными объектами одинаковых или различных уровней. Объединение систем управления одинаковых уровней (например, предприятий) принято называть *горизонтальной интеграцией,* а разных уровней (например, предприятие — главк) — *вертикальной интеграцией.*

Для осуществления вертикальной интеграции каждый экономический объект, за исключением самого мелкого, структурируется, т. е. разбивается на соответствующие подразделения. Взаимодействия с системами управления своих подразделений, система управления объектов осуществляет структуризацию найденных планов. Поскольку каждое подразделение может рассматриваться как отдельный экономический объект, то процесс объемного планирования для него производится в принципе тем же способом, который уже был описан для объекта в целом. В случае, когда подразделение выступает в качестве неотделимой составной части в едином технологическом процессе, сплапированном для всего объекта целиком, оно получает соответствующее плановое задание в готовом виде. В процессе же последующей диалоговой оптимизации могут прорабатываться различного рода изменения этих заданий (по инициативе как сверху, так и снизу). Те из них, которые дали положительный результат (в смысле уменьшения величины имеющихся дефицитов), должны приниматься и включаться в план. В случае невозможности полного устранения дефицитов в процессе системной оптимизации для получения окончательно сбалансиро-

ванного плана необходимо уменьшение плановых заданий. Подобное решение принимается ЛПР также целенаправленно за счет отклонения в первую очередь таких позиций плана, которые наименее отражаются (с точки зрения ЛПР) на системе показателей, характеризующих качество плана.

Заметим, что все изменения в планах в процессе как оптимизации, так и окончательной балансировки инициируются людьми прежде всего в *направлении уменьшения максимального из имеющихся на данный момент относительных дефицитов* и обсчитываются на ЭВМ для нахождения точной величины такого уменьшения (если оно вообще имеет место). Для ориентировки о местах, откуда можно черпать ресурсы для покрытия дефицитов, из ЭВМ выводятся сведения не только о дефицитах, но и об имеющихся *избытках* ресурсов. В задачу оптимизации планов входит не только уменьшение дефицитов, но и максимально возможное устранение имеющихся избытков. С этой целью рассматриваются предложения о перераспределении тех или иных ресурсов между различными подразделениями, а также предложения о загрузке избыточных ресурсов дополнительным производством такой продукции, которая не затрагивает неизбыточных, а тем более дефицитных ресурсов.

После окончания процесса системной оптимизации объемных планов этот процесс повторяется для всех выделенных подпериодов исходного планового периода T. Он выполняется тем же способом, который уже был описан выше, и поэтому не требует дополнительного описания. Заметим лишь, что при использовании проекции динамики в статику сбалансированность плана по всему периоду в целом не обязательно гарантирует его сбалансированность по отдельным подпериодам. Поэтому в процессе разбивки составленного плана по подпериодам, т. е. перехода к *объемно-календарному плану*, может возникнуть необходимость возврата на прежний (чисто объемный) уровень и дополнительной корректировки ранее составленного объемного плана. Этот процесс может повторяться несколько раз.

Процесс системной оптимизации объемных и объемно-календарных планов резко упрощается при переходе на *непрерывное планирование*. Ведь в таком случае каждый раз речь идет не о полном плане на весь период T, а о его дополнении на относительно малый период ΔT. Поскольку до вновь планируемого временно́го интервала $[T, T+\Delta T]$ план уже был составлен и оптимизирован ранее, то проекцию можно сразу делать на этот последний интервал, дополняя ее проекцией на весь плановый период только в относительно редких и, главное, более простых в расчетном отношении случаях возникновения необходимости внесения коррекций в ранее составленный план. Тем самым этап объемного планирования для периода $[T, T+\Delta T]$ фактически

объединяется с задачей объемно-календарного планирования для полного периода $[\Delta T,\ T+\Delta T]$.

Заметим еще, что при составлении объемных и объемно-календарных планов для границ плановых периодов определяются объемы незавершенного производства. При этом для производств с длительными циклами изготовления продукции (например, в судостроении) может фиксироваться для неоконченного изделия не только список уже выполненных производственных операций, но и степень завершенности незаконченных операций (обычно в процентах). Для основных фондов на границах плановых периодов производятся также вычисления их текущих стоимостей (с учетом амортизации). Аналогичным образом на этих границах фиксируются планируемые суммы денег на различных статьях банковских счетов рассматриваемых экономических объектов. На каждый же плановый промежуток рассчитываются (также по различным статьям) денежные расходы и доходы этих объектов. Баланс расходов и доходов должен быть при этом увязан с планируемыми переходными остатками денег на соответствующих статьях банковских счетов. Алгоритмы подобных расчетов хорошо известны из курсов бухгалтерского учета, так что их дальнейшую расшифровку мы опускаем. Необходимо лишь иметь в виду динамический характер соответствующего программного обеспечения: при любых изменениях планов производства или инвестиций должны автоматически выполняться необходимые корректировки всех позиций *финансового плана* (т. е. планируемых расходов, доходов и остатков по всем статьям бухгалтерского учета).

Если основной плановый период T достаточно велик, то подпериоды, на которые он делится, могут быть также достаточно большими. В этом случае обычно первый из подпериодов (например, первый год пятилетки) разбивается на еше более мелкие подпериоды (например, месяцы). Для первого из самых мелких подпериодов объемно-календарного плана осуществляется оперативно-календарное планирование, доводящееся до сменных заданий на отдельные производственные операции. Внутри же очередной смены производственный ритм в случае необходимости может рассчитываться по часам, минутам и даже секундам. Вся эта иерархия планов разной длительности и степени дезагрегации должна храниться в памяти ЭВМ в динамическом виде — с автоматическим распространением на все взаимосвязанные компоненты планов любых, вносимых в планы изменений.

При наличии подобной динамической иерархической системы непрерывно пролонгируемых планов *текущее управление* их выполнением состоит главным образом в регистрации всех возникающих отклонений и их устранении. Устранение отклоне-

ний может производиться за счет использования заранее запланированных резервов (как для отдельных подразделений, так и для всего объекта в целом). Если этих резервов оказывается недостаточно, производятся быстрые пересчеты планов, которые вначале затрагивают самый короткий начальный подпериод, и лишь затем, в случае невозможности устранения отклонений в этом подпериоде, распространяются на более дальние участки планов.

11.10. Автоматизация плановых расчетов на общегосударственном уровне

Здесь мы опишем (в упрощенном виде) лишь одну из возможных схем *макроэкономического* (общегосударственного) планирования в условиях безбумажной информатики. Она основана на выдвинутой автором концепции новой технологии планирования, так называемой *системы Дисплан*, которая тесно увязана с изложенными выше идеями (прежде всего в § 11.9).

Основной принцип, на котором строится система,— планирование от *конечного продукта* с использованием иерархической системной оптимизации, причем вместо классического экономического подхода к динамическим макроэкономическим моделям широко используется принцип проекции динамики в статику, уже излагавшийся в § 11.9. Под конечным продуктом в *узком смысле слова* понимается все *внеэкономическое потребление*, подразделяющееся на *личное потребление* (одежда, пища, культурно-бытовые товары и т. п.), *системы коллективного пользования*, обслуживающие непосредственно население (жилищно-бытовое общественное хозяйство, медицина, общее образование, культура, средства массовой информации, общественный транспорт и др.) и системы, обслуживающие интересы всего общества в целом (оборона, охрана общественного порядка, фундаментальная наука и т. п.). В состав конечного продукта в *широком смысле слова* дополнительно включается потребление, осуществляемое всеми программами развития экономических ресурсов (производственное строительство, прикладные исследования и разработки, профессионально-техническое образование и др.).

Исходным инструментом постановки целей для программ (как национального, так и ведомственных масштабов) являются непрерывно действующие *системы прогнозирования*, составляющие первый (прогнозный) блок описываемой системы. Важнейшую роль в ней играет система *научно-технического прогнозирования*, описанная в § 11.8. Заметим сразу, что эта система (в виде, предложенном автором) дает не только цели научно-технического прогресса, но и пути их достижения. Иными слова-

ми, она является постоянным источником для формирования целевых программ. *Прогноз социального развития, внешнеполитических ситуаций*, а также *прогноз цен на мировом рынке* могут выполняться средствами кибернетического моделирования, описанными в предыдущей главе. При условии применения предложенного автором метода (§ 10.8) эти прогнозы также делаются непрерывными. К ним добавляются (также непрерывно актуализируемые) *прогноз* изменения разведанных *природных ресурсов*, а также *демографический прогноз*. Эти виды прогнозов выполняются специальными средствами, описания которых мы опускаем.

Второй блок — блок *программно-целевого управления* — включает в себя фактически все развитие как в производственной, так и в непроизводственной сферах. Наряду с крупными мероприятиями, сразу формируемыми как программы национального масштаба (БАМ, Камаз, КАТЭК и др.), на макроэкономическом уровне могут возникать *агрегированные программы*, представляющие собой объединение более мелких ведомственных и территориальных программ (с соответствующим укрупнением этапов, а также номенклатуры ресурсов и материально-технического снабжения). Будучи представлены на макроэкономическом уровне в виде укрупненных сетевых графиков, подобные программы (которые могут исчисляться многими сотнями и даже тысячами) управляются методами, описанными в § 11.6. При этом собственно ресурсное обеспечение программ замыкается в рамках рассматриваемого блока, а вектор $c' = c'_T$ материально-технического снабжения программного блока за период планирования T выдается на следующий — *балансовый блок*.

Уровень агрегации продукции при вычислении вектора c'_T соответствует агрегированной номенклатуре продукции, устанавливаемой для планового периода T в балансовом блоке. Установление списка агрегированной продукции делается так, чтобы агрегация производилась лишь применительно к тем видам близкой по своему потребительскому назначению продукции, относительные доли которых в агрегированном продукте в рассматриваемом плановом периоде не предполагается сколько-нибудь значительно изменять. В противном случае (например, при резком увеличении производства пластмассы определенного вида) соответствующая продукция, вплоть до наиболее подробно специфицированной, должна выделяться в пределах данного периода в отдельную позицию номенклатуры Госплана. Подобная *плавающая номенклатура* является еще одной особенностью, отличающей предлагаемую методику. Значение этого факта легко понять, поскольку одной из основных задач плана является как раз установление правильных пропорций производства тех или иных видов продукции, а при агрегации эти пропорции, будучи

заданы априори, фактически перестают быть управляемыми параметрами (вычисляемыми в процессе оптимизации плана).

Вектор $c' = c'_T$ представляет собой лишь одну из составляющих конечного продукта, рассматриваемого в балансовом блоке. Вторая составляющая $c'' = c''_T$ есть вектор *непрограммного* (текущего) внеэкономического потребления. В него входят, в частности, пища, одежда и другие компоненты личного потребления населения, а также *текущее потребление* систем коллективного пользования, обслуживающих население (в том числе и общество в целом). Компоненты этого вектора определяются *системами изучения потребительского спроса.* Такие системы строятся сегодня прежде всего на основе экстраполяции потребительского спроса, регистрируемого системами учета в розничной торговле и описанными в предыдущей главе системами опроса населения по выяснению будущего спроса на новые потребительские товары и услуги. Заметим, что услуги (например, перевозки пассажиров общественным транспортом или количественные показатели медицинского обслуживания) также входят в агрегированную номенклатуру, о которой говорилось выше и которая из соображений краткости называлась просто *номенклатурой продукции,* а не *номенклатурой продукции и услуг.*

В заданной агрегированной номенклатуре формируются также векторы $u = u_T$ и $v = v_T$ *экспорта* и *импорта* за рассматриваемый период T, а также вектор $\Delta_T z$ изменения (за тот же период) *государственных запасов.* По формуле

$$c_T = c'_T + c''_T + u_T - v_T + \Delta_T z \qquad (11.17)$$

определяется вектор *конечной продукции* (конечного продукта), которую надо произвести (включая услуги) в стране за данный период T в чисто производственной сфере (исключая строительство, реконструкции, опытно-конструкторские разработки и другие конечные продукты программно-целевого блока).

После отработки программно-целевого блока (осуществления проекций программ) появляется возможность рассчитать динамику нарастания ресурсов за рассматриваемый период и определить вектор $b = b_\tau$ наличного запаса ресурсо-часов в течение этого периода. При таком расчете (применительно к трудовым ресурсам) учитывается планируемая политика в области изменений продолжительности рабочего дня, недели и года (отпуски, праздничные дни и т. п.), методами, описанными в гл. X, осуществляется агрегирование нормативов (включая нормативы для новых технологий). В результате получаются матрицы $A = \|a_{ij}\|$ и $B = \|b_{ij}\|$ нормативов прямых затрат продукции и ресурсов (точнее, ресурсочасов): a_{ij} — прямые затраты i-го продукта, b_{kj} — прямые затраты ресурсо-часов k-го ресурса на производство од-

ной единицы j-го продукта. Термин «прямые» применительно к затратам означает, что учитываются лишь те затраты, которые производятся в отрасли (ведомстве), занятой непосредственно производством данного вида продукции. Эти затраты, вообще говоря, меньше *полных затрат*. Например, полные затраты электроэнергии на производство автомобиля складываются не только из прямых затрат электроэнергии непосредственно на автомобильном заводе. В скрытом виде затраты электроэнергии содержатся фактически в любой позиции материально-технического снабжения, необходимого для производства автомобиля.

Обозначая через $A^* = \|a_{ij}^*\|$ матрицу *нормативов полных затрат* продукции (a_{ij} — полные затраты i-й продукции на производство одной единицы j-й продукции), легко показать, что она вычисляется по формуле

$$A^* = (E - A)^{-1}, \tag{11.18}$$

где E — единичная матрица. Умножая на эту матрицу вектор c конечного продукта, получим вектор $x = c^*$ *полного продукта* (и услуг), который нужно произвести за плановый период T:

$$x = c^* = A^* c. \tag{11.19}$$

Полные же затраты ресурсов b^* (точнее, ресурсо-часов) на это производство выразятся формулой

$$b^* = Bc^* = BA^* c. \tag{11.20}$$

Вычитая этот вектор из вектора b, получим вектор d *невязки ресурсов* (ресурсо-часов):

$$d = b - b^*. \tag{11.21}$$

Отрицательные позиции (компоненты) в этом векторе соответствуют *дефицитам* соответствующих ресурсов (ресурсо-часов), а положительные — их *избыткам*. *Сбалансированность плана* эквивалентна отсутствию дефицитов ресурсов. Однако в предлагаемой методике предполагается, что первоначально построенный план является несбалансированным, т. е. по существу не является планом, который можно принять к исполнению. (Несбалансированность легко может возникнуть при увеличении потребления, но именно увеличение потребления и является конечной целью, во имя чего должна работать экономика.)

Программное обеспечение балансового блока, помимо формирования матриц A, B векторов b и c и вычислений по формулам (11.12)—(11.21), производит ранжирование дефицитов и избытков ресурсов по их относительным величинам и выдает по требованию *лиц, готовящих решения* (ЛГР), списки максимальных

дефицитов и избытков. В процессе системной оптимизации ведется подготовка *предложений* P_i $(i=1, 2, \ldots)$, *направленных на уменьшение максимального (относительного) дефицита*. Эти предложения могут готовиться параллельно в различных отраслях. Например, если максимальным является дефицит мощностей по производству проката черных металлов, то могут одновременно готовиться предложения по экономии проката во всех потребляющих его отраслях, предложения по увеличению производства проката на существующих мощностях, предложения по ускорению ввода новых мощностей по производству проката и, наконец, предложения по изменению соотношения экспорта и импорта проката в направлении увеличения импорта *).

Каждое предложение после отработки его на ЭВМ (в диалоговом режиме) представляется в виде $(\Delta_i A,\ \Delta_i B,\ \Delta c,\ \Delta \bar{b})$, т. е. в виде приращений, которые дает реализация этого предложения матрицам A, B и векторам c, $\bar{b}$ (некоторые из приращений могут быть нулевыми). Индекс i приращений $\Delta_i A$ и $\Delta_i B$ подчеркивает, что благодаря отраслевому характеру предложений (соответствующему специализации ЛГР) изменения нормативов в результате любого подготовленного предложения будут затрагивать лишь одну отрасль (технологию) или, иными словами, один столбец матриц A и B (в данном случае i-й). Для таких одностолбцовых приращений имеет место формула

$$\Delta_i A^* = \frac{1}{1-\alpha_i} D_i A^*, \quad D_i = \| d_{ij} \| = A^* \Delta_i A, \quad \alpha_i = d_{ii}. \tag{11.22}$$

По этой формуле вычисление $\Delta_i A^*$ требует выполнения лишь $3n^2$ арифметических операций (n — размерность матрицы A) вместо порядка n^3 операций, которые требует обращение матрицы $(E - A + \Delta_i A)$ обычными методами.

Новый план выпуска $c^* + \Delta c^*$ может быть теперь получен по формуле

$$c^* + \Delta c^* = (A^* + \Delta_i A^*)(c + \Delta c), \tag{11.23}$$

требующей порядка $3n^2$ арифметических операций, а новый вектор $d + \Delta d$ невязки ресурсов $d + \Delta d$ — по формуле

$$d + \Delta d = b + \Delta b - (B + \Delta B)(c^* + \Delta c^*), \tag{11.24}$$

по которой вычисление требует порядка $2mn$ арифметических операций (m — число агрегированных ресурсов). К этому надо добавить n операций для вычисления матрицы $A + \Delta_i A$, чтобы получить полностью пересчитанные величины, необходимые для следующего тура расчетов. Таким образом, полный цикл работы

*) Для сохранения внешнеторгового баланса последние предложения должны предусматривать компенсацию увеличения импорта проката за счет увеличения экспорта другой продукции.

по обсчету одного предложения занимает порядка

$$N = 6n^2 + 2mn \qquad (11.25)$$

арифметических операций, что при больших m и n значительно экономит время для обсчета поступающих предложений по сравнению с прямыми расчетами измененного плана, когда необходимо затрачивать порядка n^3 операций.

Даже при $m = n = 10\,000$ формула (11.25) дает $N = 0{,}8 \cdot 10^9$. ЭВМ с быстродействием в 1 млн. операций в секунду выполнит такой объем вычислений за 800 секунд или приблизительно за 13 минут (против 2 недель по традиционному методу).

Правда, благодаря «дифференциальному» характеру описанного метода (т. е. работы по приращениям) при большом числе обрабатываемых предложений возможно нежелательное накопление ошибок. Поэтому время от времени (после окончания обработки достаточно большого потока предложений) можно производить контрольные пересчеты плана традиционным методом (с учетом принятых предложений).

Процесс принятия или отклонения очередного предложения прост: если предложение уменьшает максимальный относительный дефицит, оно принимается, в противном случае — отклоняется. ЭВМ должна при этом вести архив как принятых, так и отклоненных предложений. Имея такие архивы, в любой момент можно отказаться от принятого ранее предложения $P_i = (\Delta_i A, \Delta_i B, \Delta c, \Delta \bar{b})$, произведя «принятие» противоположного предложения $-P_i = (-\Delta_i A, -\Delta_i B, -\Delta_i c, -\Delta_i \bar{b})$. Ввиду того, что в общем случае мы имеем дело с нелинейной задачей оптимизации, подобные возвраты назад в принципе не исключены: хотя в предшествующей ситуации принятие предложения P_i способствовало уменьшению максимального дефицита, в новых условиях (после принятия ряда других предложений) подобный же эффект может дать отказ от предложения P_i (т. е. принятие предложения $-P_i$). Аналогично, отклоненное ранее предложение в новых условиях может оказаться полезным и может быть принято (для чего и нужен архив отклоненных предложений). Разумеется, в линейном случае подобные изменения ценности предложений в процессе оптимизации не имеют места, так что раз принятые предложения не отбрасываются, а раз отвергнутые — не принимаются при продолжении процесса. Заметим, что с линейным случаем мы имеем дело, когда все рассматриваемые предложения имеют либо вид $(\Delta_i A, \Delta_i B, 0, 0)$, либо $(0, 0, \Delta c, \Delta b)$.

Если в описанном процессе удается полностью ликвидировать дефициты, то мы получаем сбалансированный *оптимизированный план* (но не обязательно оптимальный в строго математическом смысле), который может быть в принципе принят. Обычно, однако, полезно продолжить процесс оптимизации, загружая избы-

точные ресурсы и тем самым либо увеличивая потребление, либо улучшая сальдо внешнеторгового баланса (увеличивая валютные поступления от внешней торговли). Если же в процессе оптимизации полностью ликвидировать дефициты не удается, то, наоборот, производится «отключение» тех или иных блоков от вектора потребления или экспорта до достижения полной сбалансированности плана. Разумеется, подобные отключения делаются лишь по разрешению ЛПР соответствующего ранга.

Описанному процессу системной оптимизации может помочь дополнительный процесс агрегации планов, составленных отраслями, в общий народнохозяйственный план x. Разность векторов x и c^*:

$$\delta = x - c^* = x - A^*c \qquad (11.26)$$

представляет собой *невязку планов*, по которой можно также производить операции постепенного уничтожения невязок (в диалоге с отраслями). Однако подобный процесс не дает столь глубоких возможностей управления научно-техническим прогрессом (а следовательно, и оптимизации плана), как уже описанный выше процесс. Поэтому вектор невязок планов δ целесообразнее использовать в качестве дополнительной информации для ЛПР в целях лучшей ориентировки в подготовке предложений по улучшению планов.

В рамки описанной идеологии укладываются и вопросы денежного баланса (приходов и расходов населения); с этой целью в общем балансе учитываются все расходы на денежные выплаты населению (зарплата, пенсии и т. д.) в качестве одной из компонент затрат во всех производственных и непроизводственных отраслях. В качестве же производимой продукции деньги учитываются прежде всего в планировании розничной торговли, а также всех других отраслей, производящих платные услуги (общественный транспорт, жилищно-бытовое хозяйство, кино, театры и т. п.). Все эти денежные поступления могут быть получены простым перемножением вектора товаров и услуг, поставляемых населению (части вектора c''), на вектор средних розничных цен на рассматриваемый плановый период. Для управления денежным балансом, помимо уже упомянутого вектора невязок ресурсов, рассматривается еще один дополнительный скалярный показатель — сальдо денежного баланса. В случае наличия дефицита (отрицательного сальдо), точно так же как и выше, рассматриваются различные предложения по его уменьшению и ликвидации. В число таких предложений наряду с увеличением выпуска и продажи потребительских товаров и услуг могут включаться также предложения по изменению уровня розничных цен. Разумеется, вопрос о принятии таких предложений должен решаться ЛПР соответствующего (высокого) уровня.

Заметим, что при подведении основного баланса (нахождения вектора невязки ресурсов) единицы измерения для различных видов продукции могут выбираться любыми. В частности, некоторая (а если надо, то и вся) продукция может исчисляться в денежном выражении. Для стран, ведущих активную внешнюю торговлю, может оказать немалую пользу возможность пересчета плана в стоимостном выражении в соответствии с уровнем цен на мировом рынке. Такой пересчет делает ясной картину возможностей «производства» валюты различными отраслями.

Скажем несколько слов о планировании транспорта. При составлении вариантов планов отдельными предприятиями, как правило, должны предусматриваться пункты доставки производимой ими продукции. Таким образом, вместе с объектами выпуска продукции в исходных планах появляются и объемы предстоящих перевозок, по возможности разбитые по видам транспортных средств, которые предполагается использовать. Транспортные ведомства, ответственные за развитие и эксплуатацию соответствующих средств, разрабатывают нормативы материально-технического и ресурсного обеспечения перевозок различных видов, которые в агрегированном виде должны включаться в конце концов в матрицы A и B. После этого программы развития транспортных ресурсов и невязки по этим ресурсам включаются в рассмотренную выше схему системной оптимизации наравне с другими товарами и услугами.

Аналогичным образом можно включить в общую схему и планы развития и использования систем связи и управления для производственных нужд. Что же касается планов удовлетворения потребности населения в средствах связи, то соответствующие потребности (и плановые задания на их удовлетворение) должны изучаться в основном теми же приемами, что и потребности в других услугах и в продуктах. Напомним, что соответствующие задания должны включаться в правые части балансовых уравнений (векторы c' и c'').

После выполнения балансировки и оптимизации по всему плановому периоду T те же самые процедуры используются для отдельных его подпериодов $T_1, T_2, \ldots, T_c$. Как и в общей схеме объемно-календарного планирования (изложенной в § 11.9), особенно просто такая разбивка делается при непрерывном планировании. В этом случае задача *пролонгирования плана* с периода $[0, T]$ на период $[\Delta T, T+\Delta T]$ сводится по существу к выполнению описанной выше процедуры балансировки и системной оптимизации для периода $[T, T+\Delta T]$ с возможными корректировками для периода $[\Delta T, T]$ (относительно небольшими).

Сделаем несколько замечаний об отраслевом и территориальном разрезе плана. При вычислении матриц A, B векторов $x=c^*$, δ и b^* в исходном (общегосударственном) балансе про-

дукция, ресурсы группируются по естественным технологическим группам — *условным отраслям* (например, токарные станки, литейное оборудование, телевизоры, мясные консервы и т. п.). В действительности как оборудование, так и сама производимая продукция должны быть разбиты не по условным, а по реально существующим *юридическим узаконенным отраслям* (министерствам и ведомствам). С этой целью вектор b наличного запаса ресурсо-часов должен быть разбит по юридическим отраслям: $b = b^1 + b^2 + \ldots + b^f$. После этого первое (грубое) разбиение плана $c^* = x$ по юридическим отраслям может быть получено (методами, изложенными в § 8.6) из решения системы линейных неравенств $Bx^i \leqslant b^i$ $(i = 1, 2, \ldots, f)$ с дополнительным условием

$$\sum_{i=1}^{f} x^i = x, \tag{11.27}$$

где x^i есть вектор плана производства для i-го министерства или ведомства. Эта система неравенств при условии сбалансированности исходного (общегосударственного) плана совместна (поскольку $B\Sigma x^i = Bx \leqslant b = \Sigma b_i$). Однако следует учесть, что матрицы нормативов прямых затрат ресурсов B_i для каждой юридической отрасли могут быть различными. Поэтому вместо неравенств (11.27) в действительности имеют место уточненные неравенства

$$B^i x^i \leqslant b^i, \quad i = 1, 2, \ldots, f, \tag{11.28}$$

$$\sum_{i=1}^{f} x_i = x.$$

Если бы агрегация нормативов прямых затрат ресурсов в матрицу B производилась через юридические отрасли в *соответствии с пропорциями деления планов между ними*, то система неравенств (11.28) также должна была бы быть совместной. Однако в действительности точные значения этих пропорций априори (до решения неравенств (11.28)) неизвестны. Поэтому действительная разбивка планов по отраслям должна претерпеть ряд итераций: отправляясь от системы (11.27), находим первое приближение для разбивки планов. В соответствии с пропорциями, определенными этой разбивкой, вычисляем новые значения «отраслевых» матриц прямых затрат ресурсов, находим из (11.28) новую (уточненную) разбивку планов и т. д.

После получения разбивки плана по отраслям можно получить уточненные отраслевые матрицы A^i — нормативы прямых затрат продукции, получаемой извне в качестве материально-технического снабжения. Тогда соответствующие отраслевые векторы a^i материально-технического снабжения получаются по

формулам

$$a^i = A^i x^i, \quad i = 1, 2, \ldots, f. \tag{11.29}$$

Тем же способом, что и разбивка плана по отраслям, может быть осуществлена его разбивка по территориям. При этом появляется дополнительная возможность оценить перетоки продукции с одних территорий на другие и тем самым уточнить транспортные балансы. Заметим, что для планирования по любой территории годится в исходном виде весь описанный метод целостного (без разбивки по отраслям и территориям) метода общегосударственного планирования, причем место экспорта и импорта соответственно занимают вывоз и ввоз на данную территорию продукции с других территорий.

Сделаем еще одно замечание, касающееся планов развития экономики (инвестиций). В рамках программно-целевого метода в процессе составления программ попутно производится разбивка планируемых работ по отраслям и территориям, так что территориально-отраслевая разбивка этой части планов по существу сводится лишь к переупорядочиванию информации, уже имеющейся в общегосударственных планах.

При планировании на всех уровнях, в том числе и на общегосударственном, должны быть предусмотрены *резервы* на случай различных непредвиденных обстоятельств. Эти резервы могут быть в форме страховых запасов, методы расчета которых были изложены выше (в § 11.5). Другая форма — создание *резерва мощностей* — используется в том случае, когда делать запасы продукции невозможно (как это имеет место в электроэнергетике), трудно (в газоснабжении) или экономически невыгодно (в случае скоропортящейся или быстро меняющейся продукции). Размер резерва определяется минимизацией суммы потерь от недоиспользования резервных мощностей для производства продукции и математического ожидания суммарных потерь в народном хозяйстве в результате случайных сбоев в работе отдельных звеньев экономики, которые невозможно устранить при выбранной величине резерва.

11.10.1. Предплановые ориентировки. Как уже упоминалось выше, для предварительной ориентации ЛПР могут использоваться различного рода загрубленные макроэкономические модели. Чаще всего загрубление предполагает использование моделей малой размерности. Другие направления загрубления — использование тех или иных априорных предположений о динамике изменения нормативов, переход от многокритериальных постановок к однокритериальным и т. д. В то время как в реальном планировании речь, как правило, идет о выборе между конечным множеством *реально подготовленных проектов*, в предплановых ориентировках допустимы абстрактные постановки с не-

прерывно меняющимися параметрами. В линейных моделях такого рода получающиеся решения обычно требуют полного «закрытия» одних технологий и максимального использования других, что в реальной жизни сделать сразу невозможно. Тем не менее даже такие решения, указывая желательные направления «переключения» технологий, оказываются полезными как качественные ориентировки при формировании реальных предложений по улучшению плана.

Например, имея набор технологий и соответствующие им матрицы A и B прямых затрат продуктов и ресурсов, можно решить задачу об ориентировке экономики в направлении получения наибольшего конечного продукта C в денежном выражении. Для этого решается задача линейного программирования

$$\max p(E-A)x \quad \text{при} \quad Bx \leqslant b,\ x \geqslant 0,$$

где $p = \|p_1, \ldots, p_n\|$ — вектор цен продукции различных видов, предусмотренных в плане $x = \left\| \begin{matrix} x_1 \\ \cdots \\ x_n \end{matrix} \right\|$. Как это обычно бывает при линейных постановках задач оптимизации, решение полученной задачи разделит продукцию на выгодную ($x_i > 0$) и невыгодную ($x_i = 0$), хотя, разумеется, в силу большого числа причин, в реальной жизни нельзя принять предлагаемый план x сразу, поскольку он связан с закрытием невыгодных производств с соответствующим перемещением и переквалификацией рабочих, что, естественно, быстро осуществить невозможно (особенно при необходимости перемещений рабочей силы в другие районы страны).

Однако, получив подобную ориентировку, ЛПР получают «начальный импульс» для выработки реальных предложений, изменяющих структуру плана в желательном направлении. Подобная ориентировка особенно полезна для решения задач специализации экономики малых стран в рамках международного разделения труда. В этом случае оптимизация с использованием вектора p цен на мировом рынке определяет те отрасли национальной экономики ($x_i > 0$), которые наиболее выгодно развивать в данной стране для максимального увеличения положительного сальдо ее внешней торговли.

В качестве другого примера рассмотрим предложенную автором модель для оптимизации распределения национального дохода между потреблением и накоплением (инвестициями) в условиях научно-технического прогресса. С этой целью разделим экономику, как это принято в политэкономии, на группу А (производство средств производства) и группу Б (производство предметов потребления). Подчеркнем сразу, что в группу А включается не вся тяжелая промышленность (как это нередко делается) и даже не все машиностроение. Например, производ-

ство легковых автомобилей, идущих в личное потребление, вместе со всем его обеспечением (металл, пластмассы, лаки и т. п.) должно быть включено в группу Б. Сюда же включается и вся продукция оборонной промышленности, идущая по ее прямому назначению (разумеется, также со всем обеспечением). Аналогично обстоит дело со строительством. Непроизводственное строительство (жилые дома, культурно-бытовые объекты, больницы и т. п.) относится по нашей классификации к группе Б, а производственное (независимо от вида строящихся объектов) — к группе А. Кроме того, при вводе в строй новых производственных объектов их необходимо снабжать *оборотными фондами*, которые могут быть предметами потребления, но выступают в данном случае в качестве средств производства.

Имея в виду перечисленные обстоятельства, на самом высоком уровне агрегации экономика может быть (с определенными допущениями) сведена фактически к одной обобщенной отрасли, одна часть которой работает в *режиме группы* А (производит средства производства, т. е. основные и оборотные фонды новых производственных объектов), а другая — в *режиме группы* Б (производит предметы текущего потребления). Допущения, о которых идет речь, состоят в том, что предполагается возможность беспрепятственного (мгновенного и без дополнительных затрат) переключения действующих технологий (прежде всего в строительстве и машиностроении) с производства средств производства (например, тракторов) на производство предметов потребления (например, танков) и наоборот. Разумеется, подобное допущение в полной мере никогда не имеет места на практике. Однако, имея в виду высокую степень агрегации модели, ее направленность па качественные ориентировки, а также относительную малость затрат на переориентацию существующих производственных мощностей по сравнению с затратами на их начальное создание, можно считать сделанное допущение достаточно приемлемым. Его приемлемость становится еще более обоснованной, если учесть, что обеспечивающие отрасли (например, топливная промышленность или электроэнергетика) в большинстве случаев вообще не требуют никаких переделок при переключении их продукции с режима А на режим Б и наоборот.

Введем две функции $\alpha(\tau, t)$ и $\beta(t)$, определяющие производительность труда в группах (режимах работы) А и Б в различных технологиях. Технологии мы будем характеризовать моментами их разработки до такой степени, при которой возможны прямые инвестиции для фактического их внедрения в народное хозяйство. Сделаем естественное предположение, что позже созданные технологии лучше (обеспечивают более высокую производительность труда) по сравнению с технологиями, созданными ранее.

Функция $\alpha(\tau, t)$ $(\tau < t)$ равна количеству рабочих мест в технологии в момент времени t, в единицу времени в расчете на одного работающего в группе А в момент времени τ. Функция $\beta(t)$ означает выпуск продукции в единицу времени в группе Б, приходящийся на одного работающего, по технологии в момент времени t. Будем считать, что в один и тот же момент времени не создается двух различных технологий и что в промежутке между созданием двух очередных технологий в каждой из групп А и Б внедряется первая из них. Функции $\alpha(\tau, t)$ и $\beta(t)$ предполагаются *известными* не только для прошлого, но и для *будущего* времени. В последнем случае они исчисляются из прогнозных сроков создания новых технологий (см. § 11.9) и соответствующим образом агрегированных нормативов трудовых затрат (в результате вклинивания новой технологии).

Обозначим через $x(\tau, t)$ и $y(\tau, t)$ (где $\tau \leqslant t$) функции интенсивности использования (коэффициента сменности) в момент времени t рабочих мест в момент времени τ соответственно в А и Б. Эти функции предполагаются *заданными* для прошлого времени $(t \leqslant 0)$ и *неизвестными* для будущего $(t > 0)$. Заданы лишь границы изменения этих функций в будущем (политика в области коэффициентов сменности): $0 \leqslant x(\tau, t) \leqslant k_1(t)$, $0 \leqslant y(\tau, t) \leqslant k_2(t)$, где $k_1(t)$ и $k_2(t)$ — предполагаемые максимальные коэффициенты сменности в момент времени t (соответственно в А и Б). Через $n(t)$ обозначим общее количество рабочих, которым располагало (при $t \leqslant 0$) и будет располагать (при $t > 0$) народное хозяйство, а через $[0, T]$ — предполагаемый интервал долгосрочного планирования. Задача состоит в таком распределении рабочих в каждый момент планового периода между различными местами в группах А и Б, при котором обеспечивается максимальный объем выпуска продукции группы Б (за плановый период) при условии полной занятости всех рабочих.

Обозначим через $m(t)$ *скорость* создания новых рабочих мест в экономике в момент времени t (способных по нашему допущению работать в двух режимах А и Б). Через $t_0 < 0$ будем обозначать момент создания наиболее старого из рабочих мест, существующих к началу планового периода $(t = 0)$. Тогда, как нетрудно проверить, будут иметь место следующие соотношения:

$$m(t) = \int_{-t_0}^{t} x(\tau, t)\,\alpha(\tau, t)\,m(\tau)\,d\tau$$

(уравнение баланса производственных мощностей);

$$n(t) = \int_{-t_0}^{t} (x(\tau, t) + y(\tau, t))\,m(\tau)\,d\tau$$

(уравнение баланса трудовых ресурсов);

$$f(t) = \int_{-t_0}^{t} y(\tau, t)\,\beta(\tau)\,m(\tau)\,d\tau$$

(уравнение скорости выпуска предметов потребления).

В соответствии с приведенной выше постановкой, задача состоит в нахождении функций $x(\tau, t)$, $y(\tau, t)$, $m(t)$ в интервале $[0, T]$ из условия максимума функционала $J(f) = \int_0^T f(t)\,dt$ при дополнительных ограничениях $0 \leqslant x(\tau, t) \leqslant k_1(t)$, $0 \leqslant y(\tau, t) \leqslant k_2(t)$, $0 \leqslant t \leqslant T$. Отношение

$$z(t) = \left(\int_{-t_0}^{t} x(\tau, t)\,m(\tau)\,d\tau\right) \Big/ \left(\int_{-t_0}^{t} y(\tau, t)\,m(\tau)\,d\tau\right)$$

характеризует пропорцию деления национального дохода (в момент времени t) между накоплением (группа А) и потреблением (группа Б).

В другой постановке предполагается априорное разделение рабочих мест на группы А и Б. Относительная доля рабочих мест в группе А ко всем рабочим местам, создаваемым в момент времени τ, обозначается через $y(\tau)$. Коэффициент сменности $k(t)$ для всех технологий, созданных в моменты времени между $a(t) < t$ и t, предполагается не зависящим от технологии (а зависящим лишь от момента времени t). Все технологии, созданные до момента времени $a(t)$, в момент времени t не используются вовсе. Таким образом, $a(t)$ задает переменную временну́ю границу отбрасывания устаревших технологий. При сделанных предположениях приведенные выше уравнения примут вид

$$m(t) = \int_{a(t)}^{t} \alpha(\tau, t)\,y(\tau)\,m(\tau)\,d\tau,$$

$$\frac{n(t)}{k(t)} = \int_{a(t)}^{t} m(\tau)\,d\tau,$$

$$f(t) = \int_{a(t)}^{t} (1 - y(\tau))\,m(\tau)\,\beta(\tau)\,d\tau,$$

где требуется найти неизвестные функции $m(t)$ и $y(t)$ $(0 \leqslant y(t) \leqslant 1)$ в плановом интервале $[0, T]$ при условии максимума функционала

$$J(f) = \int_0^T f(t)\,dt.$$

В отличие от прежней постановки, определяющей политику использования уже созданных *универсальных* мощностей, вторая постановка определяет политику разделения средств в создании новых *специализированных* мощностей в группах А и Б. Следует заметить, что в обеих постановках принятый критерий оптимизации приводит к эффекту минимизации задела на последующие плановые периоды, т. е. к снижению темпов инвестиций к концу рассматриваемого планового периода (в целом или в группе А). Поэтому правильную инвестиционную политику предполагаемая методика дает лишь для части планового периода от $t=0$ до $t=T_0<T$. В связи с этим необходимо все время корректировать эту политику на последующие моменты в режиме непрерывного планирования (прогнозирования конечного срока T).

11.11. Общегосударственная автоматизированная система (ОГАС)

Автоматизация организационного управления в масштабах всего народного хозяйства требует целенаправленной разработки и последующего целенаправленного *взаимодействия* автоматизированных систем организационного управления (АСОУ) различных уровней и различного назначения. В соответствии с действующим у нас территориально-отраслевым принципом управления создаются как отраслевые, так и территориальные АСОУ. В территориальном разрезе АСОУ создаются для отдельных районов, городов, областей и республик. В отраслевом разрезе соответствующая иерархия включает в себя АСОУ цеха, предприятия, объединения, отрасли (иногда с особыми АСОУ для отдельных главков). На самом верхнем уровне иерархии территориальный и отраслевой аспекты объединяются в общегосударственных органах планирования и управления, включая функциональные союзные министерства и ведомства (Госплан, Госснаб, ЦСУ, Минфин, Госкомтруд, Госкомцен, ГКНТ и др.).

В решениях 24-го съезда КПСС ОГАС определена как *Общегосударственная автоматизированная система сбора и обработки информации для учета, планирования и управления*. Функции автоматизации учета на всех уровнях (включая общегосударственный) были достаточно подробно разобраны выше (в § 7.7 и в гл. X). Заметим лишь то, что система учета по уровням и номенклатура агрегирования учетных данных должны быть увязаны с системой планирования. Ибо основная задача учета в системах организационного управления — отслеживать процесс выполнения плана и своевременно информировать соответствующие уровни управления о возникающих отклонениях. Основная задача *системы текущего (оперативного) управления* — своевре-

менно реагировать на эти отклонения. В правильно спроектированной системе большинство отклонений должно компенсироваться за счет приведения в действие резервов того или иного уровня. При этом сбой как бы локализуется, т. е. остается в рамках породившего его звена экономики, не приводя к «цепной реакции» распространения возникшего сбоя по всем связанным с ним звеньям экономики (по цепочкам поставщики — потребители). В тех относительно редких случаях, когда локализовать сбой не удается, производится *согласованный* пересчет планов во всех указанных звеньях.

Помимо учета и текущего управления главной задачей *вертикальных связей* в ОГАС является обеспечение системы объемно-календарного территориально-отраслевого планирования во всех звеньях экономики (от Госплана СССР до цеха, участка, а в краткосрочном планировании и до отдельных рабочих мест). Эта система фактически уже была описана выше в двух предшествующих параграфах, так что остается сделать лишь несколько дополнительных замечаний. Прежде всего ясно, что формирование заданий по конечному продукту в части удовлетворения потребностей населения должно вестись прежде всего в территориальном аспекте с учетом специфических особенностей населения различных районов страны. Начиная с районных и городских АСОУ, должны вестись локальные программы жилищного, культурно-бытового строительства, развития общественного транспорта, торговли, медицины и других систем коллективного обслуживания населения. Смысл вертикальных связей ОГАС в этом аспекте состоит в том, чтобы обеспечить *интеграцию локальных программ* по всем уровням иерархии территориального управления, вплоть до общесоюзного уровня. С этой целью обеспечивается динамическое представление программ в памяти ЭВМ и их автоматическое агрегирование по требованиям, задаваемым органами территориального управления и Госпланом СССР. Аппаратом для такой агрегации является Государственная сеть вычислительных центров (ГСВЦ), о которой уже говорилось выше (в § 7.7).

Заметим, что программы развития территорий должны быть тесно увязаны с программами развития отраслей. С этой целью варианты отраслевых программ после их предварительной территориальной привязки передаются через ГСВЦ в АСОУ соответствующих территорий, которые должны увязывать с ними соответствующие территориальные программы. В процессе оптимизации планов, описанном в предыдущем параграфе, возможно (и желательно) возникновение предложений по изменениям дислокации предполагаемого производственного строительства. Задачей ОГАС является автоматизация взаимодействия органов территориального и отраслевого управления (через соответству-

ющие АСОУ) при подготовке подобных предложений, с тем чтобы планы отраслевого и территориального развития оставались постоянно взаимоувязанными.

При уменьшении заданий по конечному продукту, возможному на этапе окончательной балансировки оптимизационных общесоюзных планов, ОГАС обеспечивает соответствующее взаимодействие с программами отраслевого и территориального развития, чтобы производить сбалансированные отключения (или переносы сроков) отдельных программных блоков, что призвано сделать невозможным такое положение, когда, например, предусматривалось бы жилищное строительство, не обеспеченное соответствующим расширением торговой сети, развитием общественного транспорта и т. п.

В отраслевом разрезе ОГАС обеспечивает описанные выше (в гл. X и в §§ 11.9, 11.10) процессы агрегации планов, нормативов и данных о наличных и планируемых ресурсах, а также процесс системной оптимизации планов на основе начального формирования и последующей агрегации предложений, направленных на улучшение плана. О последнем процессе нужно сказать несколько слов особо. Может оказаться, что в процессе работы по уменьшению дефицитов общесоюзного плана (описанного в § 11.10) родилось предложение, носящее первоначально локальный характер, например предложение рационализатора об улучшении технологического процесса на своем рабочем месте. Система агрегации предложений в ОГАС должна предусматривать последовательный подъем этого предложения по уровням иерархии планирования на предмет его возможного использования на других рабочих местах (данного предприятия, отрасли и даже других отраслей). В случае, когда в процессе такого подъема предложение обретает должный масштаб, оно может в конце концов выйти на общесоюзный уровень системной оптимизации в качестве предложения, привязанного к соответствующим отраслям, предприятиям и рабочим местам. В результате происходит глубокая *демократизация* процесса общегосударственного планирования (т. е. обеспечения участия в нем всех, кто способен вложить свою лепту в улучшение плана) и диалектическое разрешение противоречия между необходимостью дальнейшего развития *централизации планирования* и полного простора для личной *инициативы*. Принципиально важно, что процесс *встречного планирования* производится при этом не после составления общегосударственного плана (что чревато возможностью возникновения опасных дисбалансов), а *в процессе его составления*.

Наличие в ОГАС взаимосвязанных автоматизированных систем планирования различных уровней обеспечивает доведение планов до рабочих мест (включая любые последующие измене-

ния в них), что является непременным условием *действенности* планов. Всякий план, не доведенный до рабочих мест, является скорее набором пожеланий, чем руководством к конкретным целенаправленным и взаимосогласованным действиям (обеспеченным к тому же всем необходимым для их реализации).

Сказанное до сих пор касалось прежде всего вертикальных связей в ОГАС, т. е. связей, направленных (вверх и вниз) по линиям фактически существующей административной иерархии. Таких связей оказывается, однако, недостаточно для эффективного управления экономикой. Важнейшую роль в таком управлении играют *горизонтальные* (*межведомственные*) *связи*. Дело заключается прежде всего в том, что при реализации одних лишь вертикальных связей остается еще много нерешенных проблем по организации эффективного взаимодействия *потребителей с поставщиками*.

Действительно, в описанном выше (в § 11.10) процессе общесоюзного объемно-календарного планирования производится разбивка планов по отраслям и территориям (включая планы материально-технического снабжения). Однако на общесоюзном уровне эти планы носят по необходимости агрегированный характер и привязываются к относительно большим промежуткам времени (годам, кварталам, в лучшем случае — к месяцам). Спецификация взаимных поставок (до самой подробной номенклатуры) и точное (минимум до суток) указание их сроков возможны лишь при организации совместной работы АСОУ поставщиков и потребителей на различных уровнях.

Получив от Госплана планы выпуска продукции на тот или иной плановый период (в результате *совместной* работы с ним по вертикальным связям), отрасли должны организовать работу по детализации (спецификации) планов взаимных поставок. С этой целью АСОУ отрасли, выпускающей ту или иную *финишную* (законченную) продукцию (а не полуфабрикаты), специфицирует вектор материально-технического снабжения, обеспечивающий этот выпуск, и в процессе взаимодействия с АСОУ других отраслей производит распределение поставок по соответствующим отраслям (в соответствии со спущенными из Госплана агрегированными планами производства). Кроме того, имея план распределения выпуска продукции по своим предприятиям (или объединениям) и используя (в процессе взаимодействия АСОУ) аналогичные планы других отраслей, АСОУ отрасли, инициирующей взаимодействие, решает в процессе такого взаимодействия задачу прикрепления предприятий-потребителей (своей отрасли) к предприятиям-поставщикам (своей и других отраслей).

В целях оптимизации перевозок для прикрепления потребителей к поставщикам решается соответствующая *транспортная*

задача (см. п. 8.6.2). В случае возможностей «переброски» отдельных частей планов между предприятиями-поставщиками (лучшей специализации предприятий с учетом их собственных расходов и расходов на перевозки) возникает более общая, так называемая *распределительная оптимизационная задача.* Для ее решения применяются общие методы решения задач линейного программирования, а также некоторые более специальные приемы, сводящие эти задачи к нелинейным задачам меньшей размерности. Заметим, что планы прикрепления потребителей к поставщикам (как и все другие планы) должны храниться в соответствующих АСОУ в динамическом виде и могут подвергаться *согласованным* изменениям в процессе системной оптимизации общегосударственных планов.

Необходимо отметить также и то, что описанный простейший (поотраслевой) подход к решению задачи прикрепления потребителей к поставщикам не обеспечивает его полной оптимизации. Более полная оптимизация может осуществляться АСОУ такого межведомственного органа, как Госснаб. Собирая заявки по материально-техническому снабжению от всех отраслей, этот орган должен решать распределительные задачи, фиксируя прикрепления предприятий-потребителей к тем или иным предприятиям-поставщикам.

Рассмотрим одну из возможных постановок распределительных задач и возможные методы их решения в этой постановке. Обозначим через i индекс (номер) потребителя (предприятия или оптовой базы Госснаба); через j — индекс поставщика (производителя); через k — индекс поставляемой продукции; через α — индекс, пробегающий множество технологических ограничений на производство у поставщиков. Пусть, далее, A_{ik} — потребность (в рассматриваемом плановом периоде) i-го потребителя в k-й продукции; c_{ijk} — стоимость перевозки одной единицы k-й продукции от j-го поставщика к i-му потребителю; t_{jk}^{α} — технологические коэффициенты у j-го поставщика на производство k-й продукции; c_{jk} — себестоимость производства одной единицы k-й продукции.

Обозначим через $x = \|x_{ijk}\|$ план прикрепления потребителей к поставщикам. Здесь x_{ijk} означает количество k-й продукции, которую i-й потребитель должен получить (в течение данного планового периода) от j-го поставщика. Ясно, что $x_{ijk} \geqslant 0$. Тогда распределительная задача представляет собой задачу линейного программирования:

$$\min\left(\sum_{(i,j,k)} c_{ijk}x_{ijk} + \sum_{(i,j,k)} \alpha_{jk}x_{ijk}\right),$$

$$\sum_{(i,k)} t_{jk}^{(\alpha)}x_{ijk} \leqslant T_j^{(\alpha)} \quad \text{(для всех } j \text{ и } \alpha\text{)}.$$

При $\alpha = 1$ $T_j^{(1)}$ есть общее время (с учетом необходимого резерва), которым располагает j-й поставщик; $t_{jk}^{(1)}$ — время, затрачиваемое им на производство одной единицы k-й продукции. При $d > 1$ возможны любые другие типы ограничений; например, ограничение $\sum\limits_i x_{ijk} \leqslant T_j^{(k)}$ означает, что j-й поставщик не может выпустить более чем $T_j^{(k)}$ единиц k-й продукции (здесь $t_{jk} = 1$, $T_{jl} = 0$ при $l \neq k$).

Во многих случаях на практике получаемая задача линейного программирования имеет столь большую размерность, что обычные методы ее решения оказываются непригодными. Например, при прикреплении потребителей к прокатным станам для поставок проката черных металлов в СССР число переменных x_{ijk} достигает $5 \cdot 10^5$, а число ограничений — $2 \cdot 10^4$. Поэтому сведем задачу к задаче нелинейного выпуклого программирования:

$$\max_{\{u_j^{(\alpha)} \geqslant 0\}} \left(\sum_{i,k} A_{ik} \min_j \left(c_{ijk} + \alpha_{jk} + \sum_\alpha t_{ik}^{(\alpha)} u_j^{(\alpha)} \right) - \sum_{j,\alpha} T_j^{(\alpha)} u_j^{(\alpha)} \right).$$

Здесь число «технологических» (двойственных) переменных $u_j^{(\alpha)}$ обычно значительно меньше (в случае задачи о прикреплении к прокатным станам — не более 200—300). Используя обобщенные градиентные методы с растяжением пространства, эту задачу удается решить с относительной точностью 10^{-6} по функционалу за несколько сотен шагов (в случае прокатных станов — за 600—800). Далее, по формулам

$$v_{ik} = \min_j \left(c_{ijk} + \alpha_{jk} + \sum_\alpha t_{ijk}^{(\alpha)} u_j^{(\alpha)} \right)$$

находим величины v_{ik}. В случае, когда в правой части минимум с некоторой (заданной априори) точностью ε достигается для о д н о г о значения j, имеет место формула $x_{ijk} = A_{ik}$ (т. е. заказ A_{ik} прикрепляется полностью к j-му поставщику). Если минимум достигается на нескольких значениях $j = j_1, j_2, \ldots, j_k$, то заказ A_{ik} следует распределить между поставщиками с номерами $j_1, j_2, \ldots, j_s$. Для подобных заказов, которых на практике бывает обычно немного, распределительную задачу можно решить обычными методами линейного программирования, поскольку размерность ее, как правило, значительно меньше размерности исходной задачи. Возможно также применить прием (аналогичный уже примененному приему) сведения этой новой задачи линейного программирования к задаче нелинейного (в данном случае квадратичного) программирования значительно меньшей размерности. В конечном счете в результате указанных вычислений получится объемно-календарный план прикрепления потребителей к поставщикам.

После получения планов такого прикрепления в процессе взаимодействия АСОУ предприятий (потребителей и поставщиков) и транспортных организаций решаются вопросы о точной спецификации поставок, определении размеров партий и точного времени их доставки потребителям. Для решения этих задач в первом приближении могут употребляться описанные выше (в § 11.5) методы управления запасами. Критерием для оптимизации при этом может служить минимизация суммарных потерь у поставщиков, потребителей и транспортных организаций по сравнению с оптимальными планами производства, поставок и перевозок, составленными по раздельным для каждого предприятия и каждой транспортной организации критериям. Как уже отмечалось выше, буферные запасы при этом лучше всего создавать на предприятиях-поставщиках. Запасы на существующих территориальных базах Госснаба можно было бы тогда использовать в качестве общегосударственных резервных (страховых) запасов. Заметим, что наличие таких баз вносит дополнительные усложнения в решение распределительной задачи, поскольку подобные базы могут выступать одновременно и в качестве потребителей, и в качестве поставщиков.

Само собой разумеется, что вся перечисленная система планов взаимных поставок по горизонтальным связям должна автоматически и притом согласованно пересчитываться при всяком изменении взаимоувязанной по вертикальным связям системы объемно-календарных планов различных уровней.

Глава XII

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

12.1. Естественный интеллект и проблемы его моделирования

Как известно, материальным носителем человеческого интеллекта является его мозг, который в соответствии с современными представлениями состоит примерно из 10 миллиардов нервных клеток, называемых *нейронами* *). Каждая нервная клетка состоит из *тела клетки* (с внутренним *ядром*) и отростков — одного *аксона* и одного или нескольких *дендритов*. Тело клетки, равно как и толщина отростков, имеют микроскопические размеры. Невелика обычно и длина дендритов. В то же время аксоны могут иметь значительную длину (измеряемую иногда многими десятками сантиметров). Различают *чувствительные* нейроны, получающие информацию из внешней среды, *вставочные* нейроны, связывающие одни нейроны с другими, и *эффекторные* нейроны, посылающие импульсы к исполнительным органам (например, к мышечным волокнам). Чувствительные нейроны называются также *сенсорными* или *афферентными*, а эффекторные — *эфферентными*. Вставочные нейроны иногда именуются также *интернейронами*.

Чувствительные нейроны получают информацию либо непосредственно от внешних раздражителей, либо (через свои дендриты) от специальных *рецепторных клеток*, реагирующих на такие раздражители. Сигнал, вырабатываемый любым нейроном, передается от тела клетки через аксон. Аксоны, заключенные в специальную оболочку, образуют так называемые *нервные волокна*. Сигналы по этим волокнам передаются электрохимическим путем со скоростью от нескольких метров до нескольких десятков метров в секунду. Каждый такой сигнал представляет собой конечную последовательность нервных импульсов. Частота следования импульсов ограничивается тем, что нервное волокно, пропустив очередной импульс, должно в течение некоторого вре-

*) Мы опускаем здесь из рассмотрения другие структуры, входящие в мозг (например, систему кровоснабжения), которые играют менее существенную роль собственно в процессе мышления.

мени, называемого *периодом рефрактерности*, восстанавливать свою проводимость. В течение этого периода (длящегося обычно от 0,001 до 0,005 с) нервное волокно «отдыхает» (точнее, «накапливает силы») и неспособно проводить сигналы.

Каждый нейрон можно рассматривать как преобразователь информации, входные сигналы которого передаются через дендриты, а выходной сигнал — через аксон. В возбужденном состоянии нейрон передает через аксон на выход серию импульсов, интенсивность которой зависит от степени возбуждения. Если рассматриваемый нейрон не является эффекторным, то его выходной сигнал (через аксон) поступает на входы (дендриты) других нейронов через так называемые *синапсы* — промежутки микроскопического размера между кончиками аксона одной клетки и дендрита другой. В этом промежутке под влиянием сигнала y, пришедшего на аксон, выделяется специальное химическое вещество — так называемый *медиатор*, возбуждающий сигнал x_i соответствующего дендрита. Таким способом отдельные нейроны связываются в сложные структуры, аналогичные рассмотренным в гл. I комбинационным и последовательностным схемам.

Имеются, правда, и существенные отличия от схем, рассмотренных в гл. I. Во-первых, нейроны представляют собой не чисто дискретные элементы, поскольку в зависимости от величины возбуждения могут испускать сигналы разной интенсивности. Во-вторых, под влиянием тех или иных условий может меняться проводимость синапсов (*синаптических контактов*), что эквивалентно возможности перестройки соответствующих дикретно-аналоговых схем. Совокупность имеющихся фактов делает практичеки достоверной гипотезу, что величины ρ_j проводимости всех синаптических контактов мозга составляют содержание его *долговременной памяти*, в то время как величины $z_i = z_i(t)$ текущего возбуждения всех его нейронов — содержание его *оперативной* (краткосрочной) памяти. Синаптические контакты делятся на *возбуждающие* и *тормозящие*. В первом случае возбуждение синапса способствует, во втором — препятствует возбуждению нейрона, дендрит которого «подключен» к данному синапсу. Если среди m дендритов какого-либо нейрона первые k дендритов подключены к возбуждающим синапсам, а последние $m - k$ дендритов — к тормозящим синапсам, то *условие* этого нейрона в первом приближении представится формулой

$$b = \rho_1 x_1 + \rho_2 x_2 + \ldots + \rho_k x_k - \rho_{k+1} x_{k+1} - \ldots - \rho_m x_m - a \geqslant 0. \qquad (12.1)$$

Здесь через ρ_i обозначены проводимости синапсов, через x_i — пришедшие на них (через аксоны других нейронов) входные сигналы, а через a — *порог возбудимости* данного нейрона. Величина z возбуждения рассматриваемого нейрона зависит от

уровня превышения порога, т. е. от величины b (при $b<0$ возбуждение отсутствует полностью).

Что же касается механизма перевода информации из оперативной памяти в долговременную, то, с нашей точки зрения, наиболее вероятна гипотеза различного изменения проводимости ρ_i синаптического контакта в случае различных сочетаний уровней возбуждения двух подключенных к нему нейронов; при совпадении возбуждений проводимость возбуждающих синапсов увеличивается, а тормозящих — уменьшается, при несовпадении возбуждений изменения происходят в противоположном направлении (так называемая *положительная обратная связь*). Величина изменения при этом, по-видимому, зависит как от самих нейронов, так и от истории их предшествующих возбуждений (т. е. от достигнутых значений проводимости).

При *моделировании* нейронов зачастую игнорируют аналоговую сторону их поведения, ограничиваясь лишь дискретной. Чаще всего в качестве такого *искусственного нейрона* выбирают *пороговый элемент задержки* с двумя состояниями (0, 1) и входно-выходными сигналами, также принимающими лишь состояния 0 и 1. Величины ρ_i в условии (12.1) выбираются при этом равными единице, так что условие возбудимости элемента в следующий момент дискретного времени $t+1$ имеет вид

$$k(t)-l(t)\geqslant a, \tag{12.2}$$

где через $k(t)$ и $l(t)$ обозначены соответственно количества возбужденных (в настоящий момент времени t) возбуждающих и тормозящих входов данного порогового элемента, а через $a=$ $=\text{const}$ — величина порога. Комбинацией подобных элементов (с различными величинами порогов) можно смоделировать любую последовательностную схему (конечный автомат).

Моделирование отдельных нейронов и составление из получаемых моделей тех или иных схем представляют собой один из возможных методов (путей) построения *искусственного интеллекта*, который мы будем называть *структурным*. Этот путь весьма привлекателен с точки зрения задачи изучения человеческого мозга. Вместе с тем при современной технологии он не привел пока к созданию искусственного интеллекта, способного всерьез потягаться с естественным интеллектом хотя бы на одном поприще, важном в практическом отношении. Причина здесь та, что уже для моделирования (даже не очень точного) одного нейрона требуются достаточно сложные электронные схемы, а для проявления сколько-нибудь интересных для практики интеллектуальных свойств требуется создавать ансамбли из весьма большого числа нейронов (по крайней мере многих десятков, если не сотни тысяч).

Кроме того, необходимо выбрать достаточно хорошую схему

их соединения, соответствующую схемам, реализующимся в человеческом мозгу. Современная же экспериментальная нейрофизиология прямыми методами может разобраться в схемах соединения в лучшем случае нескольких десятков нейронов. Возможности же косвенных методов также пока не очень велики.

Ввиду перечисленных трудностей прямого (структурного) моделирования, при построении искусственного интеллекта на практике обычно используется совершенно иной подход, который естественно называть *феноменологическим* (или функциональным). Сущность его состоит в том, чтобы строить и воспроизводить на универсальных ЭВМ различные алгоритмы, определяющие те или иные функции человеческого интеллекта. Среди таких функций особый интерес представляют собой *задачи классификации*, *задачи распознавания образов* (прежде всего зрительных и слуховых), *задачи обучения* (накопления знаний), способность вести *разговор на естественных языках*, способность к *целенаправленным действиям*, к *логическому выводу* и др. Нашей дальнейшей задачей в этой главе будет ознакомление с некоторыми методами моделирования перечисленных функций.

Заметим, что задача построения искусственного интеллекта может ставиться (и уже ставится на практике) в двух различных постановках — узкой и широкой. В узкой постановке задача создания искусственного интеллекта отделяется от задачи моделирования органов чувств, человеческой речи и т. п. Принято считать, что подобная задача будет решена, когда удастся создать систему программ и соответствующее информационное наполнение для ЭВМ, которое позволит ей тем или иным способом вести любые осмысленные диалоги с человеком на естественном языке (через пишущую машинку, алфавитно-цифровой дисплей и т. п.). При этом ЭВМ должна обнаружить знания (умение) и способность к обучению новым знаниям (умению), так, чтобы человек, ведущий с ней диалог, в течение сколь угодно большого промежутка времени не смог отличить его от организованного аналогичным образом диалога с обычным собеседником — человеком.

Это условие, называемое обычно *тестом Тьюринга*, еще не достигнуто в сколько-нибудь полном объеме, хотя эффективные системы диалога ЭВМ — человек в определенных (заранее ограниченных) предметных рамках созданы и работают, полностью имитируя диалог человек — человек. То же касается умения: в определенных областях интеллектуальной деятельности (например, в игре в шахматы) уже созданы программы, намного превосходящие умение среднего шахматиста.

Вторая — более широкая постановка задачи создания искусственного интеллекта — требует моделирования не только соб-

ственно интеллектуальной деятельности, но и органов чувств (прежде всего зрения и слуха), а также речи и двигательных (моторных) функций (прежде всего функций человеческой руки). Эта задача стоит сегодня как практическая задача создания *интеллектуальных роботов.*

Хотя задача создания универсальных интеллектуальных роботов, способных заменить человека в любом виде физического труда, в чисто интеллектуальном плане может вполне довольствоваться уровнем теста Тьюринга, для автоматизации многих видов умственного труда такой уровень уже явно недостаточен. Ясно, например, что вообще не было бы никакого смысла создавать ЭВМ, если бы они могли выполнять вычислительную работу лишь на уровне рядового интеллекта.

Очевидная необходимость на отдельных направлениях автоматизации интеллектуальной деятельности намного превосходить рядовые возможности часто рождает максимализм в требованиях к искусственному интеллекту. Задача же превзойти в искусственном интеллекте наивысшие достижения человеческого интеллекта на всех направлениях (или хотя бы сравняться с ними), разумеется, несравненно сложнее задачи моделирования рядового человеческого интеллекта. Не исключено, что необходимость решения подобной задачи в полном объеме практически никогда и не возникнет.

Необходимо, однако, подчеркнуть, что никаких априорных ограничений для автоматизации интеллектуальной деятельности не существует. Нередко в качестве доказательства наличия таких ограничений приводят знаменитую теорему Гёделя о неполноте арифметики. Суть ее заключается в том, что любая формальная теория, заключающая в себе арифметику натуральных чисел, неполна в том смысле, что в ней обязательно существуют содержательно истинные, но формально недоказумые предложения (не выводимые из аксиом данной теории).

Данный аргумент, однако, неубедителен, поскольку ограничения, обуславливаемые теоремой Гёделя, справедливы в равной мере не только для машины, но и для человека, остающегося в рамках данной теории. В то же время ничто не мешает машине (так же, как и человеку) выйти за пределы чисто абстрактного мышления, включив в состав своих действий знаменитую ленинскую триаду: наглядное созерцание (эксперимент) — абстрактное мышление — практика.

Как уже отмечалось выше, ЭВМ успешно применяются для автоматизации как первого элемента триады (эксперимент), так и последнего (управление технологическими процессами). Нетрудно показать, что при таком (естественном) расширении процесса приобретения знаний теорема, аналогичная теореме Гёделя, уже не имеет места.

12.2. Математические основы классификации и распознавания образов

Задача классификации состоит в группировке заданных объектов в некоторое конечное число *классов* $c_1, c_2, \ldots, c_k$. Обычно предполагается, что эти классы не имеют общих элементов (объектов); классы могут быть заданы простым перечислением их элементов: $c_i = \{c_{i1}, \ldots, c_{ik_i}\}$ $(i = 1, 2, \ldots, k)$. В более сложных случаях классификация предполагает составление *описаний объектов* и *правил*, определяющих по этим описаниям принадлежность объектов к тем или иным классам. Процедура применения таких правил к какому-либо объекту (т. е. установления класса, к которому он принадлежит) называется *распознаванием образов*. Понятие *образа* в этом определении эквивалентно понятию класса.

Классификация на основании описаний не обязательно должна быть детерминированной. Широко распространены методы *вероятностной классификации* (и распознавания), задачей которых является установление вероятностей принадлежности классифицируемого (распознаваемого) объекта к тому или иному классу. Широко распространены постановки задач *адаптивной классификации*, называемых, иначе, задачами *обучения распознаванию образов*. При обучении специальному *обучающемуся алгоритму* (программе) предъявляются один за одним описания некоторых объектов вместе с именами классов, к которым они принадлежат. Обучающийся алгоритм с каждым очередным шагом обучения (предъявлением очередного классифицируемого описания) уточняет правила классификации. После работы в течение определенного времени *в режиме обучения* алгоритм переключается на *режим экзамена*. При экзамене наименования классов, к которым принадлежат предъявляемые алгоритму объекты, определяются с помощью выработанных правил классификации. Путем сравнения их с истинными классами (известными экзаменатору) определяется процент правильных распознаваний, по которому судят о качестве обучающегося алгоритма.

Помимо алгоритмов обучения, в классификационных задачах используются и алгоритмы *самообучения*. Их задача состоит в том, чтобы сгруппировать предъявляемые описания объектов в классы на основании свойств самих этих описаний (без указания имен классов априори).

Трудность задачи состоит в том, что правила классификации заранее не фиксируются. Они вырабатываются самим алгоритмом в результате анализа предъявляемых описаний. Разумеется, обойтись полностью без всякого априорного знания ни один практически действенный алгоритм самообучения не может. Сте-

пень совершенства алгоритма определяется прежде всего тем, насколько глубоко упрятаны эти знания. В наиболее развитой форме алгоритмы самообучения связываются со случайными *мутациями* первоначально заложенных (тоже случайных) классификационных правил и алгоритмами *отбора* полезных мутаций (аналогичными естественному отбору в развитии живых организмов) в процессах использования на практике получаемых классификаций.

12.2.1. Байесовские процедуры. Из теории вероятностей хорошо известно *правило Байеса* вычисления *условной вероятности* $p(B|A)$ совершения события B при условии, что произошло событие A:

$$p(B \mid A) = \frac{p(A, B)}{p(A)}, \tag{12.3}$$

где $p(A)$ — (безусловная) вероятность совершения события A; $p(A, B)$ — вероятность одновременного совершения событий B и A.

Байесовский подход к распознаванию образов заключается в выведении правила, устанавливающего вероятность $p(A_i|B_j)$ одной из взаимно исключающих друг друга гипотез $A_1, A_2, \ldots$ $\ldots, A_k$ о состоянии (принадлежности) наблюдаемого объекта в зависимости от исходов $B_1, B_2, \ldots, B_n$ эксперимента (результата наблюдения) с этим объектом. Для такой условной вероятности (в зависимости от исхода эксперимента) в общем случае имеет место формула

$$p(A_i \mid B_j) = \frac{p(B_j \mid A_i)\, p(A_i)}{\sum_{l=1}^{k} p(B_j \mid A_l) p(A_l)}. \tag{12.4}$$

Используя формулы (12.3) и (12.4), можно вычислять вероятность того или иного события A_i в зависимости от исхода B_j эксперимента с ним. При этом предполагается, что известны не только априорные (безусловные) вероятности гипотез A_i, но и условные вероятности различных исходов эксперимента в случае, когда любая данная гипотеза действительно имеет место. Эти последние вероятности могут вычисляться по формуле (12.3), для чего должны быть известны вероятности $p(A_i, B_j)$.

Для пояснения сказанного рассмотрим следующую задачу: пусть требуется в коллективе спортсменов, состоящем из баскетболистов (гипотеза A_1) и футболистов (гипотеза A_2) осуществлять вероятностное распознавание принадлежности спортсмена к одной из двух категорий по его росту. Предположим, что рассматриваются три градации роста: высокий (B_1), средний (B_2) и низкий (B_3). Пусть, далее, известно, что 60% коллектива составляют баскетболисты и 40% — футболисты, а условные ве-

роятности $p(B_j|A_i)$ соответственно равны $p(B_1|A_1)=0{,}7$; $p(B_2|A_1)=0{,}3$; $p(B_3|A_1)=0$; $p(B_1|A_2)=0{,}1$; $p(B_2|A_2)=0{,}8$; $p(B_3|A_1)=0{,}1$.

Иными словами, 70% баскетболистов имеет высокий рост, 30% — средний, а баскетболистов низкого роста нет вовсе. Аналогично, среди футболистов доля спортсменов высокого роста составляет 10%, среднего — 80% и низкого — 10%. Допустим, мы случайно встретили одного из спортсменов высокого роста. Какова вероятность того, что он — баскетболист? Ответ на этот вопрос дает формула (12.4) при $i=1$, $j=1$:

$$p(A_1|B_1)=\frac{p(B_1|A_1)\cdot p(A_1)}{p(B_1|A_1)\cdot p(A_1)+p(B_1|A_2)\cdot p(A_2)}=$$

$$=\frac{0{,}7\cdot 0{,}6}{0{,}7\cdot 0{,}6+0{,}1\cdot 0{,}4}\approx 0{,}91.$$

Таким образом, признак высокого роста в рассматриваемом коллективе достаточно *информативен*: он с большой вероятностью указывает на принадлежность спортсмена к первому классу. Признак среднего роста менее информативен: с вероятностью 0,64 он определяет принадлежность спортсмена ко второму классу. Признак низкого роста в рассматриваемом случае наиболее информативен. Он однозначно (с вероятностью 1) идентифицирует футболиста.

Таким образом, байесовский подход позволяет не только решать (вероятностную) задачу распознавания, но и ранжировать наблюдаемые признаки (исходы экспериментов) по их информативности. Однако крупным недостатком этого подхода, сильно ограничивающим его практическое применение, является необходимость большого объема предварительных данных об объектах классификации, а именно вероятностей $p(A_i)$ и $p(B_j|A_i)$.

Поскольку описанная процедура дает в общем случае лишь вероятности принадлежности объекта к тем или иным классам, окончательный выбор класса, к которому следует его отнести, сопряжен с возможностью ошибок, которые могут вызывать те или иные потери. Величина (математическое ожидание) потерь Q_{jl} при отнесении объекта с исходом эксперимента (описанием) B_l к классу A_j обычно задается формулой вида

$$Q_{jl}=\sum_{i=1}^{k}\omega_{ji}p(A_i|B_l), \qquad (12.5)$$

где через ω_{ji} обозначены некоторые заданные постоянные коэффициенты. Объект относится к тому классу A_i, для которого ожидаемая величина потерь Q_{jl} принимает наименьшее возможное значение.

Если этот минимум достаточно велик, то переходят к более подробным описаниям B_{lq} (делают дополнительный эксперимент).

Если при этом известны условные вероятности $p(B_{lq}|C_i)$, то можно повторить только что описанную процедуру классификации для нового (расширенного) описания. Если минимальная величина потерь все еще велика, можно производить дальнейшее расширение описания. Часто в подобной последовательной процедуре классификации при решении вопроса о продолжении экспериментов (расширении описаний) принимают во внимание не только величину возможных потерь в результате неправильной классификации, но и цену эксперимента для получения новой информации. При этом минимизируется сумма величины потерь в окончательной классификации и цен всех проделанных для ее осуществления экспериментов, что приводит к достаточно сложным задачам динамического программирования.

В случае, когда вероятности $p(C_i)$ и $p(B_j|C_i)$ заранее неизвестны (что чаще всего имеет место на практике), применяются другие классификационные процедуры, описываемые ниже.

12.2.2. Процедуры классификации по расстоянию. Если описания объектов можно представить в виде точек в метрическом (в частности, евклидовом) пространстве, то одни из простейших процедур классификации и распознавания основаны на определении расстояний $r(x, x^{(i)})$ вновь предъявляемой точки (описания) x от ранее предъявленных точек $x^{(i)}$ или других точек, характеризующих сформулированные классы объектов. В евклидовом пространстве для каждого класса A, составленного из точек $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ можно различным способом определить расстояние $r(x, A)$ от новой точки x до класса A. Можно, например, в качестве $r(x, A)$ выбрать расстояние $r(x, a)$ до точки $a = \frac{1}{n}\left(x^{(1)} + \ldots + x^{(m)}\right)$ (предполагается, что точки заданы наборами своих евклидовых координат). Другой способ — положить $r(x, A) = \min\limits_i r(x, x^{(i)})$.

Возможны и другие способы определения расстояния $r(x, A)$. Когда же способ определения такого расстояния выбран, а некоторые начальные классы $A_1, \ldots, A_k$ уже сформированы, то можно распознавать предъявляемую точку x, относя ее к ближайшему по расстоянию классу. Этим способом может быть организован и процесс пополнения классов $A_1, \ldots, A_k$ в режиме самообучения. Если априори известно, что минимальное расстояние s между двумя точками разных классов больше максимального расстояния l между двумя точками одного и того же класса, то при известном s или l очевидным образом строится процедура самообучения (автоматической классификации), начиная «с нуля» (т. е. с пустых начальных классов). Если, например, известна величина s, то, отнеся первую точку $x^{(1)}$ к классу A_1, следующую точку $x^{(2)}$ мы отнесем к тому же классу,

если $r(x^{(1)}, x^{(2)}) < s$, и к другому классу, если $r(x^{(1)}, x^{(2)}) \geqslant s$. Продолжая аналогичным образом, легко построить требуемую классификацию.

В более сложных случаях может оказаться, что интересующие классификатора классы могут получаться как объединения построенных только что указанным способом первичных классов (например, в один класс под именем «дом» могут объединяться внешне не похожие друг на друга одноэтажные хижины и небоскребы). Для правильной классификации в этом случае требуется дополнить режим самообучения режимом обучения, в котором представители объединяемых классов указываются извне. Разумеется, правильное объединение возможно и в режиме самообучения, но при условии переведения описаний в пространство других признаков (связанных в данном случае, прежде всего, с назначением объекта, т. е. с целью, которую он выполняет).

12.2.3. Линейные классификационные процедуры. Предположим, что описания объектов представляют собой точки $x = (x_1, x_2, \ldots, x_m)$ в евклидовом пространстве $\mathbf{R}^m$, заданные своими координатами. Принято говорить, что система функций $f_1(x), \ldots, f_k(x)$ *отделяет* (в пространстве $\mathbf{R}^m$) классы $A_1, A_2, \ldots, A_k$, если объект с описанием x тогда и только тогда принадлежит классу A_i, когда $f_i(x) = \max\{f_j(x), j = 1, \ldots, k\}$ $(i = 1, 2, \ldots, k)$. Классы $A_1, \ldots, A_k$ называются *линейно отделимыми* (в пространстве $\mathbf{R}^m$), если все функции $f_j(x)$ линейны, т. е. если $f_j(x) = a_{j_1}x_1 + \ldots + a_{j_m}x_m + a_{j_{m+1}}$ $(j = 1, \ldots, k)$ с постоянными коэффициентами a_{ji}. Процедуры линейного отделения классов носят наименования *линейных классификационных процедур*. Для удобства записи описания $x = (x_1, \ldots, x_m)$ обычно заменяются эквивалентными описаниями $y = (y_1, \ldots, y_t)$, где $t = m + 1$, $y_i = x_i$ $(i = 1, \ldots, t-1)$, $y_t = 1$. Тогда роль функций $f_j(x)$ выполняют скалярные произведения $(y \cdot a_j)$, где a_j — вектор $(a_{j1}, a_{j2}, \ldots, a_{jt})$ $(t = m + 1)$. Векторы a_j составляют $k \times t$-матрицу A, которую мы будем называть *классификационной матрицей*.

Если априори известно, что классы $A_1, \ldots, A_k$ линейно отделимы, то существует такая *обучающаяся классификационная процедура*:

Выбирается некоторая начальная матрица $A^{(1)}$ (обычно нулевая) и строится последовательность $A^{(1)}, A^{(2)}, \ldots, A^{(n)}$ классификационных матриц. Переход от матрицы $A^{(n)}$ к матрице $A^{(n+1)}$ $(n = 1, 2, \ldots)$ осуществляется после предъявления для классификации очередного описания $y^{(n)}$ по следующим правилам:

1) Если описание $y^{(n)}$ оказалось отнесенным к правильному классу, то $A^{(n+1)} = A^{(n)}$.

2) Если произведена неверная классификация, т. е. если объект с описанием $y^{(n)}$ оказался отнесенным не к своему классу A_i, то найдутся такие j, что $y^{(n)}a_j^{(n)} \geqslant y^{(n)}a_i^{(n)}$, где $a_j^{(n)}$ и $a_i^{(n)}$ — соответственно j-я и i-я строки матрицы $A^{(n)}$. Для каждого такого j полагаем $a_i^{(n+1)} = a_i^{(n)} + y^{(n)}$, $a_j^{(n+1)} = a_j^{(n)} - y^{(n)}$. Тем самым формируется новая классификационная матрица $A^{(n+1)}$, некоторые строки которой отличны от соответствующих строк матрицы $A^{(n)}$.

Доказано, что в случае линейной отделимости заданных классов описанная процедура через конечное число шагов приводит к правильной классификационной матрице $A^{(\gamma)}$. Возможны и другие обучающиеся процедуры линейной классификации, которые не обязательно приводят к той же самой классификационной матрице $A^{(\gamma)}$.

12.2.4. Об одной нелинейной классификационной процедуре. Если отделяющими функциями $f_i(x)$ являются полиномы (от переменных $x_1, x_2, \ldots, x_m$) некоторой степени $s > 1$, то для нахождения их коэффициентов можно применить процедуры линейной классификации, перейдя от признаков $x_1, \ldots, x_m$ к признакам $x_1, x_2, \ldots, x_m, x_1x_2, x_1x_3, \ldots, x_1^{i_1}x_i^{i_2}, \ldots, x_m^{i_m}$, где $i_1 + i_2 + \ldots + i_m = s$. Поскольку полиномами можно аппроксимировать функции более сложной природы, при достаточно большом s с помощью указанного приема можно осуществить практически любую нелинейную классификацию. Однако, поскольку размерность пространства новых признаков очень быстро (экспоненциально) растет с ростом s, на практике подобный *прием сведения нелинейных* классификационных процедур к линейным употребляется лишь при небольших значениях s.

12.2.5. Персептрон. Гипотетическое классификационное устройство, называемое *персептроном*, представляет собой фактически нелинейную классификационную процедуру особого вида, работающую в *пространстве булевых признаков* $(x_1, x_2, \ldots, x_m)$. Напомним, что булева переменная (в данном случае признак) может принимать лишь два значения: истина (и) и ложь (л). Удобно сводить, однако, булевы признаки к арифметическим, полагая и = 1, л = 0.

Персептрон представляет собой двухуровневое устройство (процедуру). На первом уровне с помощью системы булевых функций $y_i = f_i(x_1, x_2, \ldots, x_m)$ $(i = 1, 2, \ldots, n)$, называемых *первичными предикатами*, происходит преобразование системы *первичных булевых признаков* $(x_1, x_2, \ldots, x_m)$ в систему *вторичных булевых признаков* $(y_1, y_2, \ldots, y_n)$. При этом, хотя для каждой булевой функции f_i выписан полный набор аргументов $(x_1, x_2, \ldots, x_m)$, не обязательно, чтобы она фактически зависела от них всех. Более того, как правило, на практике более

употребительны так называемые *локальные предикаты*, каждый из которых зависит от относительно небольшой части признаков $x_1, x_2, \ldots, x_m$. Персептрон называется *ограниченным* k-го порядка, если все его первичные предикаты являются булевыми функциями не более чем от k переменных ($k < m$).

Производя арифметизацию (и = 1, л = 0) булевых признаков $y_1, \ldots, y_n$, мы погружаем пространство этих признаков в обычное евклидово пространство, в котором организуется та или иная линейная процедура обучения классификации, например та, которая была описана в п. 12.2.3. Классификационные способности персептронов при удачном выборе первичных предикатов могут быть очень высокими. Более того, если на первичные предикаты не налагать никаких ограничений, то персептрон можно обучить классификации произвольных разделенных образов (попарно непересекающихся классов). Для этой цели для каждого образа (класса) A_j ($j = 1, 2, \ldots, k$) нужно построить первичный предикат $y_j = f_j(x_1, \ldots, x_m)$, принимающий значение и = 1 на всех описаниях, принадлежащих классу A_j, и значение л = 0 на всех остальных описаниях, а в пространстве вторичных признаков $(y_1, y_2, \ldots, y_k)$ в качестве классификационной матрицы выбрать матрицу $A = \|a_{ij}\|$ ($a_{ii} = 1$, $a_{ij} = 0$ при $i \neq j$ для всех $i, j = 1, 2, \ldots, k$).

Более сильным является утверждение о том, что любая классификация может быть реализована персептроном, у которого в множество первичных предикатов входят все так называемые *маски*. Под маской здесь понимается предикат, который принимает значение и = 1 тогда и только тогда, когда все заданные признаки (признаки маски) $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ принимают значения и = 1, т. е. конъюнкция (логическое произведение) заданных признаков $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$.

12.2.6. Общий случай преобразования описания. Если классифицируемые объекты характеризуются признаками $x_1, x_2, \ldots, x_m$, каждый из которых может принимать любое заданное множество значений (зависящее от номера признака), то с целью удобства последующей классификации часто применяют прием замены *первичных признаков* $x_1, x_2, \ldots, x_m$ *вторичными признаками* $y_1, y_2, \ldots, y_n$, вычисляемыми по заданным формулам: $y_i = f_i(x_1, \ldots, x_m)$ ($i = 1, \ldots, n$). В случае распознавания классов двумерных или одномерных функций вещественного переменного в качестве первичного (приближенного) представления берутся просто *таблицы значений* этих функций в выделенном множестве точек, достаточно густо заполняющих области определений этих функций. Такие таблицы и будут первичными описаниями исследуемых объектов.

Например, $m \times n$-матрица $z = \|z_{ij}\|$ может рассматриваться как таблица значений двумерной функции $z = f(x, y)$ в точках

с координатами α_i, α_j ($i, j = 1, 2, \ldots, m$). Ее можно трактовать как числовое (приближенное) представление некоторого *рисунка*, если считать значение z_{ij} степенью черноты рисунка в точке (α_i, α_j). Если, скажем, матрица z имеет вид

$$z = \begin{Vmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{Vmatrix},$$

то ее можно трактовать как изображение черного знака умножения $\times$ на белом фоне.

Получаемые таким образом непосредственно из «изображений» классифицируемых функций первичные признаки z_{ij} обычно даже вообще не называются признаками. Наименование *признаки* присваивается только получаемым из них вторичным признакам $v_l = g_l(z_{11}, \ldots, z_{nn})$. С такой терминологией мы будем иметь дело ниже, рассматривая зрительные (графические) и звуковые образы.

12.2.7. Системы признаков, инвариантные относительно преобразований. Очень часто, особенно при распознавании зрительных образов, возникают ситуации, когда в один класс зачисляются объекты, получаемые один из другого с помощью тех или иных *преобразований*. Например, в один класс геометрических фигур зачисляются обычно все квадраты, независимо от их расположения и размеров. В данном случае все представители класса квадратов могут быть получены из одного фиксированного квадрата путем применения к нему преобразований подобия, смещения и поворота. В других случаях для порождения всех элементов класса может оказаться недостаточно одного элемента и потребуется использовать для этой цели некоторое множество элементов.

В случае возникновения описанных ситуаций проблемы классификации и распознавания существенно упрощаются при переводе описаний объектов в пространство признаков, *инвариантных* относительно рассматриваемых преобразований. Смысл инвариантности состоит в том, что признаки должны иметь одни и те же значения для объектов, получаемых один из другого с помощью рассматриваемых преобразований. Например, если отдельный треугольник на плоскости характеризуется шестью параметрами (обычно координатами вершин), то класс равных (конгруэнтных) треугольников может быть характеризован тремя параметрами (например, длинами сторон). Эти параметры (признаки) инвариантны по отношению ко всем сдвигам и поворотам. Для характеристики всех подобных треугольников системой инвариантных параметров могут служить величины двух

углов или отношения длин двух любых сторон к длине третьей стороны.

12.2.8. Группы, полугруппы и частичные группы преобразований. В рассмотренных случаях *допустимые преобразования*, т. е. преобразования, которые не выводят преобразуемые объекты за пределы рассматриваемого класса, образуют *группу преобразований G*. Это означает, что для любых двух допустимых преобразований γ_1 и γ_2 их *произведение* $\gamma_1\gamma_2$ (последовательное выполнение преобразований γ_1 и γ_2) также допустимо. *Тождественное преобразование* ε (сохраняющее преобразуемый объект неизменным), очевидно, всегда допустимо. Оно обязательно включается в любую группу преобразований. Кроме того, в группу включаются преобразования γ^{-1}, *обратные* всем преобразованиям группы. Обратное преобразование, выполненное после прямого преобразования, возвращает объект в исходное состояние, т. е. $\gamma \cdot \gamma^{-1} = \varepsilon$; аналогично, $\gamma^{-1}\gamma = \varepsilon$.

Если исключить из условий обязательное наличие обратного элемента и тождественного преобразования ε, приходим к более широкому понятию *полугруппы преобразований*. Если к тому же произведение $\gamma_1\gamma_2$ не обязательно принадлежит допустимому множеству преобразований (или вообще не существует), то говорят о *частичной полугруппе* (или *частичной группе*) преобразований.

Чтобы пояснить необходимость введения частичных полугрупп (групп) допустимых преобразований, рассмотрим пример распознавания буквы Е. При небольших поворотах исходного образца этой буквы она продолжает идентифицироваться как та же буква. Однако при повороте на 90° против часовой стрелки она примет вид Ш и будет идентифицироваться как другая буква (буква «ш»). Поскольку поворот на 90° может быть представлен в виде произведения малых поворотов, то в данном случае допустимой будет не группа (и даже не полугруппа), а лишь частичная группа преобразований.

Часто (как это имело место и в рассмотренных случаях) допустимые преобразования задаются конечными наборами (вещественных) *непрерывных параметров* $\gamma = \gamma(t_1, t_2, \ldots, t_l)$, причем законы перемножения и обращения преобразований являются непрерывными функциями этих параметров. Подобные группы (частичные группы) называются *непрерывными группами* (непрерывными частичными группами) или *группами Ли* (частичными группами Ли). Наиболее часто частичные группы Ли состоят из всех преобразований, достаточно б л и з к и х (по значениям характеризующих их параметров) к тождественному преобразованию ε (или, как принято иначе выражаться, составляющих некоторую *окрестность* преобразования ε). Подобные образования принято называть *локальными группами Ли*.

На понятии локальной группы Ли может быть основан один из наиболее мощных типов классификационно-распознающих процедур, которые мы будем называть процедурами *локальных покрытий* классифицируемых образов. С этой целью для каждого класса A выбирается конечное множество *образцов-эталонов* $a_1, a_2, \ldots, a_r$ этого класса, для каждого из которых строятся локальные группы Ли преобразований $z_1, z_2, \ldots, z_r$, не выводящих преобразованный объект $(a_i' = l(a_i))$ за пределы рассматриваемого класса или по крайней мере не приводящих его в другие классы. Обозначая через $a_i z_i$ множество объектов, получаемых из объекта a_i всеми преобразованиями из z_i, выбирают такое число эталонов a_i, чтобы множества $a_i z_i$ в совокупности *покрывали* все множество A, т. е. чтобы каждый элемент $a \in A$ входил в состав хотя бы одного из множеств $a_i z_i$.

Осуществив подобные покрытия всех классифицируемых образов, мы полностью решим задачу классификации. При предъявлении для распознавания нового объекта q осуществляется решение задачи минимизации расстояния (в пространстве признаков) между объектом q и объектом каждого из множеств $a_i z_i$ (для всех классов A). Объект будет принадлежать тому классу, для которого это расстояние равно нулю (на практике — достаточно близко к нулю).

В общем случае решение задачи распознавания подобным образом может приводить к весьма громоздким вычислениям. На практике, однако, достаточно часто приходится встречаться с частным случаем, когда, во-первых, все z_i одинаковы (равны z) и, во-вторых, $z^{-1} = z$, т. е. для каждого элемента из z существует обратный элемент, также входящий в z. Тогда для решения задачи распознавания какого-либо объекта a необходимо найти такой эталон a_j, для которого минимальное расстояние между шаблоном и элементами множества z имеет наименьшее значение. На практике это обычно означает, что выбирается некоторое число элементов $l_1, l_2, \ldots, l_s$ из z и находится пара (a_{l_j}, a_j) с минимумом расстояний между элементами пары. Возможно, разумеется, и применение общих приемов нахождения минимума между элементами пары $(a(t_1, t_2, \ldots, t_r), a_j)$, где $a(t_1, t_2, \ldots, t_r)$ есть результат воздействия на объект a переменного преобразования $l(t_1, t_2, \ldots, t_r)$, пробегающего всю локальную группу z.

Вместо вычисления расстояний между векторами a_l и a_j иногда прибегают к вычислению коэффициента $\rho_{l,j} = \frac{(a_l \cdot a_j)}{|a_l||a_j|}$, где в числителе стоит скалярное произведение векторов a_l и a_j, а в знаменателе — произведение их длин. Ищется пара (l, j), для которой коэффициент $\rho_{l,j}$ принимает максимальное значение, после чего объект относится к классу с полученным таким об-

разом номером j. Ввиду аналогии между приведенной формулой и формулой (9.16) гл. IX (§ 9.4), коэффициент $\rho_{l,j}$ иногда называют *коэффициентом корреляции* между j-м эталоном и преобразованным объектом.

12.2.9. Формальные грамматики. Перечисленные выше процедуры классификации и распознавания наиболее эффективны при распознавании относительно простых объектов. Для составных объектов обычно применяются комбинированные процедуры, состоящие из нескольких уровней. На нижнем уровне производятся классификация и распознавание отдельных составных частей объекта, а на последующих уровнях — их различных комбинаций. Один из наиболее простых случаев подобной *многоуровневой классификации* представляет собой классификация лексических образов, выражаемых словами или фразами на формальном или естественном языке. Средством для подобной классификации лексических образов служат так называемые *грамматики*, или, более точно, *формальные грамматики*. Рассмотрим сначала так называемые *контекстно-свободные* грамматики.

Для построения *языка* совокупности лексических объектов с помощью такой грамматики задаются двумя классами (алфавитами) символов. Так называемые *терминальные символы*, которые мы будем обозначать строчными буквами русского алфавита, представляют собой элементарные единичные объекты рассматриваемого языка (слова или буквы). *Нетерминальные символы*, обозначаемые прописными буквами латинского алфавита, а также их сочетания будут обозначать те или иные *лексические образы* (классы фраз или отдельных слов). Сама грамматика G задается с помощью конечного числа правил подстановок, называемых *продукциями*. Пусть, например, алфавит нетерминальных символов содержит буквы p, s, v, o, а алфавит терминальных символов включает все буквы русского алфавита и дополнен знаком *пробела* $\square$.

Рассмотрим систему продукций:

$p \to s\square v$ $\quad$ $s \to$ волк $\quad$ $v \to$ ест

$p \to s\square v\square o$ $\quad$ $s \to$ лиса $\quad$ $v \to$ видит

$\quad$ $\quad$ $s \to$ белка $\quad$ $o \to$ небо

$\quad$ $\quad$ $s \to$ тушканчик $\quad$ $o \to$ мясо

. . . . . .

Отправляясь от единственного *начального символа*, в качестве которого выберем символ p, будем производить соответствующие *подстановки* вместо левых частей продукций их правых частей. На первом уровне подстановок получим два обобщенных лексических образа $s\square v$ и $s\square v\square o$, которые будут обозначать классы фраз, состоящих в первом случае из подле-

жащего (s) и сказуемого (v), а во втором — из подлежащего (s), сказуемого (v) и дополнения (o). Осуществляя дальнейшие подстановки, получим фразы русского языка: «волк видит», «белка ест», «лиса видит дерево», «белка ест мясо», «волк ест небо» и т. п.

Все полученные фразы являются правильно построенными с точки зрения *синтаксиса* русского языка, хотя с точки зрения *семантики* языка (т. е. его смыслового содержания) последние две фразы лишены смысла. Однако с помощью грамматик можно описывать не только синтаксис, но и семантику языка. Для этой цели в грамматику вводятся дополнительные нетерминальные символы. В данном случае достаточно объекты, выполняющие роль подлежащего (s), разделить на два класса (отряда): хищников (C) и грызунов (R). Тогда система продукций:

$p \to s \square$ ест	$o \to$ небо	$p \to C \square$ ест $\square$ мясо
$p \to s \square$ видит	$o \to$ мясо	$C \to$ волк
$p \to s \square$ видит $\square\ o$	$s \to C$	$C \to$ лиса
	$s \to R$	$R \to$ белка
		$R \to$ тушканчик

будет порождать лишь осмысленные (семантически правильные) фразы русского языка.

Построение подобных *семантических грамматик* может быть положено в основу классификации фраз любого естественного языка по принципу осмысленных и лишенных смысла. Правда, ограничение лишь классом контекстно-свободных грамматик вынуждает при этом, как правило, значительно увеличивать как число нетерминальных символов, так и число продукций.

Более мощным средством формального представления синтаксиса и семантики языков являются так называемые *контекстно-зависимые* грамматики. Продукции в таких грамматиках имеют вид $\xi A \eta \to \xi F \eta$, где нетерминальный символ A заменяется последовательностью из нетерминальных и терминальных символов, лишь находясь в контексте $\xi \ldots \eta$ двух последовательностей ξ и η нетерминальных и терминальных символов. Простая замена A на F (независимо от контекста) не допускается. В случае, когда последовательности ξ и η пусты, возвращаемся к уже рассмотренным выше контекстно-свободным продукциям.

Контекстно-зависимые грамматики, будучи более сложными для изучения и оперирования, обеспечивают вместе с тем наиболее мощное средство для порождения языков. Среди контекстно-зависимых языков, содержащих бесконечное множество фраз, нетрудно найти примеры языков, не порождаемых никакими конечными коптекстно-свободными грамматиками.

Заметим, что, используя понятия «близко», «далеко», «внутри», «снаружи», «справа», «слева» и др., можно описать лингвистическими образами пространственные (зрительные) образы для составных объектов. Тем самым круг применения грамматического подхода к задачам классификации может быть существенно расширен.

12.2.10. Семантические сети, фреймы и сценарии. Для обычных языковых форм характерно линейное представление информации. Это обстоятельство вносит дополнительные трудности в представление семантических связей между отдельными элементами языка. Поэтому гораздо удобнее вместо линейных структур пользоваться более общими структурами данных. Особенно большое значение в задачах искусственного интеллекта приобрели сетевые структуры (см. гл. VI), которые в этом случае получили специальное наименование *семантических сетей*. В вершинах таких сетей располагаются различные *понятия*, направленные же связи между вершинами соответствуют различного рода *отношениям* между этими понятиями. Заметим, что структура языка, включающая в себя не только терминальные записи, но и все *дерево вывода*, определяемого формальной грамматикой, представляет собой *иерархическую структуру* (см. гл. VI), являющуюся, как известно, частным случаем сетевой структуры. Для целей искусственного интеллекта возможно использование и реляционных структур данных.

По линиям вертикальных (иерархических) связей располагаются отношения вхождения в классы: например, понятие «белка» является частным случаем понятия «грызун» (представитель отряда грызунов). Понятие же «грызун» является частным случаем понятия «млекопитающее», «животное» и т. д. По линиям боковых связей располагаются другие отношения. Например, понятия «белка» и «шерсть» связаны между собой отношением «быть покрытым».

Семантические сети могут быть сделаны обучаемыми и *растущими*. Последнее означает возможность автоматического добавления в сеть новых узлов по мере появления в опыте ее использования новых понятий. При стабилизации числа узлов сети в процессе ее обучения между узлами могут устанавливаться новые связи. В ряде случаев оказывается целесообразным устанавливать силу связей, придавая им характеризующий эту силу параметр с вещественными значениями. Значению этого параметра, равному нулю, соответствует фактическое отсутствие указанной в структуре сети формальной связи. Увеличение значения параметра означает усиление роли данной связи, что может быть использовано при установлении порядка распределения «возбуждения» от одних узлов сети к другим. Так, в зависимости от аспектов рассмотрения (зависящего от

характера использования сети) связь понятий «белка» — «хвост» может оказаться более важной, чем понятий «белка» — «шерсть», и наоборот. Значения весов связей увеличиваются при каждом полезном использовании связи (например, для целей классификации) и уменьшаются, когда она длительно не используется. Короче говоря, обучение семантической сети напоминает процесс обучения естественных нейронных сетей.

В семантические сети общеязыкового назначения включаются обычно лишь универсальные (всегда существующие) связи. Возможны также семантические сети, описывающие ситуации и составные образы, которые могут иметь частное значение и зависеть от точки зрения или личного опыта наблюдателя. В качестве примера можно указать на связь понятий «сердитый» или «добрый» с именем конкретного человека, понятия «окно» с понятием «комната» (бывают ведь комнаты без окон), наконец, связи между отдельными элементами поверхности геометрического тела (например, куба), рассматриваемого с определенной позиции. Семантические сети, описывающие подобные ситуации, принято называть *фреймами*. С помощью фреймов описываются те или иные конкретные *сцены*.

Для описания последовательности действий используются так называемые *сценарии*, т. е. конечные множества сцен, связанных между собой *условиями перехода* от одних сцен (фреймов) к другим. Таким образом, отдельные *статические* фреймы оказываются связанными в семантическую сеть (блок-схему алгоритма развертывания сценария), создавая в результате сложный *динамический* фрейм. Например, сценарий, заключающийся в том, что наблюдатель, рассматривавший в первой сцене куб с его торца (когда видима лишь одна грань), сместился во второй сцене вправо и вверх (в результате чего ему оказались видимыми пары граней), может быть представлен в виде последовательности двух фреймов (рис. 12.1). Здесь через A обозначено понятие «куб», через B — «поверхность куба», через x — «квадрат», через y — «параллелограмм». Отношение «α» связывает куб с его поверхностью, а отношения ε, β, γ, δ — поверхность с составляющими ее элементами. Отношения ν, μ представляют собой отношения примыкания (соседства) элементов (сверху, справа, сверху). Описания сцен на рис. 12.1 соответствует видимым картинкам на рис. 12.2.

Разумеется, отношение примыкания должно включать в себя требование полного совпадения соответствующих сторон примыкающих друг к другу параллелограммов (для простоты мы считаем видимые элементы поверхности параллелограммами, что можно считать справедливым при достаточном удалении наблюдателя от куба).

Как и в случае общих семантических сетей, для фреймов может быть задействован режим обучения. Кроме того, часто применяются фреймы с избыточными элементами, которые при сравнении контрольного фрейма с заданной оценкой могут опускаться, так что узнавание сцены будет происходить по неполному совпадению.

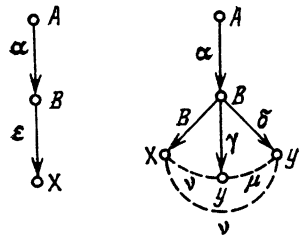

Рис. 12.1

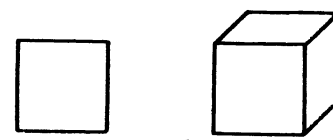

Рис. 12.2

Фреймы могут формироваться из семантических сетей. Так, найдя в семантической сети узел, соответствующий понятию «комната», мы можем, начиная от него, двигаясь по связям, выражающим отношение «входит в состав» (является составной частью), восстановить все предметы, которые могут быть в комнате, включая и составляющие ее элементы (стены, пол, потолок, окно, дверь, стол, стул, кровать и т. д.). Добавляя к найденным узлам связи, выражающие относительное расположение объектов (окно *в* стене, стул *на* полу и т. д.), сформируем полный фрейм, соответствующий понятию «комната». Параметры сил связей будут отражать частоту встречаемости соответствующих комбинаций узлов в практике обучения семантической сети.

12.3. Распознавание зрительных образов

Зрительные образы в задачах автоматического распознавания обычно формируются с помощью оптических систем на прямоугольной решетке искусственных рецепторов (фотоэлементов), называемой, по аналогии с человеческим глазом, *сетчаткой* (ретиной). Каждый рецептор образует аналоговый сигнал (величину уровня освещенности), который затем преобразуется в цифровую форму или, как иногда еще говорят, квантуется по уровням. В простейшем случае двухуровневого квантования на выходах рецепторов возникают булевы сигналы, достаточные для распознавания контурных рисунков; при многоуровневом квантовании сигналы интерпретируются как целые или рациональные числа, кодирующие *полутоновые* рисунки. Устанавли-

вая рецепторы для различных цветов, можно кодировать и цветные изображения.

Помимо описанного *параллельного ввода* зрительных образов может применяться и более простой *последовательный ввод*. Световой (точечный) зайчик обегает распознаваемый рисунок в той или иной последовательности (системе развертки), а отраженный свет, фиксируясь фотоумножителем, преобразуется в аналоговый, а затем и в цифровой сигнал. Полученный таким образом набор (последовательность) сигналов, так же как и в параллельном случае, удобно представлять в виде матрицы $\|x_{ij}\|$, где x_{ij} — булевы или числовые величины.

Распознавание зрительных образов в общем случае основывается, как правило, на методах, уже описанных в предыдущем параграфе. В случае простых (не составных) образов наиболее употребителен метод сравнения с эталонами, описанный в п. 12.2.8. Например, если требуется распознавать детали какого-либо механизма или прибора, можно запомнить в качестве эталонов *трехмерные* образы всех таких деталей с соответствующей градацией окраски их поверхностей. Затем последовательно решают для каждого шаблона задачу проекции эталона в перемещающийся глаз наблюдателя (на двумерную сетчатку) при его освещении перемещающимся источником света. Если через (x, y, z) обозначить координаты глаза, через (u, v, w) — координаты источника света, а через t — интенсивность этого источника, то на сетчатке воображаемого глаза возникнет изображение $\|y_{ij}\|$, каждая компонента которого будет функцией семи переменных: x, y, z, u, v, w, t. Находя минимум расстояния между заданным изображением $\|x_{ij}\|$ и переменным изображением эталона $\|y_{ij}\|$, считают распознаваемое изображение изображением того эталона (детали), для которого указанный минимум имеет наименьшее абсолютное значение.

Указанный метод в чистом виде на практике почти не применяется, ввиду сложности вычислений. Размерность задачи (число переменных) можно уменьшить. Во-первых, вводя в зрительную систему дальномер, можно определить расстояние от глаза до распознаваемой детали и тем самым уменьшить число независимых координат (x, y, z), определяющих положение глаза относительно детали, с трех до двух. Далее, вводя вместо абсолютных освещенностей относительные (в сравнении с какой-либо фиксированной точкой детали), можно устранить переменную t. Наконец (и этот прием наиболее часто применяется на практике), можно предварительно выделить *контуры* наблюдаемой детали и осуществлять сравнение также с контурной проекцией эталона на сетчатку воображаемого глаза. При этом освещенности вообще исключаются из рассмотрения, а варьируемыми (при наличии дальномера) оказываются только два пара-

метра, определяющие относительное расположение глаза и наблюдаемой детали. В случае наличия эталонов с одинаковыми контурами, различающихся окраской, цвет (или чередование полутонов) отдельных частей поверхности детали может быть привлечен для распознавания в качестве дополнительного признака.

Даже при всех указанных упрощениях задача распознавания даже не очень сложных объемных деталей требует достаточно большого объема вычислений. Чтобы решать ее за разумное время (сравнимое с восприятием человека), требуется ЭВМ высокой производительности. Поэтому задача эффективного и притом достаточно дешевого распознавания зрительных образов требует создания специализированных схем, в которых выделение признаков (в частности, контуров изображения) и преобразований эталонов выполняется не последовательными, а параллельными процедурами. В случае распознавания двумерных изображений (например, букв и цифр) положение еще более упрощается, поскольку (в соответствии с п. 12.2.8) можно подвергать преобразованиям (притом не всей группы движений, а лишь локальной группы сдвигов) не эталоны, а само распознаваемое изображение. На этом принципе построены и успешно работают высокоэффективные читающие автоматы (превосходящие возможности человека), специализированные на чтении печатных букв и цифр.

Высокоэффективные (и притом обучающиеся и переобучающиеся) системы автоматического распознавания относительно простых зрительных образов строятся также на принципе персептрона (см. п. 12.2.5).

При распознавании сложных (составных) изображений наиболее эффективными оказываются многоуровневые (чаще всего — двухуровневые) автоматические распознающие системы. На первом (нижнем) уровне производится распознавание отдельных деталей и описание их взаимного расположения. На втором уровне производится сопоставление полученных описаний с заполненными в ЭВМ фреймами — эталонами. Задача распознавания деталей на изображениях составных *пространственных* объектов, как правило, усложняется за счет того, что изображения одних деталей могут частично перекрывать изображения других деталей. С помощью дальномера, однако, можно разобраться, какие детали находятся ближе, а какие дальше. Тем самым появляется возможность определить, какая часть наблюдаемой детали заменяется более близко расположенными деталями. Это обстоятельство учитывается соответствующим отсечением частей эталонов, проектируемых на воображаемую сетчатку, так что опознание идет не по целому изображению, а по некоторой его части. Разумеется, при таком опозна-

нии возможны ошибки, которые, однако, имеют место и в случае обычного (человеческого) распознавания.

Заметим, что при распознавании рукописных букв и цифр, написания которых варьируются в достаточно широких пределах, бывает удобно рассматривать их как составные объекты. Например, цифра 3 считается при этом составленной из двух «деталей», буква *А* — из трех «деталей» и т. п. При распознавании подобных «деталей» используются специальные признаки (например, углы наклонов касательных в начале и в конце «детали»). Можно также использовать более сложные, не сводящиеся к сдвигам и поворотам, локальные группы Ли преобразований наблюдаемых деталей для их последующего сравнения с эталонами либо, наконец, просто иметь достаточно много эталонов для возможных вариаций начертаний каждой детали. При вписывании букв и цифр в специальные шаблоны (как это имеет, например, место при индексации писем) возможны и более простые методы распознавания. Примером таких методов может служить метод подсчета числа пересечений контура распознаваемого знака с некоторыми отрезками (векторами), положение которых относительно поля знака точно фиксировано (например, системы горизонтальных и вертикальных отрезков, пересекающих поле знака).

Следует подчеркнуть, что задача распознавания зрительных образов произвольной природы представляет собой одну из важнейших задач искусственного интеллекта. В этом нетрудно убедиться, вспомнив, что у человека решением этой задачи занята добрая половина нейронов головного мозга. Поэтому создание систем машинного зрения, которое по *всем* параметрам превзошло бы (или хотя бы сравнялось) зрительные способности человека,— задача не такого уж близкого будущего. Практически же сегодня задача ставится таким образом, чтобы к концу столетия системы машинного зрения превзошли возможности человека во всех (или хотя бы почти всех) областях, важных для прикладных целей и, прежде всего, для обеспечения функционирования интеллектуальных роботов. В определенных областях (например, в быстром отыскании мелких различий распознаваемого отображения от эталона) эта задача уже решена. Неплохо решается сегодня задача автоматического распознавания букв и цифр, напечатанных стандартным шрифтом. ЭВМ делает это не только быстрее человека, но и надежнее распознает плохо пропечатанные знаки (правда, пока лишь в том случае, когда человек не прибегает к угадыванию неразборчивых букв из контекста). Однако в целом в автоматизации распознавания зрительных образов предстоит выполнить еще немалый объем работ, чтобы достичь упомянутой выше практической цели.

12.4. Генерация изображений

Задача *генерации изображений* обратна задаче распознавания зрительных образов. В задаче распознавания по изображению надо дать его словесное описание (в простейшем случае — указать имя класса, к которому оно принадлежит). В задаче генерации, наоборот, по словесному описанию производится восстановление описываемого изображения на экране дисплея или с помощью графопостроителя. В простейшем случае эта задача решается тривиально: по имени класса выбирается из памяти ЭВМ один из эталонов, принадлежащих этому классу, и производится его отображение. Поскольку эталон, как правило, представляет собой просто закодированные цифровым кодом значения освещенности различных точек сетчатки, проблема сводится по существу к простому цифро-аналоговому преобразованию.

При необходимости генерации изображений составных объектов используют в качестве исходной информации их фреймы. При этом в качестве изображений элементарных объектов берутся соответствующие эталоны, а их комбинация в составной объект производится в соответствии с отношениями, заданными во фрейме.

Наиболее сложна задача генерации изображения по произвольному (неформализованному) его описанию на одном из естественных человеческих языков. Она распадается на две задачи: перехода от неформализованного описания к формализованному (обычно в виде фрейма) и генерации изображения по полученному формализованному описанию. Первая задача по существу эквивалентна пониманию с м ы с л а текстов на естественных языках. Она пока еще (в полном виде) далеко не решена. Однако основные принципы ее решения достаточно ясны. Речь идет о распознавании *лингвистических образов*, прежде всего — законченных фраз. К их классификации и распознаванию можно применить рассмотренный выше (в п. 12.2.9) аппарат формальных (семантических) грамматик или расширенных семантических сетей, в которых для выражения понятий и отношений между ними фиксируются различные (по возможности все) их конкретные языковые представления. Сети должно быть известно, например, что отношение «А слева от Б» эквивалентно отношению «Б справа от А», что в состав описания отношения могут быть включены слова «находится», «расположен» и т. п.

12.5. Распознавание и генерация речевой информации

Подобно тому как письменная речь представляется последовательностью букв (включая пробел и знаки препинания), устная речь может быть представлена в виде последовательности элементарных звуков (включая паузу), называемых *фонемами*.

Обычно для русского языка достаточно использовать 42 фонемы, которые включают в себя паузу, гласные звуки а, о, у, э, и, ы, полугласный й, твердые согласные ж, ш, ц, з, мягкую согласную ч и два варианта (мягкий и твердый) всех других согласных; увеличение количества фонем улучшает описание тонких нюансов произношения, но затрудняет решение задач генерации и распознавания речи.

Записывая отдельные фонемы на специальные аналоговые ЗУ (например, на специальный магнитный барабан или диск), можно (с помощью ЭВМ) управлять выборкой и воспроизведением на громкоговорителе этих фонем в любых желаемых последовательностях. Такие последовательности задаются в обычной (цифровой) памяти ЭВМ текстами, которые вместо обычных букв используют символы соответствующих фонем. При упрощенном (42-фонемном) представлении речи задача автоматического перевода текстов с обычного (буквенного) представления в фонемное не представляет особых затруднений, так как описанный автоматический *синтезатор речи* может переводить в звуковую форму обычные (письменные) тексты.

Подобные устройства созданы и работают в различного рода справочно-информационных системах. Правда, при сокращенном наборе фонем и отсутствии управления интонациями подобная синтаксическая речь имеет не вполне человеческое звучание, но, будучи полностью (и притом без особого напряжения слуха) понятной и разборчивой, вполне удовлетворяет обычным требованиям к такого рода системам. Впрочем, пути дальнейшей шлифовки машинной речи достаточно ясны, так что в случае возникновения надобности в ее дальнейшем «очеловечивании» особых затруднений не предвидится. Заметим также, что для целей автоматического обучения дикции и правильному произношению (при изучении иностранных языков) может употребляться другой способ искусственного воспроизведения речи. В этом способе в аналоговую память записываются не фонемы, а отдельные слова, части фраз и даже целые фразы, произносимые диктором с хорошо поставленным голосом; по заданному письменному тексту ЭВМ воспроизводит нужные комбинации этих записей.

Задача автоматического *распознавания речи* несравненно сложнее задачи ее автоматического синтеза (генерации) и решается сегодня на уровне, пока еще существенно уступающем возможностям человеческого слуха. Прежде всего сделаем несколько замечаний о превращении речевых сигналов в дискретные (цифровые) описания. Наиболее простой путь — прямые замеры, с превращением в цифровую форму, амплитуд речевого сигнала в достаточно близкие моменты времени (обычно равноотстоящие). При подобном прямолинейном подходе к кодированию речевой информации, с тем чтобы не потерять нужной для

распознавания информации, на одну секунду звучания речи должно приходиться порядка 7000 замеров амплитуд с точностью порядка 1%, что составляет около 50 тысяч бит информации на одну секунду звучания речи. Это количество чересчур велико для эффективного решения задачи распознавания. Поэтому на практике употребляются различные способы *сжатия первичной информации* о речевом сигнале.

Мы опишем кратко лишь один из наиболее употребительных на практике методов сжатия первичной речевой информации, относящийся к классу так называемых *параметрических методов*. При его использовании с помощью аналого-цифровых устройств непрерывно вычисляются значения параметров, характеризующих *огибающую* и *тонкую структуру текущего спектра* воспринимаемого речевого сигнала. Например, огибающая, показанная на рис. 12.3, может быть приближенно представлена параболой $y = a_0 + a_1\varphi + a_2\varphi^2$, а тонкая структура — синусоидой $y = b\sin(k\varphi + c)$, так что весь изображенный на рисунке спектр может быть представлен шестью параметрами: a_0, a_1, a_2, b, k, c (в действительности их может быть, разумеется, и больше).

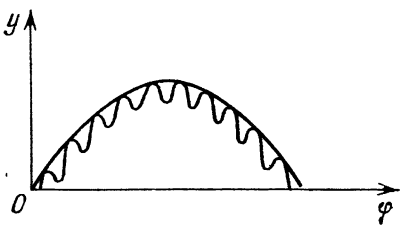

Рис. 12.3

Речевая информация характеризуется тем, что полученные параметры являются относительно медленно меняющимися функциями времени, так что дискретность их измерений может быть сделана достаточно большой. В результате таких измерений удается сжать первичную информацию до 2400 и даже до 1200 бит на секунду звучания речи. Именно к этой информации (или к сжатой любым другим способом) будут применяться описываемые ниже методы классификации и распознавания.

Наиболее просто задачи классификации и распознавания решаются для отдельно произносимых слов, когда число таких слов невелико (обычно от 20 до 50). В этом случае достаточно эффективно действует простейший механизм сравнения (по расстояниям или коэффициентам корреляции) с эталонами. Правда, для надежного распознавания голосов различных факторов чаще всего требуется иметь свой набор эталонов для каждого из них. Для настройки системы на голос нового диктора ему требуется произнести в определенном порядке все распознаваемые слова. Пофонемный метод распознавания при малом количестве слов оказывается более сложным, поскольку границы между фонемами в параметрическом представлении (как, впрочем, и в непосредственном функциональном) установить нелегко.

При распознавании слитной речи трудности еще более возрастают, особенно в том случае, когда размеры *словаря* (количества используемых слов) вырастают до нескольких сотен и тем более нескольких тысяч слов. Здесь, как и в случае фонем, нелегко определить границы между отдельными словами. Причин такого положения по крайней мере три: во-первых, между словами может не быть четких пауз; во-вторых, паузы чаще всего заполняются шумами, трудно отличимыми от речевого сигнала; в-третьих, паузы внутри слов (например, в слове «четыре») могут быть более длительными, чем паузы между словами. В силу этих причин для распознавания слитной речи используются методы динамического программирования, минимизирующие различие не между отдельными распознаваемыми словами и соответствующими эталонами, а между последовательностями слов и различными последовательностями эталонов. При этом используются различные *сегментации* (членения фразы на отдельные слова) и минимизируются суммы различий сегментов от эталонов при разных сегментациях. Подобная процедура весьма трудоемка и при реализации на универсальных ЭВМ требует очень высокого быстродействия для распознавания речи в темпе ее произнесения (или, как обычно говорят, в реальном масштабе времени).

Выход здесь, как и при распознавании зрительных образов, состоит в создании специализированных (относительно дешевых) схем, распараллеливающих процесс распознавания. Заметим также, что описанный метод предполагает предварительную настройку системы на распознаваемый голос. Настройка состоит в создании эталона каждого используемого слова, для чего необходимо произнести и ввести в систему (по одному разу) в определенном порядке все слова заданного словаря. Одним из преимуществ рассматриваемого метода является возможность относительно быстрой смены (с последующей настройкой) и расширения используемого словаря.

Подобный же метод варьируемых сегментаций и минимизации различий последовательностей можно применить не для пословного, а для пофонемного распознавания (хотя переход к пофонемному распознаванию связан обычно с увеличением количества ошибок). Для увеличения надежности пофонемного распознавания следует более тщательно изучить различные конкретные формы представления одних и тех же фонем, произносимых различными голосами, с различными интонациями, в различных темпах и в различных словах. Короче говоря, необходимо изучить и найти достаточно удобные и экономные представления для локальных групп Ли возможных преобразований различных фонем. В этом случае можно применить описанные выше (в п. 12.2.8) общие методы классификации и преобразо-

вания. Еще более простые методы решения этих задач стали бы возможными при нахождении параметрических представлений отдельных фонем, инвариантных относительно преобразований построенных (локальных) групп (см. п. 12.2.7).

На практике сегодня построены адаптивные (настраивающиеся на любой заданный голос) системы распознавания слитной человеческой речи со словарем до 1000 слов, способные достаточно надежно работать даже на фоне шумов. Системы распознавания отдельных слов с небольшим словарем строились еще в 60-е годы. Развитие микропроцессоров позволило выпускать компактные и достаточно дешевые системы распознавания отдельных слов с достаточно большими словарями (несколько сотен слов).

12.5.1. Общая задача распознавания звуковой информации. На описанных принципах могут быть построены системы распознавания не только речевой, но и любой другой звуковой информации. Для распознавания нот (автоматического перекладывания на ноты) мелодий, исполняемых одним музыкальным инструментом, такие системы оказываются даже проще систем для распознавания речевой информации. Для более специальных целей (например, диагностики неисправностей механизмов по издаваемому ими шуму) могут с успехом использоваться другие методы классификации и распознавания, изложенные в § 12.2.

12.6. Проблема понимания текстов на естественных языках и обучения знаниям

Ответ на вопрос о путях автоматизации понимания текстов затрудняется отсутствием общепринятого определения того, что представляет собой понимание. С нашей точки зрения, суть понимания текста состоит в возбуждении этим текстом всех узлов семантической сети, связанных с понятиями и отношениями, используемыми в тексте. Разумеется, речь идет прежде всего о возбуждении всех узлов, непосредственно связанных с данным текстом. Такое возбуждение в свою очередь должно вызвать возбуждение новых узлов (но уже в более слабом виде) и т. д. В результате на сети как бы «высвечиваются» фреймы, сцены и сценарии, расширяющие фреймы и сценарии исходного текста (в соответствии с накопленными в сети знаниями). Например, фраза «я вхожу в комнату» должна вызвать высвечивание фреймов конкретных комнат после возбуждения узлов «дверь», «окно», «стол» и других понятий, связанных с понятием «комната». Сочетание слов «комната» и «вхожу» должно приводить к тому, что наиболее ярко высветится понятие, связанное с ними обоими, а именно понятие «дверь» (разумеется, если предшествующий опыт, проторивший связи в сети, был свя-

зан с более частым вхождением в компату через дверь, а не через окно или, скажем, простую дыру в стене).

Дальнейшая задержка внимания на приведенной фразе может привести к дальнейшему расширению круга высвечиваемых понятий. В их число могут войти, например, понятия «квартира», «дом» и др. В случае, когда в предложенной фразе речь идет не об абстрактных категориях, желательно дополнить процесс высвечивания фреймов выводом соответствующих ему зрительных (а возможно, и звуковых) образов. Адекватность последовательностей таких образов распознаваемому тексту явится внешним свидетельством того, что система действительно поняла предъявленный ей текст. Способы генерации зрительных и звуковых образов по их описаниям (включаемым в фреймы) уже излагались выше.

Следует подчеркнуть, что в семантическую сеть, помимо понятий и отношений, которые можно определить через другие, более простые понятия и отношения, могут включаться и первичные объекты (неопределимые словесно). Для их понимания системой необходимо включать в сеть описания образов таких объектов (наглядных примеров). Например, для объяснения отношения «справа от» может использоваться несколько картинок, изображающих те или иные объекты, расположенные справа от другого объекта, или группы объектов. Описания таких образов помещаются в семантическую сеть (в виде некоторых ее узлов) наравне с понятиями, выражаемыми словами.

Далее, проблема понимания текстов на естественных языках не может считаться до конца решенной, если предназначенная для этих целей автоматическая система не способна вести осмысленный диалог с человеком и, самое главное, обучаться в результате такого диалога. Обучение состоит в изменении и расширении семантической сети системы. Встретив в процессе диалога незнакомое ей слово, например «банан», система должна спросить: что это такое? Получив объяснение типа «банан есть разновидность плода», «банан съедобен», «он имеет продолговатую форму» и т. п., система должна автоматически ввести в сеть узел с именем «банан» и связать его с уже известными ей понятиями «плод», «съедобный», «продолговатый» и др. Желательно также, чтобы в дополнение к подобным словесным описаниям системе были представлены образцы реальных бананов (или их рисунков). Составляя их зрительные описания и запоминая полученные в результате эталоны, система связывает их с понятием «банан» в качестве наглядных примеров этого понятия.

При ведении диалога важно уметь осуществлять автоматический перевод с внешнего языкового представления на язык семантической сети и обратно. С этой целью удобно использовать

аппарат формальных семантических грамматик с процедурами классификации и объединения языковых оборотов, равнозначных по смыслу. Поскольку описания классов эквивалентных оборотов в полном словаре были бы очень громоздкими, можно употребить специальные способы сжатия таких описаний с помощью использования возможно более общих понятий (нетерминальных символов грамматики). Например, вместо того, чтобы запоминать огромное количество эквивалентностей типа: фраза «человек идет» эквивалентна фразе «идет человек», фраза «заяц бежит» эквивалентна фразе «бежит заяц» и т. д., следует запомнить лишь одну эквивалентность: фраза «объект движется» эквивалентна фразе «движется объект».

Разумеется, любые семантические эквивалентности в языке до известной степени условны, поскольку даже в приведенных примерах перестановка подлежащего и сказуемого заключает в себе небольшое изменение смысловых акцентов. Однако именно подобные условные эквивалентности позволяют системе изложить предъявленный ей текст «своими словами», что также является одним из признаков понимания его смысла. Более того, для возможности упрощенного изложения предъявленного системе текста, помимо эквивалентностей, необходимо использовать и процедуры усечения фраз, например отбрасывание дополнений и определений, и т. п. В состав описания классов эквивалентности могут быть введены коэффициенты предпочтительности различных (эквивалентных друг другу) языковых оборотов. При условии использования в первую очередь наиболее употребительных оборотов система как бы получает свой индивидуальный «литературный стиль». Формированию такого стиля способствует также введение предпочтительности тех или иных сокращений или, наоборот, вставок, относительно мало влияющих на смысл фраз, но придающих им особый языковый колорит.

Следует заметить, что построение семантических грамматик в значительной мере облегчается применением процедур, строящих семантическую классификацию в результате анализа предъявляемых системе правильных и неправильных фраз (как с точки зрения синтаксиса, так и с точки зрения семантики). Один из способов такой классификации был предложен автором еще в 1961 г. Суть его состоит в следующем. Имея набор правильных сочетаний разных подлежащих с одним и тем же сказуемым, например «студент говорит», «профессор говорит», «отец говорит», «сын говорит» и т. д., система автоматически объединяет все такие подлежащие в один класс (в данном случае в класс говорящих) и дает ему некоторое имя A. Далее, если оказывается, что определенная часть объектов построенного класса сочетается с другим сказуемым, например со сказуемым «думает», то система выдвигает гипотезу, что это верно и для остальных

ных объектов данного классса. Сформулировав (случайным образом) некоторые фразы с новым сказуемым, например «студент думает», «отец думает» и получив подтверждение от человека, что это — правильные фразы, система объединяет оба сказуемых в один класс B и образует фрагмент грамматики: AB, $A \to$ студент, $A \to$ профессор, $A \to$ сын, $A \to$ отец; $B \to$ говорит, $B \to$ думает.

В последующем опыте система может узнать, что образованный ею класс A может быть назван именем «человек» и тем самым включает соответствующий нетерминальный символ в свою семантическую сеть. Разумеется, описанная процедура не исключает того, что первичная классификация окажется неверной (например, что профессор будет помещен в один класс с граммофоном). Однако в последующем опыте подобные ошибки, как правило, выясняются и исправляются.

12.6.1. Формальные диалоговые системы. От описанных систем, ведущих *осмысленный диалог* с человеком (которые пока еще достаточно далеки от полной реализации), следует отличать *формальные диалоговые системы*, которые могут вести внешне осмысленный диалог на достаточно узкую тему. Понимание вопроса (или ответа) человека здесь заменяется обычно выделением из него заранее записанных в память системы слов — дескрипторов и формированием в зависимости от них одного из заранее записанных в память системы (или формируемых специальной вероятностной или детерминированной процедурой) ответов (вопросов). Например, на вопрос «какая сегодня погода», выделив дескрипторы «погода» и «сегодня», система может сформировать ответ: «тепло, но сыро», а на вопрос «какая погода была вчера» — ответ: «ясно, но прохладно». При повторении первого вопроса может быть сформирован ответ: «об этом меня уже спрашивали». При наличии в вопросе неизвестных системе дескрипторов она обычно формирует просьбу: «сформируйте вопрос иначе», а при вторичном непонимании ей остается признаться в своей несостоятельности или перевести разговор в заданное русло.

В настоящее время построено довольно много автоматических диалоговых систем подобного рода. В ряде из них вместо простой идентификации дескрипторов используется грамматический анализ фраз. При достаточно большом словаре такие системы производят впечатление довольно умного и даже интересного собеседника. Однако, не будучи способны обучаться и не имея достаточно мощной семантической сети, они не могут считаться действительно интеллектуальными системами. Их ограниченность обычно легко обнаруживается при изменениях темы разговора и особенно при попытке научить систему какому-нибудь целенаправленному поведению, например какой-нибудь игре.

12.7. Планирование целенаправленных действий

Проблема *планирования целенаправленных действий* может возникать как в *статическом* виде, когда цель в процессе ее достижения не меняется, так и в *динамическом*, когда цель переменна. С первой постановкой мы сталкиваемся, например, когда требуется спланировать движение руки робота, чтобы захватить заданный неподвижный предмет, со второй — когда этот предмет перемещается. В обоих случаях речь идет о нахождении законов изменения некоторых непрерывных параметров. Иными словами, задача планирования возникает здесь в *непрерывной* постановке. Как уже отмечалось в гл. 1, любая непрерывная задача в результате ограниченной точности измерений сводится к *дискретной*, когда характеризующие планируемое действие параметры меняются дискретными шагами.

В чистом виде дискретная постановка задачи планирования целенаправленных действий заключается в нахождении некоторого пути на графе. Более того, задается некоторое конечное множество *состояний* $\{a_1, a_2, \ldots, a_n\}$ (вершин графа) и некоторое множество *элементарных действий* (операторов) $\{\varphi_1, \varphi_2, \ldots, \varphi_k\}$, под воздействием которых состояния могут переходить друг в друга: $\varphi_i a_j = a_k$, где k — некоторая функция от i и j. Предполагается, что в начальный момент времени объект находится в состоянии a_1. Если теперь задать *конечную цель* планируемых действий в виде некоторого множества M состояний a_i, в которые объект требуется перевести, то *планом достижения поставленной цели* считается последовательность $\varphi = \varphi_{i_1}, \ldots, \varphi_{i_l}$ элементарных действий, которая переводит рассматриваемый объект из состояния a_1 в одно из состояний множества M.

Состояния, которые проходит объект при последовательном применении к нему элементарных действий $\varphi_{i_1}, \ldots, \varphi_{i_{l-1}}$, следует рассматривать как *промежуточные цели* при достижении конечной цели φa_1. Поскольку все эти цели так или иначе определяются при решении задачи планирования достижения конечной цели, то такую задачу иногда называют *задачей целеполагания* или, более точно, задачей *промежуточного целеполагания*. Для пояснения постановки и методов решения задачи планирования целенаправленных действий (целеполагания) рассмотрим конкретный пример, в качестве которого выберем известную логическую задачу о волке, козе и капусте. Смысл ее в том, что в лодке, вмещающей, кроме лодочника, лишь один объект (волка, козу или капусту), требуется перевезти три объекта с одного берега на другой. При этом нельзя оставлять на одному берегу без присмотра лодочника, волка с козой, а козу с капустой.

Предположим, что первоначально все три объекта вместе с лодкой находились на правом берегу. Обозначая наличие

объекта единицей, а отсутствие — нулем и располагая объекты в последовательности: волк, коза, капуста, лодка, в начальный момент на правом берегу имеем состояние 1111, а на левом — 0000. Ясно, что всего имеются 16 различных состояний, которые можно изобразить номерами от 0 до 15 так, как это сделано в табл. 12.1. По условиям несовместимости волка с козой, а козы с капустой из этих 16 состояний допустимыми будут лишь 10 состояний, отмеченных в табл. 12.1 звездочками.

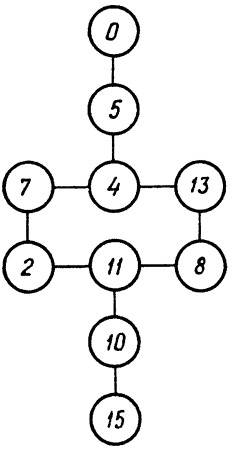

Рис. 12.4

Элементарными действиями будут порожние рейсы лодки, в которых одна последняя единица перемещается с одного берега на другой, а также рейсы с грузом, для которых характерны аналогичные перемещения пар единиц: первой с четвертой, второй с четвертой и третьей с четвертой. Эти действия дают граф (составленный только из допустимых состояний), который изображен на рис. 12.4. Из этого рисунка непосредственно следует, что имеются два кратчайших плана, характеризуемых последовательностями состояний 0, 5, 4, 7, 2, 11, 10, 15 и 0, 5, 4, 13, 8, 11, 10, 15.

В обоих из них первоначально на левый берег перевозится коза (достижение цели 5), затем лодка возвращается на правый берег (цель 4), перевозит на левый берег капусту в первом и волка во втором варианте (цели 7 и 13), возвращается на правый берег с козой (цели 2 и 8), перевозит на левый берег оставшийся на правом берегу объект (цель 11), затем снова совершает пустой рейс на правый берег (цель 10) и, наконец, перевозит на левый берег козу (цель 15).

При нахождении кратчайших путей на графе с помощью ЭВМ наиболее просто реализуется так называемый *алгоритм*

Таблица 12.1

N	Левый берег	Правый берег	N	Левый берег	Правый берег
0*	0000	1111	8*	1000	0111
1	0001	1110	9	1001	0110
2*	0010	1101	10*	1010	0101
3	0011	1100	11*	1011	0100
4*	0100	1011	12	1100	0011
5*	0101	1010	13*	1101	0010
6	0110	1001	14	1110	0001
7*	0111	1000	15*	1111	0000

перебора в ширину. При этом последовательно исследуются вершины графа, отстоящие от начальной вершины a_1 на один *элементарный шаг* (т. е. на шаг, получаемый в результате одного элементарного действия). Будем говорить, что эти вершины непосредственно следуют за вершиной a_1. Затем точно таким же образом последовательно вовлекаются в рассмотрение все новые вершины, непосредственно следующие за найденными на предшествующем шаге. Продолжая этот процесс, рано или поздно мы либо придем к одной из *заключительных* вершин (конечной цели), либо придем в *тупиковое* состояние, когда процесс перестанет порождать новые вершины. Поскольку во втором случае процесс останавливается (стабилизируется), не дойдя ни до одной из заключительных вершин, поставленная задача промежуточного целеполагания оказывается неразрешимой.

Описанный процесс дает полный перебор всех путей на графе в порядке возрастания их длины. Будучи в принципе хорош для машины, он мало подходит для человека, который обычно применяет так называемый *алгоритм перебора в длину*. При этом исследуется до конца (тупика или заключительного состояния) сначала один путь, потом другой и т. д., пока не достигается одно из заключительных состояний либо пока не исчерпаются все пути. Такой алгоритм может применяться (и обычно применяется именно так) без предварительного отсева недопустимых состояний и полного построения связей. Вместо этого граф как бы заново порождается вдоль исследуемых путей от начальной вершины, а каждая вершина, вновь появившаяся на этом пути, проверяется на ее допустимость. В случае допустимости вершины, если задача ею не решается, процесс порождения новых вершин продолжается далее; в противном случае последнее из примененных элементарных действий заменяется на другое и процесс исследования продолжается далее.

Алгоритм перебора в длину, как и алгоритм перебора в ширину, может быть легко реализован на ЭВМ, однако уже без гарантии того, что первое же найденное решение представляет собой наилучший (т. е. наикратчайший) план. Для рассмотренного примера оба эти алгоритма быстро приводят к нахождению решения (причем первый алгоритм находит оба решения, а второй — одно из них).

Дальнейшие усовершенствования алгоритмов поиска путей на графе связаны с введением функций, характеризующих степень близости вершин к множеству M заключительных вершин. Используя такие функции, можно организовать более целенаправленный поиск (особенно в случае перебора в длину). Выбор очередной достижимой подцели (следующей по порядку верши-

ны графа, добавляемой к исследуемому пути) производится в первую очередь по принципу максимизации ее близости к заключительному множеству M. Подобные *стратегии целенаправленного перебора* могут, разумеется, заводить и в тупик, для выхода из которого надо пойти на временное удаление очередной вершины от множества M. В рассмотренном примере это делается, когда коза, уже перевезенная на нужный (левый) берег, временно возвращается назад.

Сложность задачи поиска кратчайшего пути к цели при использовании различного рода *оценочных функций* для промежуточных целей определяется в первую очередь количеством и глубиной подобного рода тупиков. По мере исследования подобных тупиков в алгоритме перебора в длину производится размечивание ведущих к ним дуг и вершин графа (т. е. подпутей), с тем чтобы в процессе поиска не повторять заходов в те же самые тупики. Заметим, что при наличии оценочных функций задача нахождения минимального пути на графе, ведущего к заданной статической цели, обычно сводится к задачам дискретной оптимизации, о методах решения которых мы достаточно подробно говорили выше (§ 8.8).

Большое значение в сложных задачах планирования целенаправленного поведения могут приобретать различного рода *эвристические приемы* (обычно формируемые в результате приобретенного опыта) постановки достаточно крупных промежуточных целей и применения описанных выше методов достижения целей применительно не к конечным, а именно к этим промежуточным целям. Именно так поступает, например, шахматист, ищущий пути к выигрышу партии. Нередко при этом приходится осуществлять поиск на одном и том же графе, но с различными начальными и заключительными состояниями. В этом случае обычно производится последовательное изучение этого графа, запоминание удачных эвристических приемов и просто участков путей (или алгоритмов их нахождения), ведущих к требуемой цели. Последнее в случае шахматной игры приобретает вид запоминания стандартных алгоритмов для выигрыша (или достижения ничьей) в различных видах эндшпилей или в некоторых классах позиций более ранних стадий игры. Кроме того, в таких случаях полезно запоминание возможно большего числа начальных участков путей (дебютов), ведущих нас в желательном направлении.

12.7.1. Игровые задачи. В игровых задачах (например, в шахматах) участники обычно преследуют различные и даже прямо противоположные цели. Процесс проведения игры — это процесс последовательного разворачивания некоторого пути на графе всех возможных состояний игры. При парной игре ее участники, делая ходы поочередно, на каждом ходе пытаются повернуть

путь в сторону «своего» заключительного множества. Поэтому, решая задачу целенаправленного планирования, каждый игрок должен планировать ходы и за себя и за противника. В сложных играх (какими являются, например, шахматы) методы полного перебора (обычно путем использования перебора в ширину) могут работать на относительно небольшую глубину (обычно порядка 3—5 ходов с каждой стороны). Поскольку глубина дерева игры обычно значительно больше, используют оценочные функции, оценивающие анализируемые заключительные (для данного уровня перебора) позиции и выбирают очередной ход таким образом, чтобы при любых возможных продолжениях минимальное значение целевой функции («промежуточного выигрыша» игрока, делающего ход) было бы максимальным.

Использование подобной нехитрой стратегии при достаточно удачном выборе целевой функции (функции оценки позиции) может приводить к неплохим практическим результатам, особенно, если при этом используются описанные выше приемы, основанные на запоминании схем действия в «стандартных» позициях. Так, использующая этот метод шахматная программа «Bell» (США), реализованная на специализированном шахматном процессоре, достигла в 1980 г. по силе своей игры уровня кандидата в мастера. Ее рейтинг (основанный на результатах игры с людьми) достиг 2150, что намного превышает оценку силы среднего шахматиста. Следует подчеркнуть, что использование специального шахматного процессора (автомата, моделирующего шахматную доску и ходы различных фигур) повысило скорость анализа шахматных позиций в сотни раз (по сравнению с обычными универсальными ЭВМ), что дало возможность увеличить глубину перебора по сравнению с другими шахматными программами минимум на один ход (два полухода).

Идея полного перебора по своей сущности глубоко антиинтеллектуальна: ведь человек (в данном случае шахматист) обычно так никогда не поступает. Применение же к сложным играм, вроде шахмат, процедур целенаправленного перебора при обычно принятых способах построения оценочных функций ведет к возникновению большого числа тупиков, что в конце концов по существу тоже возвращает нас к полному перебору. В случае шахмат в качестве оценочной функции позиции выбирается обычно функция, равная разности сумм *постоянных* весов фигур каждого игрока, дополненная в лучшем случае компонентами, оценивающими такие позиционные элементы, как слабые поля, наличие сдвоенных пешек, открытых линий и т. п. При таком выборе оценочной функции всякая комбинация, связанная с *временной* жертвой материала, ведет сначала к ухудшению оценки, т. е. представляет собой типично тупиковую ситуацию.

М. Ботвиннику принадлежит идея разбивки постановки целей в шахматной игре на цели отдельных фигур и изменения в зависимости от их вклада в общую цель относительной ценности фигур. Тем самым оценочная функция приобретает *динамический характер*, т. е. меняется при изменении достигаемой конкретной цели (на данный момент игры). Такой подход позволяет резко уменьшить количество тупиковых ситуаций и тем самым резко сократить перебор (а иногда и вовсе устранить), находя ходы, улучшающие целевую функцию на каждом шаге.

12.7.2. Динамическое целеполагание. Случай, когда достигается цель, разбивается на два основных случая. В первом из них цель меняется независимо от действий, предпринимаемых для ее достижения. В этом случае планирование достижения цели разбивается на две задачи: определения закона изменения цели $M(t)$ (множества заключительных состояний) и собственно планирования действий для ее достижения. Первая задача сводится фактически к нахождению закона изменения цели по нескольким наблюдениям и соответствующей экстраполяции на его основе положения цели в каждый последующий момент времени. Методы решения этой задачи были рассмотрены в § 9.3. Что же касается задачи планирования достижения цели, то каждый шаг в поиске плана должен сопровождаться оценкой времени, необходимого на этот шаг (равно как и на все предшествующие шаги) при последующей реализации плана. Поэтому, например, в методе перебора в длину (глубину) с каждой очередной исследуемой вершиной a_i будет связана оценка времени t_i ее достижения при реализации плана. Задача будет решена, если $a_i \in M(t_i)$.

Практически задача экстраполяции решается чаще всего не заблаговременно, а в процессе фактического достижения цели. При этом после выполнения очередного шага по достижению цели уточняется как экстраполяция (в результате появления новых наблюдений), так и, вообще говоря, план достижения цели (поскольку может измениться описание закона ее изменения). Подобный метод *динамического уточнения* плана достижения цели применим и во втором случае, когда цель меняет свое поведение в зависимости от наших действий. Таким способом могут решаться задачи динамического целеполагания в игровых постановках, изучаемых в так называемой теории *дифференциальных игр* (в дискретном случае — *разностных*), и прежде всего — так называемые *задачи погони* в обычном евклидовом пространстве.

12.8. Автоматизация дедуктивных построений

Дедуктивные построения разбиваются на два класса задач. Первый из них связан с *постановкой проблем*, т. е. с формированием *высказываний*, справедливость (истинность) или ошибочность (ложность) которых нужно доказать. Поиск самого *доказательства* (в случае истинности высказывания) или *опровержения* (в случае его ложности) носит наименование *логического вывода*. При этом предполагается, что в процессе доказательства (опровержения) могут использоваться лишь ранее приобретенные *формализованные знания*, а не новые эксперименты и наблюдения. Именно это обстоятельство подчеркивается применением термина *дедуктивный* к рассматриваемым нами построениям. Формализованные знания могут заключаться как в высказываниях, истинность которых была установлена ранее (теоремы) либо постулируется априори (аксиомы), так и в *процедурах* (алгоритмах), позволяющих строить доказательства и опровержения. Для сокращения описаний высказываний и процедур в систему формализованных знаний обычно включаются также *определения* различных понятий, которые могут использоваться в формулировках.

Формализация знаний, о которой идет речь в данном случае, заключается в их организации в так называемое *исчисление*. Основой всякого исчисления является прежде всего формальный язык, с помощью которого записываются предложения данного исчисления, называемые обычно *формулами* данного исчисления. Вторая часть исчисления — *процедуры* (правила) *вывода*.

При автоматизированном выводе они могут представляться либо непосредственно в виде машинных программ, либо в виде алгоритмов, записанных на том или ином входном языке программирования. Во втором случае в состав автоматизированной системы должна включаться соответствующая *программа-интерпретатор*, позволяющая применять эти алгоритмы для фактического вывода (или опровержения) предъявляемых системе предложений (формул исчисления) из уже имеющихся в ней предложений.

12.8.1. Исчисление высказываний. Простейшее из исчислений, называемое *исчислением высказываний* (или *пропозициональным исчислением*), оперирует с высказываниями, построенными из *элементарных высказываний* с помощью операций этого исчисления. В качестве таких операций в исчислении высказываний применяются уже рассматривавшиеся в гл. I элементарные *булевы операции* (см. п. 1.5.1): *конъюнкция* ($\wedge$), *дизъюнкция* ($\vee$), отрицание ($\neg$) и еще так называемая операция *импликации* ($\rightarrow$). Элементарные высказывания мы будем обо-

значать буквами латинского алфавита (с индексами или без них) и рассматривать как булевы переменные, принимающие лишь два значения: «истина» (сокращенно «и» или 1) и «ложь» (сокращенно «л» или 0). Значения функции истинности для конъюнкции, дизъюнкции и отрицания были приведены в п. 1.5.1. Для импликации полагаем, что значения $0 \to 1$, $0 \to 0$, $1 \to 1$ функции $A \to B$ являются истинными, а значение $1 \to 0$ — ложным.

В число процедур вывода исчисления высказываний мы будем включать эквивалентные преобразования формул исчисления, являющиеся результатом последовательного выполнения элементарных соотношений вида

$$x \to y = (\neg x) \vee y, \quad x \vee y = y \vee x, \quad x \wedge y = y \wedge x,$$
$$\neg(\neg x) = x, \quad \neg(x \vee y) = (\neg x) \wedge (\neg y), \quad \neg(x \wedge y) = (\neg x) \vee (\neg y),$$
$$x \vee (y \vee z) = (x \vee y) \vee z, \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z.$$

Благодаря двум последним соотношениям (называемым правилами *ассоциативности*) можно конъюнкции (равно как и дизъюнкции) любого числа членов писать без скобок. Благодаря двум соотношепиям перед ними, называемым *правилами де Моргана*, можно все отрицания довести до элементарных высказываний. Еще два соотношения $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ и $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$, называемые правилами *дистрибутивности*, вместе с остальными соотношениями могут привести любую форму исчисления высказываний к *дизъюнктивной* или *конъюнктивной форме*. Первая из них представляет дизъюнкцию $K_1 \vee K_2 \vee \ldots \vee K_m$ конечного числа *конъюнктивных термов* K_i, а вторая — конъюнкцию $D_1 \wedge D_2 \wedge \ldots \wedge D_n$ конечного числа *дизъюнктивных термов* D_j. Конъюнктивный (соответственно дизъюнктивный) терм представляет собой конъюнкцию (соответственно дизъюнкцию) *элементарных термов*, в качестве которых могут выступать элементарные высказывания (булевы переменные) или их отрицания.

Учитывая соотношения (правила) коммутативности $x \vee y = y \vee x$ и $x \wedge y = y \wedge x$ для дизъюнкции и конъюнкции, а также соотношения $x \vee x = x \wedge x = x$, $(\neg x) \vee x = 1$, $(\neg x) \wedge x = 0$, $1 \vee x = 1$, $0 \vee x = x$, $1 \wedge x = x$, $0 \wedge x = 0$, можно добиться, чтобы в одном и том же конъюнктивном или дизъюнктивном терме не было одинаковых элементарных термов, равно как и пар $(x, \neg x)$. Дизъюнктивные (конъюнктивные) формы с этим дополнительным условием принято называть *нормальными формами* (соответственно д. н. ф. и к. н. ф.).

Если задан набор элементарных высказываний $A_1, A_2, \ldots$, то задача постановки проблем в исчислении, построенном на их базе, состоит в генерации формул этого исчисления и решается

весьма простым (очевидным) алгоритмом. Задача логического вывода состоит в том, чтобы определить, к какому из трех возможных классов принадлежит рассматриваемая формула. Первый класс составляют так называемые *тождественно истинные формулы* (теоремы исчисления). Они характеризуются тем, что для любых комбинаций значений истинности входящих в них элементарных высказываний они оказываются всегда истинными. Второй класс составляют *тождественно ложные формулы*, ложные при любых возможных комбинациях значений входящих в них элементарных высказываний. Наконец, класс *выполнимых формул* включает в себя все формулы, истинные хотя бы при одной комбинации значений входящих в них элементарных высказываний. Все тождественно истинные формулы являются выполнимыми, но обратное, разумеется, не имеет места.

Легко проверить, что формула $A \to A$ тождественно истинна, формула $\neg(A \to A)$ тождественно ложна, а формула $(\neg A) \to A$ выполнима, но не тождественно истинна (она истинна при $A = 1$ и ложна при $A = 0$). Проверка на тождественную истинность (ложность) или выполнимость всегда может быть выполнена фактическим вычислением ее значений при всех возможных комбинациях значений входящих в нее элементарных высказываний. Если число таких высказываний равно n, то число различных комбинаций их значений равно, очевидно, 2^n, т. е. при больших значениях может становиться непомерно большим. Поэтому, хотя процедура прямого вычисления значений формулы при всех таких комбинациях дает решение вопроса, является ли заданная формула теоремой исчисления высказываний или нет (т. е. является одной из возможных процедур логического вывода), на практике она почти не применяется.

Одной из наиболее удобных процедур логического вывода в исчислении высказываний является процедура, основанная на так называемом *принципе резолюции*. В этой процедуре, которую мы будем называть *R-процедурой*, вместо заданной формулы F, предполагаемой тождественно истинной, рассматривают ее отрицание $\neg F$ и пытаются доказать ее противоречивость (тождественную ложность). Это так называемое *доказательство от противного*. В процедуре требуемая противоречивость (если она действительно имеет место) устанавливается как одновременная справедливость двух исключающих друг друга высказываний L и $\neg L$, как это имеет место и в обычном варианте доказательства от противного.

Для этой цели формулу $\neg F$ приводят к конъюнктивной нормальной форме $\neg F = D_1 \wedge \ldots \wedge D_n$. Принцип резолюции состоит в поиске двух дизъюнктивных термов D_i и D_j, в один из которых входит какой-либо элементарный терм A, а во второй — его отрицание $\neg A$. Тогда, если $D_i = B \vee A$, а $D_j =$

$= C \vee (\neg A)$, то из термов D_i и D_j выводится новый терм $D_{n+1} = B \vee C$, называемый *резольвентой* членов D_i и D_j, который может быть добавлен к конъюнктивной нормальной форме $D_1 \wedge D_2 \wedge \ldots \wedge D_n$. Разумеется, такое расширение формы новым членом D_{n+1} следует делать лишь в том случае, когда этот член ранее отсутствовал в рассматриваемой нормальной форме.

Процесс *вывода* (резолюции) новых членов продолжается до тех пор, пока новых членов получаться уже не будет (это всегда возможно ввиду конечности числа таких членов). При этом возможны два случая. В первом из них мы закончим процесс, получив тождественно ложную резольвенту из двух взаимно противоречивых термов $D_k = C$ и $D_l = \neg C$. В этом случае полученное противоречие доказывает тождественную ложность формулы $\neg F$ и, следовательно, тождественную истинность исходной формулы F. Если же процесс вывода резольвент оборвется, не приведя к противоречию, то формула $\neg F$ оказывается выполнимой, а следовательно, формула F не может быть тождественно истинной. Иными словами, высказанная теорема неверна.

Применим R-процедуру к доказательству тождественной истинности формулы $F = (A \wedge B) \rightarrow A$. Предположим противное: $\neg F = \neg((A \wedge B) \rightarrow A) = \neg(\neg(A \wedge B) \vee A) = (A \wedge B) \wedge (\neg A) = (A) \wedge (B) \wedge (\neg A)$. Поскольку имеется противоречие (термы A и $\neg A$), то теорема доказана.

Преимущество R-процедуры заключается в том, что при надлежащем выборе последовательных резолюций противоречие может быть выявлено (и, следовательно, исходная теорема доказана) намного раньше, чем исчерпывается возможность выполнения новых продуктивных резолюций (поставляющих новые конъюнктивные термы). Поэтому такая процедура обычно намного экономнее упомянутой выше процедуры прямого вычисления значений исследуемой формулы при всех значениях переменных. Нахождение кратчайшего пути к доказательству сводится к задаче нахождения кратчайшего пути на графе. Однако, поскольку в этом случае этап планирования слит в единое целое с этапом реализации плана, поиск лучшего плана должен быть целенаправленным: от добытого с большими трудностями оптимального плана (например, с помощью метода перебора в ширину) в подобной ситуации мало пользы. Разработано много эвристических правил (стратегий), позволяющих улучшить целенаправленность перебора в R-процедуре. Например, при наличии в конъюнктивной нормальной форме одночленных дизъюнктивных термов целесообразно начинать резолюции именно с них. Недостаток места не позволяет нам более подробно остановиться на этом вопросе.

Заметим еще, что существует двойственная процедура, когда прямо доказывается (или опровергается) тождественная ис-

тинность исходной формулы F. С этой целью формула F приводится к дизъюнктивной нормальной форме $F = K_1 \vee \ldots \vee K_n$ и вводится в действие правило *двойственной резолюции*, порождающее новый конъюнктивный терм $K_{m+1} = B \wedge C$ из двух конъюнктивных термов $K_i = A \wedge B$ и $K_j = (\neg A) \wedge C$. Как и в прямой R-процедуре, процесс заканчивается доказательством тождественной истинности формулы F при выполнении (двойственной) резолюции вида $(A) \vee (\neg A) \to 1$. Возникновение подобной ситуации означает, что в дизъюнктивной нормальной форме $K_1 \vee \ldots \vee K_m$ появляется терм $A \vee (\neg A)$, который при любом значении переменной A (элементарного высказывания) тождественно равен единице, что, очевидно, достаточно для тождественной истинности всей формы, а следовательно, и исходной формулы F.

Для рассмотренного выше примера применение двойственной процедуры дает $F = (A \wedge B) \to A = \neg(A \wedge B) \vee B = (\neg A) \vee \vee(\neg B) \vee B = 1$. Тем самым тождественная истинность формулы F доказана.

12.8.2. Узкое исчисление предикатов и формальные теории. В основу *узкого исчисления предикатов* кладется некоторая *предметная область* или множество M объектов x_1, x_2, ..., на которых определены функции (предикаты) $P(x_{i_1}, \ldots, x_{i_n})$, значениями которых является истина или ложь.

Предикаты представляют собой высказывания. Поэтому, если дано некоторое множество *первичных предикатов* P_1, P_2, ..., P_m, то над ним может быть построено исчисление высказываний, каждая формула которого может рассматриваться как *составной предикат*; число *предметных* (содержательных) переменных x_i, от которых зависит предикат, называется его *местностью*. Мы будем различать таким образом *нуль-местные* предикаты P (не зависящие от предметных переменных), *одноместные* предикаты $P(x)$, *двуместные* $P(x, y)$ и т. д.

Заметим, что в узком исчислении предикатов все первичные предикаты (в том числе и нуль-местные) представляют собой конкретные высказывания и сами по себе, без учета предметных переменных, от которых они зависят, не могут рассматриваться как переменные. В частности, нуль-местные предикаты P, Q, ... в узком исчислении предикатов всегда рассматриваются как конкретные *постоянные высказывания*.

Помимо операций исчисления высказываний, в узкое исчисление предикатов вводятся операции *связывания* предметных переменных так называемыми *кванторами*. Различают *квантор всеобщности* $\forall x$ (для *всех* x) и *квантор существования* $\exists x$ (существует такое x). Кванторы ставятся обычно впереди предиката, составляя так называемую *кванторную приставку*. С такой приставкой предикат превращается, вообще говоря, в новое вы-

сказывание. Например, выражение $\forall x \exists y P(x, y)$ означает высказывание «для всякого x существует такое y, что имеет место $P(x, y)$». Это высказывание можно, разумеется, также рассматривать как предикат (в приведенном примере как нульместный), в котором из числа *свободных* предметных переменных исключены переменные, *связанные* кванторами.

Формулы исчисления предикатов, не содержащих свободных предметных переменных, делятся на *истинные* (теоремы исчисления) и *ложные*. Например, формула $\forall x (P(x) \to P(x))$ истинна, а формула $\forall x ((\neg P(x)) \to P(x))$ ложна для любого предиката $P(x)$, принимающего хотя бы при одном значении x значение «ложь».

Предметная область M, заданная абстрактно в рамках собственно логического исчисления, может конкретизироваться при приложениях этого исчисления. Таким образом, на базе данного логического исчисления возникают те или иные *формальные теории*. Например, если M есть множество чисел $\{0, 1, 2, \ldots\}$, на котором заданы обычные арифметические операции (в случае, когда они выполнимы) и обычные арифметические отношения (первичные предикаты) $=$, $\neq$, $>$, $<$, $\geqslant$, $\leqslant$, то с помощью добавления формализмов узкое исчисление предикатов превращается в *формальную арифметику* неотрицательных чисел, если выделить некоторое нужное множество высказываний в качестве аксиом и задать соответствующую процедуру логического вывода следствий из этих аксиом.

Задание аксиом, разумеется, определяется видом рассматриваемой теории. Что же касается логического вывода, то он включает в себя прежде всего процедуры общелогического характера (применимые ко всякой формальной теории). Помимо уже описанных процедур вывода в исчислении высказываний, в исчислении предикатов возникают дополнительные операции с кванторами.

Прежде всего опишем, каким образом производится вынесение кванторов в начало формулы. Если имеет место выражение $K(P(\ldots, x, \ldots)) \vee \forall x (K'(Q(\ldots, x, \ldots)))$, где K и K' — произвольные кванторные приставки, то оно преобразуется в эквивалентное ему выражение $\forall x (\widetilde{K}(P(\ldots, \tilde{x}, \ldots)))$, в котором вхождения переменной x, не охваченные выносимым за скобки квантором, заменяются на новую переменную $\tilde{x}$, отличную от всех остальных переменных формулы. Тот же самый прием применяется и при вынесении квантора $\exists x$. Он же действует, если выносится за скобки первый квантор приставки K или когда вместо дизъюнкции $\vee$ в формуле участвует конъюнкция $\wedge$. При вынесении кванторов $\forall x$, $\exists x$ за знак отрицания они переходят один в другой: $\neg \forall x A$ эквивалентно $\exists x (\neg A)$, а $\neg \exists x A$ эквивалентно $\forall x (\neg A)$. Вынесению кванторов за знак имплика-

ции можно предварить ее выражение через операции отрицания и дизъюнкции, так что особых правил для действий с нею можно не вводить.

После вынесения кванторов к оставшейся бескванторной формуле могут быть применены процедуры приведения к конъюнктивной и дизъюнктивной нормальным формам и использованы различные обобщения принципа резолюции для доказательства или опровержения построенных таким образом высказываний. Мы, однако, не будем останавливаться на этих обобщениях, а опишем другой подход к реализации логического вывода в логических теориях, основанный на использовании так называемых функций Сколема.

12.8.3. Функции Сколема. Начнем рассмотрение с частного случая. Выражение $\forall x \exists y P(x, y)$ узкого исчисления предикатов можно трактовать таким образом, что существует *предметная функция* $y = f(x)$, определенная на всех x и такая, что при ее подстановке в предикат $P(x, y)$ он превращается в высказывание $F(x) = P(x, f(x))$, истинное для всех x. Эта функция, позволяющая устранить кванторную приставку, называется *функцией Сколема* для исходного выражения $\forall x \exists y P(x, y)$. Если квантор существования стоит впереди: $\exists y \forall x P(x, y)$, то функция Сколема сведется к константе f, а исходное высказывание — к высказыванию $F'(x) = P(x, f)$. При сложных кванторных приставках число соответствующих им функций Сколема будет равняться числу вхождений в приставки кванторов существования. Процесс сворачивания кванторов и выписывания соответствующих функций Сколема идет слева направо. При этом переменные, связанные квантором всеобщности, добавляются в список переменных (вначале пустой), от которых будут зависеть все последующие функции Сколема. Каждая же переменная y, связанная квантором существования, порождает очередную функцию Сколема $y = f(x_{i_1}, \ldots, x_{i_k})$, зависящую от всех переменных, включенных в список на данный момент. Например, для выражения $F = \forall x \exists y \exists z \forall u \forall v \exists w P$ (x, y, z, u, v, w) порождаются три функции Сколема $y = f(x)$, $z = g(x)$, $w = h(x, u, v)$, а выражение приводится при этом к бескванторному виду: $F = G(x, u, v) = P(x, f(x), g(x), u, v, h(x, u, v))$.

12.8.4. Метод логического вывода, основанный на синтезе функций Сколема. Каждое верное предложение (аксиома или теорема) любой данной формальной теории может быть представлено (в соответствии с описанной в предыдущем пункте процедурой) совокупностью порождаемых им функций Сколема и тождественно истинным бескванторным предикатом. Рассмотрим процесс последовательного развертывания формальной теории в результате присоединения к ее *первоначальной базе знаний* (состоящей из аксиом) новых верных предложений по мере

их доказательства. *Текущая база знаний* теории будет включать в себя, таким образом, некоторое конечное множество S конкретных (определенных в аксиомах или построенных в ходе доказательств теории) функций Сколема $f_1, \ldots, f_m$ и некоторое конечное множество T конкретных тождественно истинных предикатов $P_1, \ldots, P_n$. Сюда нужно добавить различные приемы (алгоритмы) порождения различных функций (в порядке роста их сложности) в заданной предметной области, например, для арифметики натуральных чисел, функций $f(x) = x + 1$, $f(x, y) = x + y$ и т. д. Такие функции мы будем называть *независимо порождаемыми*.

Пусть в этом состоянии теории предъявляется новое предложение F (не содержащее свободных переменных), которое требуется доказать. Приводя это предложение к бескванторному виду, получаем множество пока неизвестных функций Сколема $g_1, \ldots, g_k$ и бескванторный предикат P, в состав которого входят эти функции. Предложение F будет доказано, если мы построим представления неизвестных функций $g_1, \ldots, g_k$ в виде *суперпозиции* известных функций $f_1, \ldots, f_m$ и независимо порождаемых функций $f, g, h, \ldots$ так, чтобы предложение $Q = (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \rightarrow P$ было *выводимым* (т. е. являлось тождественно истинной формулой) в исчислении высказываний. Это именно тот способ, каким математик строит свои доказательства. Заметим лишь, что, поскольку истинность предложения F априори неизвестна, описанные манипуляции производятся обычно не только с предложением F, но и с его отрицанием $\neg F$. После окончания доказательства использовавшиеся для него функции и предикат добавляются к текущей базе знаний.

Разумеется, на практике в доказательстве используется лишь часть (притом обычно небольшая) накопленных в теории функций f_i и предикатов P_j. Выбор нужных для доказательства функций и предикатов в обычном (не автоматизированном) доказательстве определяется интуицией математика. При автоматизации на смену интуиции приходят разнообразные эвристические правила, разработка которых находится еще в самом начале своего развития. Поэтому целесообразно развивать (особенно на первых порах) наряду с чисто автоматическими диалоговые (человеко-машинные) методы логического вывода. Основная трудность здесь — в создании эффективных языков для такого диалога, с тем чтобы человек понимал трудности, возникающие у машины, и вовремя давал необходимые подсказки. В этом смысле подход к выводу на основе функций Сколема наиболее эффективен, поскольку он фактически моделирует мышление математика.

На базе этого подхода автором разработаны принципы построения языка описания знаний и языка диалога, относящих-

ся к классическому языку узкого исчисления предикатов примерно так, как развитые входные языки программирования относятся к языкам, описывающим поведение простейших автоматов — так называемых *машин Тьюринга.* Важно также подчеркнуть, что в принятом подходе основной (и наиболее трудный) этап доказательства — синтез функций Сколема — делается на основе «крупноблочного» метода, использующего не только аксиомы теории, но и все накопленное в ней богатство знаний. Благодаря этому конструкции суперпозиций получаются относительно несложными («малоэтажными», или, лучше сказать, «малозвенными»), что упрощает их нахождение обычными методами перебора (с относительно простыми эвристиками).

12.8.5. Ограниченные кванторы. На практике относительно редко случается, чтобы предметная область была однородной. В нее обычно включаются объекты различной природы (множества, числа, функции и т. п.). Кроме того, даже внутри одного и того же класса объектов с помощью специальных предикатов, которые мы будем называть *определяющими* или *ограничивающими*, можно выделять удовлетворяющие им объекты (например, конечные множества, простые числа и т. п.). Поэтому в формальных теориях обычно используются *ограниченные кванторы*, которые с помощью *системы* ограничивающих предикатов (включающих в себя предикаты принадлежности объектов к тем или иным классам) специфицируют области определений связываемых или предметных переменных.

Мы будем обозначать ограниченные кванторы символами $\forall_S$ и $\exists_Q$, где через S и Q обозначены *спецификации кванторов*, т. е. *системы* ограничивающих их предикатов. При описанных в п. 12.8.2 операциях над кванторами (в частности, при вынесении кванторов за знак отрицания) эти спецификации не меняются. В процессе логического вывода принадлежащие спецификациям предикаты добавляются к текущему базису рассматриваемой формальной теории. При построении же суперпозиций необходимо дополнительно проверять их допустимость. Это означает, что при подстановке вместо некоторой переменной y предметной функции $y = f(x_{i_1}, \ldots, x_{i_l})$ область значений функции должна входить в область определения переменной y. Сами же эти области задаются спецификациями кванторов, связывающих соответствующие переменные в исходных предложениях, а для независимо порождаемых функций — описаниями задающих их алгоритмов.

В результате проверок на допустимость суперпозиций могут возникать дополнительные предложения, которые требуется доказать прежде, чем будет доказано исходное предложение. Тем самым происходит автоматическое размножение целей логического вывода.

12.8.6. Современное состояние автоматизации дедуктивных построений. Что касается постановки проблем, то автоматизация этого процесса в принципе не составляет особых трудностей. Трудно лишь выделить из порождаемых машиной проблем действительно интересные проблемы. Некоторые подходы к решению этой задачи уже наметились, однако в целом проблема все еще далека от решения.

Логический вывод в рамках простейших логических исчислений (особенно в исчислении высказываний) машины выполняют сегодня гораздо быстрее и надежнее человека. Что же касается содержательных (внелогических) разделов математики, то здесь успехи автоматизации пока еще невелики. Правда, уже есть примеры, когда ЭВМ помогла решить до конца некоторые трудные математические проблемы (например, знаменитую проблему четырех красок). Однако эти впечатляющие успехи достигнуты в результате составления программ, специально предназначенных для решения данных конкретных проблем, и не годятся в других случаях.

Развитие же универсальных автоматических систем логического вывода пока делает свои первые шаги.

ЗАКЛЮЧЕНИЕ

Безбумажная информатика развивается исключительно быстрыми темпами. Во всех развитых странах темпы увеличения числа ЭВМ, АСУ, терминалов и особенно суммарной производительности ЭВМ и объемов накопленной в них информации резко опережают темпы роста всех других показателей, характеризующих экономику и научно-технический прогресс. Сращивание средств телекоммуникации с машинной информатикой (реализующихся в сетях ЭВМ и ВЦ с удаленными терминалами) уже привело к появлению нового термина *телематика.* Наиболее рьяные апологеты телематики предсказывают, что уже недалек тот день, когда исчезнут обычные книги, газеты и журналы. Взамен каждый человек будет носить с собой «электронный блокнот», представляющий собой комбинацию плоского дисплея с миниатюрным радиоприемопередатчиком. Набирая на клавиатуре этого «блокнота» нужный код, можно (находясь в любом месте на нашей планете, вызвать из гигантских компьютерных баз данных, связанных в сети, любые тексты, изображения (в том числе и динамические), которые и заменят не только современные книги, журналы и газеты, но и современные телевизоры.

Зайдет ли дело в обозримом будущем столь далеко — гадать трудно. Несомненно одно, что прогресс электронной технологии, машинной информатики и телематики происходит столь бурными темпами, что фантастика в этой области становится реальностью буквально на наших глазах.

В. М. Глушков

СПИСОК ЛИТЕРАТУРЫ

1. Айцман П. З. С и ЦМД ЗУ — новые компоненты для высококачественных компонент памяти//Электроника.— 1979.— № 8.
2. Акинфиев А. Б. и др. Развитие запоминающих устройств в ЕС ЭВМ//Вопросы радиоэлектроники. Сер. ЭВТ.— 1980. Вып. 3.
3. Амосов Н. М. и др. Автоматы и разумное поведение.— Киев: Наукова думка, 1973.
4. Амосов Н. М. Алгоритмы разума.— Киев: Наукова думка, 1979.
5. Байцер Б. Архитектура вычислительных комплексов.— М.: Мир, 1974.— Т. 1.— 468 с.; Т. 2.— 566 с.
6. Библиография работ по технологии программирования//Технология программирования.— Киев: ИК АН УССР, 1979.— С. 52—70.— (Тезисы докл. 1 Всес. конф.)
7. Бонгард М. М. Проблема узнавания.— М.: Наука, 1967.
8. Ботвинник М. М. О кибернетической цели игры.— М.: Сов. радио, 1975.
9. Ботвинник М. М. О решении неточных переборных задач.— М.: Сов. радио, 1979.
10. Ботев М. и др. Перспективные внешние ЗУ на магнитных дисках для мини ЭВМ // Приборы и системы управления.— 1980 — № 6.
11. Брусенцов Н. П. Миникомпьютеры.— М.: Наука, 1979.— 272 с.
12. Вельбицкий И. В., Ходаковский В. Н, Шоломов Л. И. Технологический комплекс программиста на машинах ЕС ЭВМ и БЭСМ-6.— М.: Статистика, 1980.— 263 с.
13. Вирт Н. Систематическое программирование.— М.: Мир, 1977.— 183 с.
14. Вопросы проектирования преобразователей формы информации/Под ред. Кондалева А. И.— Киев: Наукова думка, 1977.— 244 с.
15. Вычислительная машина и мозг//Кибернетический сб. № 1. М.: ИЛ, 1960.— С. 11—60.
16. Гладкий А. В. Синтаксические структуры естественного языка в автоматизированных системах общения.— М.: Наука, 1985.— 144 с.
17. Гаазе—Раппопорт М. Г. Автоматы и живые организмы.— М.: Физматгиз.— 1961.
18. Глушков В. М. Введение в АСУ.— Киев: Техника, 1974.— 317 с.
19. Глушков В. М. Введение в кибернетику.— Киев: Изд-во АН УССР, 1964.— 322 с.
20. Глушков В. М. Макроэкономические модели и принципы построения ОГАС.— М.: Статистика, 1975.— 160 с.
21. Глушков В. М. Синтез цифровых автоматов.— М.: Физматгиз, 1962.
22. Глушков В. М., Барабанов А. А., Калиниченко Л. А. и др. Вычислительные машины с развитыми системами интерпретации.— Киев: Наукова думка, 1970.— 259 с.
23. Глушков В. М., Игнатьев М. Б. и др. Рекурсивная машина.— Препринт/ИК АН УССР.— Киев, 1974.— № 74—57.

24. Глушков В. М., Калиниченко Л. А. и др. Сети ЭВМ.— М.: Радио и связь, 1977.— 279 с.
25. Глушков В. М., Капитонова Ю. В., Летичевский А. А. Автоматизация проектирования вычислительных машин.— Киев: Наукова думка, 1975.
26. Глушков Б. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра, языки, программирование.— Киев: Наукова думка, 1974.— 328 с.
27. Головкин Б. А. Параллельные вычислительные системы.— М.: Наука, 1980.— 519 с.
28. Голографические ИПС//Информационные процессы и системы. Сер 2.— 1980.— № 1. С. 7—11.
29. Горелик А. Г. Автоматизация инженерно-графических работ с помощью ЭВМ.— М.: Высшая школа, 1980.— 208 с.
30. Дал У., Дейкстра Э., Хоар К. Структурное программирование.— М.: Мир, 1978.— 275 с.
31. Дейкстра Э. Дисциплина программирования.— М.: Мир, 1975.— 247 с.
32. Дроздов Э. А., Комарницкий В. А., Пятибратов А. Я. Электронные вычислительные машины единой системы.— М.: Машиностроение, 1976.
33. Единая система ЭВМ/Под ред. Ларионова А. М.— М.: Статистика, 1974.— 136 с.
34. Ермольев Ю. М. Методы стохастического программирования.— М.: Наука, 1976.
35. Йодан Э. Структурное проектирование и конструирование программ.— М.: Мир, 1979 — 410 с.
36. Информационные системы общего назначения/Под ред. Ющенко Е. Л.— М.: Статистика, 1975.— 472 с.
37. Каган Б. М. Электронные вычислительные машины и системы.— М.: Энергия, 1979.
38. Канторович Л. В., Гостко А. Б. Оптимальные решения в экономике.— М.: Наука, 1972.— 232 с.
39. Кейпас А. ЗУ : Обзор//Электроника.— 1978.— № 22.
40. Кликов Ю. И., Горьков Л. Н. Банки данных для принятия решений.— М.: Сов. радио, 1980.— 208 с.
41. Кнут Д. Искусство программирования.— М.: Мир.— Т. 1: Основные алгоритмы.— 1976.— 735 с.; Т. 2; Получисленные алгоритмы.— 1977.— 725 с.; Т. 3: Сортировка и поиск.— 1978.— 845 с.
42. Королев Л. Н. Структуры ЭВМ и их математическое обеспечение.— М.: Наука, 1978.— 351 с.
43. Королев М. А., Клемко Г. Н., Мишенин А. И. Информационные системы и структуры данных.— М.: Статистика, 1977.— 184 с.
44. Кохан Д., Якобс Г. Ю. Проектирование технологических процессов и переработка информации.— М.: Машиностроение, 1981.— 312 с.
45. Кохонен Т. Ассоциативная память.— М.: Мир, 1980.— 236 с.
46. Крайзмер Л. П. Память кибернетических систем:— М.: Сов. радио, 1971.
47. Ларионов А. М., Левин В. К. и др. Основные принципы построения и технико-экономические характеристики Единой системы ЭВМ// УСиМ.— 1973.— № 2.— С. 1—12.
48. Лебедев В. Н., Соколов А. П. Введение в систему программирования ОС ЕС.— М.: Статистика, 1978.— 144 с.
49. Мальцев А. И. Алгебраические системы.— М.: Наука, 1970.
50. Мальцев А. И. Алгоритмы и рекурсивные функции.—М.: Наука, 1965.
51. Мартин Дж. Организация баз данных в вычислительных системах.— М.: Мир, 1978.
52. Мидоу Ч. Анализ информационно-поисковых систем.— М.: Мир, 1970.— 367 с.

53. Микроэлектронные цифроаналоговые и аналоговые цифровые вычислительные преобразователи информации./Под ред. Смолова В. Б.— Л: Энергия, 1976.— 336 с.
54. Михалевич В. С. Последовательные алгоритмы оптимизации и их применение//Кибернетика.— 1965.— № 1.
55. Михалевич В. С., Кукса А. И. Методы последовательной оптимизации.— М.: Наука, 1983.— 207 с.
56. Михалевич В. С., Шор Н. З., Галустова Л. А. и др. Вычислительные методы выбора оптимальных проектных решений.— Киев: Наукова думка, 1977.— 178 с.
57. Многопроцессорные вычислительные системы.— М.: Наука, 1975.— 143 с.
58. Мультиплексоры передачи данных./Под ред. Лапина В. С., Корчинского А. И.— М.: Энергия, 1980.
59. Мультипроцессорные системы и параллельные вычисления.— М.: Мир, 1976.— 384 с.
60. Невельсон М. Б., Хасьминский Р. З. Стохастическая аппроксимация и рекуррентное оценивание.— М.: Наука, 1972.
61. Номенклатурный справочник к заявке 5—13 на 1985 г. на ЭВМ и комплексы общего назначения, вычислительных машин специального назначения, аппаратуры передачи данных телемеханики и изделий АСВТ.— М.: Изд-во АН СССР, 1983.
62. Петренко А. И., Семенков О. И. Основы построения систем автоматизированного проектирования.— Киев: Вища школа, 1985.— 294 с.
63. Позин И. Л., Щербо В. К. Телеобработка данных в автоматизированных системах.— М.: Статистика, 1976.
64. Поспелов Д. А. Фантазия или наука. На пути к искусственному интеллекту.— М.: Наука, 1982.
65. Рабинович З. Л. Элементарные операции в вычислительных системах.— Техника, 1966.— 303 с.
66. Саати Т. Целочисленные методы оптимизации и связанные с ними экстремальные проблемы.— М.: Мир, 1973.— 304 с.
67. Селтон Дж. Динамические библиотечно-информационные системы.— М.: Мир, 1979.— 558 с.
68. Сергиенко И. В., Лебедева Т. Т., Рощин В. А. Приближенные методы решения дискретных задач оптимизации.— Киев: Наукова думка, 1980.— 273 с.
69. Система ILLIAC — IV//Труды ИИЭР.— 1972. — 60, № 4.— С. 36—62.
70. Скурихин В. И., Шифрин В. Б., Дубровский В. В. Математическое моделирование.— Киев: Техника, 1983.— 360 с.
71. Системы автоматизированного проектирования. Материалы конференции IFIP по системам автоматизированного проектирования/Под ред. Аллана Дж.— М.: Наука, 1985.— 376 с.
72. Справочник по цифровой вычислительной технике: Процессоры и память./Под ред. Малиновского Б. Н.— Киев: Наукова думка, 1979.
73. Супервизор ОС ЕС ЭВМ/Под ред. Райкова Л. Д.— М.: Статистика. 1975.— 86 с.
74. Сэлтон Г. Автоматическая обработка, хранение и поиск информации.— М.: Сов. радио, 1973.
75. Твердохлеб П. Е. Голографическая память и информационные машины//Автометрия,— 1980.— № 2.— С. 9—24.
76. Теория самовоспроизводящихся автоматов.— М.: Мир, 1971.— 382 с.
77. Тербер К. Дж. Архитектура высокопроизводительных вычислительных систем.— М.: Наука, 1985.— 272 с.
78. Тетененбаум Э. Многоуровневая организация ЭВМ.— М.: Мир, 1979.— 547 с.
79. Флорес А. Организация вычислительных машин.— М.: Мир, 1972.— 430 с.

80. Хассон С. Микропрограммное управление.— М.: Мир.— Вып. 1. 1973.— 240 с.; Вып. 2. 1974.— 477 с.
81. Хигмэн Б. Сравнительное изучение языков программирования.— М.: Мир, 1974.— 204 с.
82. Хилбурн Д., Джулич П. Микро-ЭВМ и микропроцессоры.— М.: Мир, 1979.— 464 с.
83. Чачко А. Искусственный разум.— М.: Молодая гвардия, 1978.
84. Шелихов А. А., Селиванов Ю. П. Вычислительные машины.— М.: Энергия, 1978.
85. Шор Н. З. Методы минимизации недифференцируемых функций и их приложения.— Киев: Наукова думка, 1979.— 200 с.
86. Эндрю А. Искусственный интеллект.— М.: Мир, 1985.— 264 с.
87. Ющенко Е. Л. Адресное программирование.— Киев: Техиздат, 1963.
88. Якубайтис Э. А. Вычислительные системы и сети.— Рига: Знание, 1977.— 47 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ