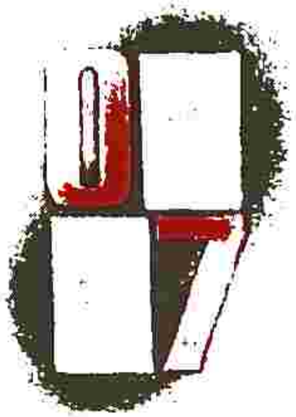


Цифровая вычислительная техника и программирование

ЦИФРОВАЯ ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРОГРАММИРОВАНИЕ

СБОРНИК СТАТЕЙ

под редакцией

А. И. КИТОВА

Выпуск 7

ИЗДАТЕЛЬСТВО «СОВЕТСКОЕ РАДИО»

МОСКВА — 1972

УДК 681.3

«Цифровая вычислительная техника и программирование». Сб. статей под ред. А. И. Китова, вып. 7. Изд-во «Советское радио», 1972. 192 стр., т. 12500 экз., ц. 1 руб.

Сборник посвящен теоретическим и практическим вопросам разработки цифровых вычислительных машин, их применения и эксплуатации, а также проблемам программирования для цифровых вычислительных машин.

В сборнике освещаются следующие аспекты цифровой вычислительной техники и программирования: вычислительные машины и их эксплуатация; теория вычислительных машин; алгоритмические языки; автоматизация программирования и отдельных работ при программировании; методы и приемы программирования; специальные и стандартные алгоритмы и программы обработки данных; вопросы структуры вычислительных комплексов и систем автоматической обработки данных.

Сборник рассчитан на широкие круги инженерно-технических и научных работников, занятых в области вычислительной техники и программирования, а также для студентов технических вузов и университетов, готовящихся к работе в указанной области.

2-3-11
48-71

ПРЕДИСЛОВИЕ

Статьи настоящего сборника могут быть объединены в следующие разделы:

— теория формальных языков,

— общие вопросы применения вычислительной техники;

— вопросы разработки информационных систем и алгоритмов поиска и выдачи данных;

— вопросы построения структуры управляющих цифровых машин;

— перспективные вопросы восприятия зрительной и звуковой информации с помощью ЭВМ.

Приведем краткий обзор статей по указанным разделам. К первому разделу относится статья Н. А. Криницкого, в которой намечается подход к построению теории алгоритмических языков на основе понятий математической лингвистики. Интересна попытка формализации понятия семантики.

Ко второму разделу относятся 4 статьи.

В статье А. И. Китова «Американские автоматизированные системы для медицины» дается анализ принципов построения двух крупных систем: библиографической системы и централизованной системы обработки данных о психических заболеваниях.

Статья К. В. Тараканова посвящена рассмотрению общего подхода к построению математического обеспечения автоматизированных систем управления.

В статье В. В. Семакова рассмотрена методика оценки вероятности выдачи ошибочных данных вычислительным центром в связи с общим технологическим процессом обработки данных в ВЦ.

В статье В. А. Германа «О расчете времени исполнения машинных программ» приведена методика исследования алгоритмов на основе их представления в виде стохастических граф-моделей. Даются расчетные формулы, алгоритм расчетов, типовые фрагменты граф-моделей и метод расчленения больших граф-моделей, упрощающий их расчет.

Третий раздел является наиболее крупным и содержит 10 статей.

В статье В. А. Павельева и Д. А. Степанченко описываются проблемно-ориентированный язык и система генерирования программ для задач обработки данных, разработанные американской фирмой IBM.

В статье А. И. Китова и Е. К. Грачевой описывается информационный язык с грамматикой, используемой как для повышения точности описания и поиска документов, так и для выдачи на печать формализованных фраз естественного языка. Описываются практически реализованный алгоритм формирования формализованных фраз.

В статье Ф. Д. Кожурина и др. приводится один практический способ построения информационных массивов, удобный для решения технико-экономических задач, в которых требуется осуществлять обращение к записям по их признакам.

В статье М. Б. Горизонтовой описываются принципы организации табличных фактографических информационно-поисковых систем, использующих специализированную библиотеку стандартных программ.

Е. А. Матэр и Г. Л. Фельдман описывают метод объединения массивов различной длины, записанных на магнитных лентах. Приводится алгоритм объединения и дается его оценка по числу прогонов магнитной ленты.

В статье Т. М. Аскерова и Л. П. Фейзуллаева дается оригинальный метод организации информационных систем, используемых в автоматизированных системах управления промышленного назначения. Метод проверен практически и показал достаточно высокую эффективность.

Ю. Н. Вуль и В. Ф. Пинкин в статье «Метод простых чисел для кодирования поисковых образов объектов в информационно-поисковых системах» предлагают оригинальный метод представления наборов дескрипторов, характеризующих документы, в виде простых чисел.

В статье Д. Д. Арнаудова «Об одном способе организации поискового массива в библиографических ИПС» приводится многоуровневая ассоциативно-адресная схема построения информационно-поисковой системы.

Статья С. К. Керимова и А. А. Мехтиева содержит информацию о практически реализованном способе хранения и поиска данных о химических соединениях и представляет интерес для разработчиков фактографических ИПС.

В статье М. Г. Гаазе-Рапопорта и С. Н. Лукьянова описывается алгоритм и программа анализа графов, используемая для исследования документооборота при проектировании систем обработки данных.

К четвертому разделу относятся 5 статей.

Статья В. В. Липаева и К. К. Колина «О составе операций и статистике их использования в программах управляющих ЦВМ» содержит фактический материал и представляет интерес для разработчиков специализированных ЦВМ для автоматизированных систем управления.

Важный вопрос автоматизации программирования и выпуска технической документации на программу для управляющей ЦВМ рассматривается в статье группы авторов во главе с Л. А. Серебровским. Описывается действующая система ЯУЗА-1.

К области теоретических основ управляющих ЦВМ примыкает статья В. П. Исаева и др., в которой рассматриваются вопросы моделирования процессов взаимодействия оператора и ЦВМ.

В статье С. Ф. Яшкова исследуются методами теории массового обслуживания модели вычислительных систем, работающих в режиме разделения времени и даются аналитические выражения для оценки работы этих систем.

В статье Ю. П. Горохова и Г. А. Соколова «Об одной задаче определения режима профилактики» дается математическая постановка и алгоритмы решения задачи выбора времени проведения профилактических работ в информационных системах с непрерывным режимом работы. Приводятся примеры и указываются ограничения применимости данной методики.

Пятый раздел включает в себя 3 статьи.

Важному перспективному направлению развития способов опознавания речи с помощью ЦВМ посвящены две статьи: Г. Д. Фролова и Г. Н. Русецкого и Ю. П. Скокана.

Статья Г. П. Катыс, С. Е. Здор, В. Б. Широкова «Оптимальные структуры оптоэлектронных систем поиска и распознавания» посвящена перспективам развития информационных систем, работающих в оптическом диапазоне.

Отдельно идет статья Г. А. Миронова и М. А. Островидова, в которой дается теоретическое обоснование диагностического алгоритма, использующего информацию о факте наличия неисправности.

Редакция будет признательна за все замечания и пожелания.

УДК 681.3—003

О НЕКОТОРЫХ ФОРМАЛЬНЫХ ЯЗЫКАХ

Н. А. КРИНИЦКИЙ

1. ЯЗЫКИ, ПОРОЖДАЕМЫЕ ДЕДУКТИВНЫМИ ГРАММАТИКАМИ

Условимся язык, о котором идет речь, называть *языком-объектом* в отличие от так называемого *метаязыка*, применяемого для описания языка-объекта.

В ряде случаев приходится пользоваться многими метаязыками, а также языками, которые являются метаязыками по отношению к метаязыкам, т. е. метаметаязыками и т. д. Исходным языком в наших описаниях будет один из естественных языков, а именно русский. Говоря о языке, необходимо четко различать метаязыки и язык-объект, без чего могут возникать неясности.

Пусть A и B — непересекающиеся алфавиты и σ — символ из алфавита B. Пусть, кроме того, Ω — конечное множество операций $\omega_1, \omega_2, \ldots, \omega_N$, для каждой из которых исходными данными и результатами являются слова в алфавите $A \cup B$. Предположим, что для некоторых операций однобуквенное слово σ является возможным исходным данным.

Совокупность

$$G = (A, B, \sigma, \Omega)$$

будем называть *дедуктивной порождающей грамматикой* [2].

Введем понятие выводимости следующим образом.

Если x и y являются словами в алфавите $A \cup B$, и если

$$y = \omega_i(x),$$

то будем писать

$$x \underset{u}{\Rightarrow} y$$

и говорить, что из x *непосредственно выводимо* y.

Будем говорить, что из x *выводимо* y и писать $x \Rightarrow y$ в одном из двух случаев (и никогда более):

а) если $x \underset{u}{\Rightarrow} y$;

б) если $x \underset{u}{\Rightarrow} z$; $z \underset{u}{\Rightarrow} y$.

Языком, порождаемым дедуктивной грамматикой G, будем называть множество L слов в алфавите A, выводимых из слова σ, т. е.

$$L = \{x \mid \sigma \Rightarrow x;\ x \text{ — слово в } A\}.$$

В нашем изложении L является языком-объектом, A — его алфавитом, любое x $(x \in L)$ — словом или (как иногда говорят) предложением языка L. Алфавит B является перечнем символов, служащих для обозначения грамматических объектов. Буква σ является метасимволом,

обозначающим одну из букв алфавита B. Последнее означает, что в конкретном случае роль буквы σ может выполнять другой символ. Говоря о грамматике вообще, мы такой символ (не одинаковый в разных возможных грамматиках) обозначили греческой буквой σ. В некоторых случаях в качестве букв алфавита B берут русские фразы, выделенные специальными кавычками или скобками. Например, среди этих букв могут быть <имя существительное>, <глагол> и т. п. При этом символ σ может иметь вид <предложение>.

Заметим, что приведенное здесь определение языка отличается от общеизвестного интуитивного представления о языке, во-первых, тем, что слова (предложения) языка не связываются с каким бы то ни было смыслом. Дедуктивная порождающая грамматика представляет собой, по существу, синтаксис порождаемого языка (морфологию естественного языка можно считать составной частью его синтаксиса, понятие фонетики в нашем определении не находит себе аналога, так как язык L является «письменным», не связанным со «звуковым» языком). Семантика языка должна быть задана особо.

Во-вторых, вышеприведенное определение языка отличается от привычных для нас описаний языков тем, что в нем любое слово (предложение) выводится из однобуквенного слова σ. Обычно же, образуя слова и предложения языков, идут от некоторого набора исходных элементарных слов (частиц), из которых с помощью грамматических операций конструируются все более сложные слова. Начиная с некоторого уровня получаются конструкции, являющиеся словами и предложениями языка, т. е. известные нам из практического опыта грамматики обычно являются не дедуктивными, а индуктивными, ведущими от элементарных частиц к конструкциям языка.

Рассмотрим весьма важный частный случай дедуктивной грамматики. *Марковской подстановкой* в алфавите $A \cup B$ называется операция, обозначаемая $P \longrightarrow Q$, где P и Q — слова в $A \cup B$, определенная для любых слов P, Q и заключающаяся в том, что в преобразуемом слове находят первое (если считать от начала) вхождение слова P и заменяют словом Q; то, что получится, является результатом выполнения операции. В тех случаях, когда P не входит в преобразуемое слово, операцию будем считать неприменимой к этому слову. Обозначим символом ( ) слово, которое получается при отбрасывании последних букв в слове (должно быть не больше длины слова).

Совокупность марковских подстановок

$$P \longrightarrow a,\ (P_1) \longrightarrow a_1,\ \ldots,\ (P)_{n-1} \longrightarrow a_{n-1},$$
$$a_0 \longrightarrow P,\ a_1 \longrightarrow (P_1),\ \ldots,\ a_{n-1} \longrightarrow (P)_{n-1},\ P \longrightarrow Q$$

в том случае, когда известно, что а) a_i — буквы в алфавите B, б) эти буквы не входят ни в какие другие подстановки и не встречаются в преобразуемых словах и словах P и Q, называется *подстановкой* и обозначается $P \longrightarrow Q$. С помощью подстановки (путем соответствующего применения образующих ее марковских подстановок) можно заменять не только первые, но и вторые, и третьи и т. д. вхождения слова P в преобразуемое слово (и любые их комбинации) словом Q. Подстановка не является операцией, так как для исходных слов, содержащих несколько вхождений слова P, допускается произвол результата. Мы в дальнейшем, не забывая, что подстановка — это набор операций, позволим себе некоторую «неряшливость», заключающуюся в том, что, говоря о подстановках, будем называть их операциями.

Обычно рассматривают порождающие дедуктивные грамматики, в которых Ω представляет собой некоторый конечный набор подстановок. При этом, если все подстановки в качестве левых частей (слов P) имеют однобуквенные слова в B, то язык, порождаемый грамматикой, называется *контекстно-свободным*.

II. ЯЗЫКИ, ПОРОЖДАЕМЫЕ ИНДУКТИВНЫМИ ГРАММАТИКАМИ

Индуктивной порождающей грамматикой назовем совокупность

$$\Gamma = (A, B, \sigma, \Omega, Я),$$

где

1) A и B — непересекающиеся алфавиты (A — алфавит языка-объекта, B — алфавит, буквы которого служат для обозначения грамматических классов);

2) σ — буква в алфавите B;

3) Ω — конечный набор операций $\omega_1, \omega_2, \ldots, \omega_i, \ldots, \omega_n$, исходными данными для которых служат упорядоченные наборы слов в A, а результатами являются слова в A. При этом рангом операции ω_i называется число r_i слов, входящих в набор, допустимый для ω_i в качестве исходного данного (предполагается, что для каждой ω_i это число является фиксированным). В отношении Ω делается предположение, что среди входящих в него операций есть некоторое число операций нулевого ранга, тем не менее дающих (каждая) вполне определенный результат. Совокупность таких результатов будем называть базой языка-объекта и обозначать $Б$,

4) $Я$ — конечный набор формул вида

$$y = f_j(x_1, x_2, \ldots, x_{r_j}),$$

составленных из букв в алфавите B. При этом запятая употребляется только для разделения аргументов в правых частях формул, скобки — только для выделения последовательностей аргументов в правых частях формул, а буквы f_j — только как названия операций из Ω. Предполагается, что каждой операции ω_i из Ω в составе $Я$ соответствует не менее одной формулы и что нет ни одной формулы в $Я$, которая не обозначала бы некоторой операции из Ω. $Я$ называется языком синтаксиса.

Введенное нами понятие индуктивной порождающей грамматики можно расширить, допустив в качестве языка синтаксиса язык, отличающийся структурой от $Я$, пригодный для обозначения исходных данных, операций и их результатов.

Пусть c, x_1, x_2, ..., x_n и y — слова в A. Будем говорить, что из *элементов множества M непосредственно конструируется y*, если

1) $y = f$, где f — имя операции нулевого ранга, а c — ее значение, и при этом M сводится к единственному элементу c, или если

2) $y = f_i(x_1, x_2, \ldots, x_n)$, где f — имя одной из операций и при этом M содержит в своем составе совокупность слов $x_1, x_2, \ldots, x_n$.

В обоих случаях будем писать $M :\Rightarrow y$, считая, что y — имя некоторого слова.

Теперь введем понятие *конструируемости*. Будем говорить, что *из элементов множества N* конструируется y, и писать $N \Rightarrow y$, если имеет место одно из двух:

1) $N :\Rightarrow y$;

2) $M \Rightarrow y$, причем коль скоро $x \in M$, то либо $x \in N$, либо $N :\Rightarrow x$.

Теперь язык-объект, порождаемый индуктивной грамматикой Γ, можно определить как множество тех слов σ в A, которые конструируются из элементов подмножеств базы языка-объекта, т. е.

$$L = \{\sigma \mid Б \Rightarrow \sigma,\ N\}.$$

Такое определение языка автор ранее использовал в [3]. Металингвистический символ σ в наших рассуждениях имеет смысл слова «предложение».

Иногда вместо языка $Я$ выбирают сходный с ним язык $Я^*$, который можно получить из $Я$ с помощью следующих трех правил.

1) если f — имя операции нулевого ранга, а c — ее значение, то всюду вместо f пишут c;

2) если в $Я$ содержится формула

$$y=f(x_1, x_2, \ldots, x_n),$$

отличающаяся от всех других формул, содержащихся в $Я$, левой частью, то пишут

$$y::=f(x_1, x_2, \ldots, x_n).$$

3) если в $Я$ содержится несколько формул с одинаковыми левыми частями

$$y=f_1(x'_1, x'_2, \ldots, x'_{r_1}),$$
$$y=f_2(x''_1, x''_2, \ldots, x''_{r_2}),$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$y=f_N(x_1^{(N)}, x_2^{(N)}, \ldots, x_{r_N}^{(N)}),$$

то вместо них пишут

$$y::=f_1(x'_1, x'_2, \ldots x'_{r_1})|$$
$$f_2(x''_1, x''_2, \ldots x''_{r_2})|\ldots|f_N(x_1^{(N)}, x_2^{(N)}, \ldots x_{r_N}^{(N)}).$$

Символ ::= читается как „по определению есть", а символ „|" как „или".

Получившиеся формулы будем называть *универсальными метаформулами*. Их частным случаем являются *формулы Бекуса*, использованные в грамматике алгоритмического языка АЛГОЛ-60. В формулах Бекуса используются только *) операции соединения $\omega_1, \omega_2, \ldots, \omega_N$, из которых первая — тождественная, а ω_i $(2 \leqslant i \leqslant N)$ является операцией соединения в одно слово заданной последовательности, образованной из i слов. В описании языка АЛГОЛ-60 используются операции соединения, для которых $N=5$. Предположим, что указанным операциям соединения соответствуют в $Я^*$ записи вида $S_1(a_1)$, $S_2(a_1, a_2)$, ..., $S_5(a_1, a_2, \ldots, a_5)$, где $a_1, a_2, \ldots, a_5$ — слова в A. То обстоятельство, что результаты a_1, $a_1\ a_2$, ..., $a_1\ a_2\ \ldots\ a_N$ выполнения операций соединения взаимно однозначно соответствуют вышеприведенным записям (правым частям формул) и это соответствие «видимо глазом», позволяет в формулах Бекуса опускать символы S_i, круглые скобки и запятые. Это не только сокращает записи, но и позволяет без изменения алфавита метаязыка (в котором теперь указанные символы отсутствуют) включить круглые скобки и запятые в алфавит языка-объекта. Например, среди формул Бекуса, использованных в грамматике АЛГОЛа, имеется формула (см. [1] § 4.6.1)

<элемент списка цикла> ::= <арифметическое выражение> | <арифметическое выражение> **step** <арифметическое выражение> **until** <логическое выражение> | <арифметическое выражение> **while** <логическое выражение>

При явном обозначении операций соединения эта формула имела бы вид:

<элемент списка цикла> ::= S_1 (<арифметическое выражение>) | S_3 (<арифметическое выражение>, **step**, <арифметическое выражение>, **until**, <арифметическое выражение>) | S_3 (<арифметическое выражение >, **while**, <логическое выражение>)

Возможность использования в языке-объекте скобок и запятой демонстрируют нам формулы Бекуса (см. [1] § 3.3.1 и 5.2.1):

<первичное выражение> ::= <число без знака> | <переменная> | <указатель функции> | (<арифметическое выражение>)

<список граничных пар> ::= <граничная пара> | <список граничных пар>, <граничная пара>

В первой из приведенных формул фигурируют круглые скобки, а во второй — запятая.

Заметим, что любая S_i $(i>2)$ может быть выражена через S_2. Например,

$$S_3(a_1, a_2, a_3)=S_2(S_2(a_1, a_2), a_3)\ =S_2(a_1, S_2(a_2, a_3)).$$

Это обстоятельство позволяет так построить грамматику языка-объекта, допускающего описание в виде формул Бекуса, чтобы в ее синтаксическом языке использовались лишь операции $S_1(a_1)$ и $S_2(a_1, a_2)$.

Попутно заметим, что формулы Бекуса вовсе не представляют собой универсального аппарата для описания языков. Например, грамматический класс <бидентификатор>, который представляет собой некоторый результат приписывания к <идентификатору> того же самого <идентификатора>, не может быть описан с помощью формул Бекуса, тогда как описание его с помощью универсальной метаформулы не представляет труда, например, в следующем виде: <бидентификатор> ::= D (<идентификатор>). Здесь $D(a)$ означает слово aa.

*) В [1] § 26.1 неожиданно появляется формула <чистая строка> ::= <любая последовательность основных символов, не содержащая символа ' или символа '> | <пусто>, которая противоречит общему стилю всей работы [1], так как содержит в правой части букву <любая последовательность основных символов, не содержащая символа ' или символа '> алфавита B, которая не является левой частью хотя бы одной из формул Бекуса и потому не определена.

III. СВЯЗЬ МЕЖДУ ИНДУКТИВНЫМИ И ДЕДУКТИВНЫМИ ГРАММАТИКАМИ

Докажем следующую теорему.

Теорема 1. *Для каждого языка L, порождаемого индуктивной грамматикой, существует порождающая его дедуктивная грамматика.*

Доказательство. Пусть язык L порождается индуктивной грамматикой $\Gamma=(A, B, \sigma, \Omega, Я)$. Построим дедуктивную грамматику $G=(A, B, \sigma, \Omega^*)$, выбирая в качестве Ω^* множество операций, определяемое следующим образом.

Если в $Я$ содержится формула

$$y=f(x_1, x_2, \ldots, x_n),$$

то в Ω включим две операции:

1) подстановку

$$y \longrightarrow f(x_1, x_2, \ldots, x_n)$$

и обобщенную подстановку

$$«f(a_1, a_2, \ldots, a_n)» \longrightarrow f(a_1, a_2, \ldots, a_n),$$

где $«f(a_1, a_2, \ldots, a_n)»$ означает запись функции f, в которую вместо аргументов подставлены $a_1, a_2, \ldots, a_n$ — слова в A, а $f(a_1, a_2, \ldots, a_n)$ означает результат выполнения операции, именем которой является f, при исходных данных $a_1, a_2, \ldots, a_n$. Например, если даны слово $5\cos\frac{\pi}{6}$ и обобщенная подстановка

$$«\cos a» \longrightarrow \cos a,$$

то результат применения последней будет словом $5\frac{\sqrt{3}}{2}$.

Легко показать, что каждое слово x в алфавите A, выводимое из σ, при грамматике G является словом σ, которое конструируется из элементов подмножества базы языка-объекта при грамматике Γ. И, наоборот, каждое слово x в алфавите A, которое при грамматике Γ является словом σ и конструируется из элементов подмножества базы язы-

ка-объекта, выводимо из σ в грамматике G. Действительно, рассмотрим цепочку, каждым шагом в которой является непосредственное конструирование. Такая цепочка будет начинаться с некоторых формул, в правых частях которых стоят имена операций нулевого ранга, и кончаться формулой, имеющей в левой части символ σ. Соответствующую этой цепочке цепочку, каждый шаг которой является непосредственным выводом, получим, двигаясь по первой цепочке от конца к началу и выполняя соответствующую ей подстановку столько раз, какова разность чисел появления левой части формулы до этого в правых и левых частях формул. Когда этот процесс окончен, каждую обобщенную подстановку выполним в обратном порядке такое же число раз. Таким образом, мы доказали, что каждое предложение языка L, конструируемое с помощью грамматики Γ, выводимо из σ с помощью грамматики G.

Теперь докажем обратное, предполагая, что задана цепочка подстановок и обобщенных подстановок, входящих в Ω^*, каждый шаг в которой соответствует непосредственному выводу. Нужную нам последовательность формул, ведущую от элементов подмножества базы к слову σ, мы получим, двигаясь по заданной цепочке в обратном направлении. При этом пропускаем обобщенные подстановки и записываем формулы, если такие формулы еще не были записаны, отвечающие подстановкам.

Итак, теорема доказана. Отметим интересный частный случай доказанной теоремы.

Теорема 2. *Если в индуктивной грамматике*

$$\Gamma = (A, B, \sigma, \Omega, Я)$$

множество операций Ω содержит кроме операций нулевого ранга только операции объединения, то для языка L, порождаемого грамматикой Γ, существует порождающая его дедуктивная контекстно-свободная грамматика

Очевидно, при доказательстве достаточно рассмотреть случай, когда в Ω могут присутствовать кроме операций нулевого ранга только $S_1(a_1)$ и $S_2(a_1, a_2)$. Доказательство этой теоремы проводится так же, как и доказательство теоремы 1. Затем во всех формулах переходят от обозначений $S_1(a_1)$ и $S_2(a_1, a_2)$ соответственно к обозначениям a_1 и a_1, a_2 (не содержащим букв S_1 и S_2, круглых скобок и запятых). При этом оказывается, что обобщенные подстановки превращаются в тождественные преобразования и потому могут быть отброшены. Сохраняются только обобщенные подстановки, связанные с операциями нулевого ранга, имеющие вид $f \longrightarrow c$. Операции, входящие в Ω^*, оказываются подстановками, содержащими в левых частях только однобуквенные слова в B. Таким образом, грамматика G оказывается контекстно-свободной.

Из доказательства теоремы 2 вытекает простой способ получения дедуктивных грамматик для языков, заданных с помощью формул Бекуса. Сущность его заключается в том, что каждую формулу Бекуса, имеющую в правой части несколько членов, разделенных символами «|», расщепляют на несколько формул, правые части которых являются одночленными. После этого к Ω^* относят операции, отвечающие подстановкам, получающимся путем замены в формулах Бекуса значков «::=» символами «$\longrightarrow$».

Например, формула Бекуса

<целое без знака> ::= <цифра>|<целое без знака> <цифра>

порождает две операции, отвечающие подстановкам:

<целое без знака>$\longrightarrow$<цифра>

<целое без знака>$\longrightarrow$<целое без знака> <цифра>

Легко видеть, что справедлива также следующая теорема.

Теорема 3. *Если язык L порождается контекстно-свободной дедуктивной грамматикой G, то существует порождающая его индуктивная грамматика Γ, набор операций Ω которой кроме операций нулевого ранга содержит только операции соединения слов $\widetilde{\omega}_1, \widetilde{\omega}_2, \ldots, \widetilde{\omega}_N$.*

Доказать эту теорему предоставим возможность читателю.

Справедлива и теорема, обратная теореме 1.

Теорема 4. *Для каждого языка L, порожденного дедуктивной грамматикой $G = (A, B, \sigma, \Omega)$ существует порождающая его индуктивная грамматика.*

Действительно, пусть Ω содержит n операций ω_i. Образуем алфавит B', не пересекающийся с алфавитами A и B, состоящий из букв z, f_0, $f_1, \ldots, f_{n+1}$, (,), <запятая> и σ'. Определим множество Ω', состоящее из $(n+2)$ операций:

$\omega'_0 = z$

$$\omega'_i(z) = \begin{cases} \omega_i(z), \text{ если } z \text{ не есть слово в } A, \\ \text{неопределенность, если } z \text{ — слово в } A; \end{cases}$$

$$\omega_{n+1}(z) = \begin{cases} z, \text{ если } z \text{ — слово в } A, \\ \text{неопределенность, если } z \text{ не есть слово в } A; \end{cases}$$

Будем считать, что $Я'$ — множество формул

$$z = f_0,\ z = f_i(z)\ \ (i = 1, 2, \ldots, n),\ \sigma' = f_{n+2}(z).$$

Положим, что $A' = A \cup B$. Нами получена индуктивная грамматика

$$\Gamma = (A', B', \sigma', \Omega', Я').$$

Покажем, что всякое слово, выводимое в G, конструируемо в Γ. Любой вывод в G является последовательностью шагов

$$z,\ \omega_{i_1}(z) = \sigma^1, \ldots,\ \omega_{i_{k-1}}(\sigma^{k-2}) = \sigma^{k-1},\ \omega_{i_k}(\sigma^{k-1}) = \sigma^k,$$

причем $\sigma, \sigma^1, \ldots, \sigma^{k-1}$ не являются словами в A, а σ^k — слово в A. Этой цепочке шагов соответствует последовательность формул, определяющих процесс конструирования

$$z = f_{i_1},\ z = f_{i_2}(z),\ z = f_{i_3}(z), \ldots,\ z = f_{i_k}(z),\ \sigma' = f_{n+1}(z),$$

приводящая к тому же слову. Очевидно и обратное. Всякая цепочка формул, определяющая конструирование, может иметь только вид, описанный выше, и ей соответствует единственная последовательность шагов вывода, описанная еще выше. Таким образом, теорема доказана. Но тогда справедлива следующая теорема.

Теорема 5. *Классы языков, порождаемых дедуктивными грамматиками и порождаемых индуктивными грамматиками, совпадают.*

Из теоремы 5 вытекает, что любой язык, порождаемый индуктивной или дедуктивной грамматикой, можно описать с помощью универсальных метаформул, описанных в разд. II.

IV. ФОРМАЛЬНАЯ СЕМАНТИКА ФОРМАЛЬНОГО ЯЗЫКА

Пусть $L = (x)$ — некоторый язык. Нам потребуется понятие подъязыка. При этом подъязыком мы будем называть некоторое подмножество языка. Поскольку, вообще говоря, язык не является конечным множеством слов, вводя понятие подъязыка, мы должны соблюдать определенную осторожность. Предположим, что $L' = (s)$ — некоторый другой формальный язык. Если существует алгоритм $\mathbb{W}(x, s)$, применимый к некоторым парам x, s, то множество тех x, каждое из которых входит хотя бы в одну такую пару, называется подъязыком языка L. В частности, область применимости любого алгоритма есть подъязык.

Попытаемся определить понятие семантики для языка L таким образом, чтобы оно соответствовало нашему интуитивному понятию семантики (осмысленности).

Итак, пусть Λ есть подъязык языка L. Мы будем говорить, что на Λ задана формальная семантика в любом из следующих пяти случаев:

1. Если существует язык L^*, на подъязыке Λ^* которого задана формальная семантика, и если известен алгоритм V, такой, что $V(x)=y$, где $x\in\Lambda$, $y\in\Lambda^*$.

В этом случае будем говорить, что семантика задана методом перевода (трансляции).

2. Если существует множество элементов

$$B=\{b\}$$

и если существует алгоритм U, такой, что $U(x)=b$, где $x\in\Lambda$, $b\in B$.

В этом случае слово x называется *именем* элемента b, а элемент b — *денотатом* имени x.

3. Если существует язык $L'=\{z_i\}_{i=1}^{\rho}$, где $\rho=\text{const}$ или $\rho=+\infty$, и алгоритм $W(\alpha, \beta)$, такой, что $W(x_i, z_{i,j})=y_{i,j}$, причем $x_i\in\Lambda$, $z_{i,j}\in L'$; $y_{i,j}\in\Lambda^*$, где Λ^* — подъязык языка L.

В этом случае говорят, что семантика на Λ задана внутренним образом по отношению к L.

4. Если существует язык $L''=\{z\}$ и алгоритм $W(\alpha, \beta)$, такой, что $W(x_i, z_{i,j})=y_{i,j}$, где $x_i\in\Lambda$, $z_{i,j}\in L''$, $y_{i,j}\in L''$.

В этом случае каждому x_i соответствует (в смысловом отношении) сложный объект, представляющий собой множество пар вида $(z_{i,j}, y_{i,j})$, который, в частности, может быть интерпретирован как некоторое отображение множества $Z_i=\{z_{i,j}\}_{j=1}^{\rho}$ на множество $Y_i=\{y_{i,j}\}_{j=1}^{\rho}$.

5. Если Λ представляет собой сумму конечного числа подъязыков, на каждом из которых семантика задана одним из первых четырех способов.

В практике мы встречаемся со всеми перечисленными способами задания семантики. Например, первым способом мы пользуемся, создавая искусственные языки. Смысл слов искусственного языка может быть задан с помощью словаря и системы правил перевода, позволяющих перейти от предложений искусственного языка к предложениям одного из естественных языков (например, русского). Такова может быть семантика языка эсперанто и подобных ему.

Второй способ задания семантики имеет место для подъязыка, представляющего собой множество собственных имен (например, в русском языке в такой подъязык входят слова «Москва», «Луна», «Эльбрус», «Пушкин» и др.).

При третьем способе задания семантики устанавливаются связи между некоторыми словами и классами других слов. Например, в русском языке к этому типу относится связь между словом «дом» и множеством слов «изба», «хата», «особняк», «небоскреб», и др., или между словом «дом» и множеством слов «подвал», «этаж», «чердак», «лестничная клетка» и т. д., или, наконец, связь между тем же словом «дом» и словом (фразой) «сооружение, состоящее из фундамента, крыши, подвала, этажей и лестничной клетки» (в последнем случае множество свелось к одному элементу).

Последние два примера позволяют считать, что на одном и том же подъязыке может быть задано несколько семантик. Отметим, что при третьем способе задания семантики язык L' играет вспомогательную роль, являясь как бы языком номеров, при помощи которых мы можем перечислять элементы множества слов, соответствующего слову из подъязыка Λ.

Четвертый способ задания семантики наряду с остальными способами мы видим в алгоритмических семантики, таких, как язык логических схем, АЛГОЛ-60 и др. Для алгоритмических языков он является основным. При этом способе каждому слову ставится в соответствие объект (конструктивное отображение), являющийся, вообще говоря, бесконечным множеством элементов.

Приведенные нами пояснения не претендуют на строгость и точность.

Для всех четырех основных способов задания семантики характерно то, что каждый из них ставит в соответствие слову подъязыка Λ некоторый объект с помощью *алгоритма;* это соответствие и является *смыслом* или *частью смысла* слова. Возможно, что существуют и другие способы ставить в соответствие слову те или иные объекты с помощью алгоритма. Если это так, то понятие формальной семантики придется расширить.

Следующая теорема демонстрирует «семантическую мощность» контекстно свободных языков.

Теорема 6. *Пусть L^* — язык, порождаемый индуктивной грамматикой $\Gamma=(A^*, B^*, \sigma^*, \Omega^*, Я^*)$, на котором (или на подъязыке которого) задана формальная семантика. Существует контекстно-свободный язык L и транслирующий алгоритм V, переносящий на L семантику языка L^*.*

Доказательство. Имена операций, входящие в формулы, без ограничения общности можно считать буквами. Условимся формулы из $Я^*$ вида $y=f(x_1, x_2, \ldots, x_n)$ писать без скобок, то есть в виде $y=fx_1x_2\ldots x_n$. Если в таких записях заменить знаки равенства знаками подстановок, то получим некоторое множество операций, которое обозначим Ω. Отнесем к A все буквы, являющиеся именами формул, а к B — все буквы, либо фигурирующие в левых частях формул, либо являющиеся аргументами в их правых частях. При этом σ^* войдет в B. Дедуктивная грамматика $G=(A, B, \sigma^*, \Omega)$ порождает искомый контекстно-свободный язык. Построить транслирующий алгоритм, который в словах языка L находит выполнимые формулы, и, выполняя их, получает слова языка L^*, не представляет труда.

В заключение заметим, что можно говорить и о *формальной речи*, если считать, что она задана в тех случаях, когда известны алгоритмы, ведущие от каких-то объектов к словам языка.

ЛИТЕРАТУРА

1. П/р Ершова А. П., Лаврова С. С. и Шура-Бура М. Р. Алгоритмический язык АЛГОЛ-60. Пересмотренное сообщение. Изд-во «Мир», 1965.
2. Гинзбург С. Математическая теория контекстно-свободных языков. Изд-во «Мир», 1970.
3. Кронецкий Н. А. Равносильные преобразования алгоритмов и программирование. Изд-во «Советское радио», 1970.

УДК 681.322

АМЕРИКАНСКИЕ АВТОМАТИЗИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ ДЛЯ МЕДИЦИНЫ

А. И. КИТОВ

В настоящей статье рассматриваются два вида автоматизированных информационных систем медицинского назначения:

1) система поиска медицинской литературы МЕДЛАРС (в двух вариантах: действующем — МЕДЛАРС-I и разрабатываемом — МЕДЛАРС-II);

2) централизованная система сбора и обработки информации по психиатрии для нескольких штатов

Анализируются принципы построения этих систем, их основные свойства и направления развития.

СИСТЕМА ПОИСКА МЕДИЦИНСКОЙ ЛИТЕРАТУРЫ МЕДЛАРС

Система МЕДЛАРС разработана и используется в Национальной медицинской библиотеке США (NLM), крупнейшем исследовательском и производственном центре в области обработки и распространения медицинской литературы в США и других странах.

Система МЕДЛАРС была создана в NLM в 1964 г. и в настоящее время широко используется для подготовки универсальных библиографических указателей (ИНДЕКС МЕДИКУС), ряда специализированных указателей, выполнения заказов по индивидуальному подбору (поиску) литературы по тематическим запросам.

Система имеет 3 филиала в США (Бостон, Нью-Йорк, Лос-Анжелес) и 5 филиалов в других странах (Япония, Англия, Швеция, Франция, Западная Германия). Создается филиал при ВОЗ в Женеве.

Филиалы системы МЕДЛАРС используются только для выполнения заказов для индивидуального подбора литературы по тематическим запросам. Для этого они ежемесячно получают из NLM копии магнитных лент, содержащих сведения о новых поступлениях литературы за прошедший месяц.

В качестве платы за представляемую информацию зарубежные филиалы системы МЕДЛАРС должны выполнять работу по индексации и подготовке ко вводу в машину сведений о публикациях в области медицины в своей стране. Эти сведения на бланках стандартной формы ежемесячно высылаются филиалами в главный центр системы. Ежегодно каждый филиал посылает сведения в среднем о 10—12 тыс. медицинских публикаций. Некоторые из филиалов сами стали центрами для обслуживания других стран. Так филиал МЕДЛАРС в Англии обслуживает многие учреждения во Франции (хотя в Париже имеется филиал МЕДЛАРС), филиал в Швеции обслуживает Норвегию и Финляндию. В настоящее время в МЕДЛАРС имеется массив данных о 1250 тыс. статей, опубликованных в 2 300 журналах (NLM выбрала из 6 000 медицинских журналов, издающихся в мире, 2 300 наиболее важных журналов).

Средний срок выдачи ответов на запросы составляет две недели (первоначально намечалось выдавать ответы в течение трех дней). Основной продукцией системы МЕДЛАРС является упомянутый выше универсальный библиографический указатель медицинской литературы «ИНДЕКС МЕДИКУС», выходящий ежемесячно. Кроме месячных выпусков издаются годовые выпуски, объединяющие соответствующие месячные выпуски. Ежегодно в январских выпусках печатается словарь MeSH (тезаурус) медицинских терминов, используемых в МЕДЛАРС, с указанием вновь введенных и исключенных по сравнению со словарем, опубликованным в предыдущем году.

Кроме «ИНДЕКС МЕДИКУС» NLM выпускает специализированные библиографические указатели (например, по стоматологии, по медикаментам, по вопросам реанимации и т. д.)

Избирательное распределение информации по постоянным запросам выполняется для медицинских учреждений, для которых выпускаются специализированные указатели. Для индивидуальных заказчиков избирательное распределение осуществляется как разновидность поиска по запросам.

МЕДЛАРС включает в себя кроме описанной системы обработки медицинской литературы, накапливающей и обрабатывающей статьи из журналов, вторую систему, решающую те же задачи применительно к книгам, сборникам трудов, диссертациям и другим публикациям, не являющимся журнальными статьями. Для второго вида литературы выпускается библиографический указатель, называемый «Текущий каталог» (Cuttent Catalog), а также библиографические карточки. В «Текущий каталог» включаются и журналы (целиком, а не по статьям).

Основной причиной раздельного ведения массивов информации для статей и для книг (сборников) явилась различная глубина их индексации и различные виды использования библиографической информации об этих двух видах публикаций.

Основным словарем, используемым при индексации документов и запросов, является тезаурус (Medical Subject Headings) (MeSH), включающий в себя около 8 000 дескрипторов. Кроме MeSH имеется словарь узких (профессиональных) терминов и словарь географических названий. Эти словари используются индексаторами для более точного представления содержания статей в поисковых образах документов. Всего в системе МЕДЛАРС работают около 50 индексаторов, часть из них находится в NLM, а часть — в филиалах МЕДЛАРС в других странах. Норма работы индексатора — 5 статей в час, т. е. 40 статей за 8 часов рабочего дня. В обязанности индексатора входит простановка главных терминов и стандартных дескрипторов и пометок, запись выходных данных журналов, а также перепечатка поискового образа документа на бланк. Обучение индексаторов ведется на четырехмесячных курсах.

В настоящее время система МЕДЛАРС работает на ЭВМ Honeywell-800 и частично — на ЭВМ IBM-360/50.

Кроме МЕДЛАРС в NLM имеется централизованная система поиска научно-популярных кинофильмов, терминал которой находится в библиотеке учебных фильмов (г. Атланта, штат Джорджия). Запросы о фильмах в Атланту поступают письмами и по телефону, а оттуда с помощью терминала передаются в ЭВМ NLM. ЭВМ осуществляет поиск и выдает сведения о наименовании и местонахождении требуемых фильмов, о сроках возвращения фильмов заказчиками, кроме того, ЭВМ печатает списки должников, т. е. лиц и учреждений, не вернувших фильмы в установленный срок, а также расчетные ведомости за пользование фильмами.

За время эксплуатации системы МЕДЛАРС были выявлены ее недостатки и в настоящее время в NLM ведется в широком масштабе работа по созданию новых информационно-поисковых систем для медицины. В этих работах заняты как исследовательские отделы и ВЦ NLM, так и промышленные фирмы и крупные научные учреждения, выполняющие работы по контрактам с NLM. Общее научное руководство разработками в качестве представителя NLM ведет заведующий ВЦ NLM доктор Р. А. Симмонс (R. A. Simmons), специалист по программированию. Эксплуатацией системы МЕДЛАРС, включая взаимодействие с филиалами, руководит зам. директора NLM доктор Ж. Лейтер (J. Leiter).

Развитие системы МЕДЛАРС идет по пути создания нового варианта этой системы, получившего название МЕДЛАРС-II. Эта работа была начата в 1967 г. фирмой Computer Sciences Corporation, 8728 Colsville Road Siever Spring Moryland 20910 по контракту с NLM. Система МЕДЛАРС-II должна была быть введена в действие в июле 1969 г., но она еще находится в стадии разработки. Как утверждали руководители NLM, эта система находится в стадии отладки, однако состояние ра-

бот таково, что она может быть введена в эксплуатацию не раньше конца 1972 г.

Основные отличия МЕДЛАРС-II от МЕДЛАРС-I (просто МЕДЛАРС) заключаются в следующем. МЕДЛАРС-II предусматривает:

а) значительное увеличение объема тезауруса, расширение данных, описывающих каждый термин, а также развитие аппарата перекрестных ссылок и иерархических и ассоциативных отношений между терминами словаря;

б) введение большого количества специальных терминов, не являющихся дескрипторами, но используемых при поиске литературы по запросам. В словаре имеются пометки, запрещающие использовать некоторые термины в определенных сочетаниях;

в) значительное увеличение полноты и глубины описания документов за счет увеличения числа тематических терминов до 35, числа подзаголовков до 50 и включения в бланк поискового образа документа дополнительных сведений об авторе статьи, месте его работы, характере статьи,

г) переход на новую значительно более мощную вычислительную машину IBM-370/165, которая является дальнейшим развитием машин серии IBM-360

Принципиальным отличием МЕДЛАРС-II от МЕДЛАРС-I является применение автоматической дальнодействующей системы связи между ЭВМ и заказчиками для ввода запросов и выдачи ответов и между ЭВМ и индексаторами для ввода поисковых образов документов.

Для обработки и экспериментальной проверки этих принципов, алгоритмов, программ и технических средств в NLM в настоящее время ведутся работы по двум проектам ЛИСТХИЛЛ и ЛИСТАР. Проект ЛИСТХИЛЛ находится в стадии пробной эксплуатации. Суть его заключается в следующем. В Лос-Анжелесе имеется опытный вычислительный центр системы МЕДЛАРС-II, который связан телефонными каналами связи со многими абонентами. У абонентов имеются специальные пульты с пишущими машинками, с помощью которых можно связаться с ВЦ в Лос-Анжелесе, послать туда запрос на поиск нужной литературы и получить сразу же ответ, отпечатанный на пишущей машинке этого пульта. В опытном ВВЦ хранится ограниченный массив медицинской информации — данные о статьях за 5 лет из 100 наиболее распространенных американских медицинских журналов, всего 125 000 статей. Как утверждают руководители, NLM этот массив удовлетворяет подавляющее большинство запросов практических врачей США.

Второй проект — ЛИСТАР — выполняется по контракту с Линкольновской лабораторией Массачузетского технологического института в Бостоне. Руководителем является доктор А. Арменти (A. Armenti).

Серьезный недостаток МЕДЛАРС-I, который сохранится и в МЕДЛАРС-II, заключается в необходимости ручной работы с тезаурусом MeSH при индексировании. Индексатор может использовать только те термины, которые имеются в MeSH, и для этого он должен *проверять каждый* термин на наличие его в MeSH. Если данного термина в MeSH нет, то нужно найти в MeSH его синоним, если нет синонима, то — общий или частный термин. Для этого используются перекрестные ссылки и словарь древовидной структуры (дерево MeSH). Это требует сложной ручной поисковой работы с MeSH и деревом MeSH, а также с другими специальными словарями (например, словарем названий русских лекарств, русских названий болезней, образованных от имен ученых и т. д.). Этот процесс вполне может быть автоматизирован с помощью ЭВМ, но разработчики МЕДЛАРС-II не решили этой проблемы.

Индексация запросов осуществляется, в основном, теми же индексаторами, которые выполняют работу по индексации документов. Имеется подробное руководство по индексации, определяющее использование всех терминов. Получив запрос заказчика о требуемой литературе, индексатор должен осмыслить запрос и, если требуется, произвести уточнение запроса у заказчика либо по телефону, либо по почте.

Сложный (многоаспектный) запрос индексатор расчленяет на простые аспекты (фасеты). Для каждого фасета индексатор выбирает из словаря MeSH все подходящие термины, которые соединяет связками «ИЛИ». Различные фасеты, входящие в один запрос, соединяются связками «И». Кроме связок «ИЛИ» и «И» индексатор может использовать для построения поискового предписания связку «НЕТ» в тех случаях, когда требуется найти документы, в поисковых образах которых отсутствовали бы определенные дескрипторы (термины).

Если первая попытка поиска не дает удовлетворительных результатов, индексатор заменяет в поисковом предписании один или несколько дескрипторов на более широкие или более узкие или использует другие дескрипторы, связанные с темой запроса ассоциативными отношениями. Получаемые в результате поиска по различным запросам списки литературы выдаются заказчикам и одновременно с этим накапливаются в памяти ЭВМ вместе с запросами, образуя поисковую библиотеку, используемую при последующих запросах.

Эта система индексации документов и поиска по запросам, отработанная в МЕДЛАРС-I, сохраняется, в основном, и в МЕДЛАРС-II.

Существенным недостатком системы МЕДЛАРС-I является невысокая избирательность поиска документов. Часто в ответ на тематический запрос заказчика МЕДЛАРС-I выдает большое количество избыточных документов, иногда имеющих лишь отдаленное отношение к теме запроса.

Для устранения этого недостатка в МЕДЛАРС-II применяется более глубокая индексация документов, значительно расширяется словарь терминов и система их иерархических отношений, а также вводится большое количество стандартных признаков (дескрипторов) для описания формы и содержания документов.

Следует заметить, что отбор статей из журналов для включения их в «ИНДЕКС МЕДИКУС» или в поисковый массив ведется самими индексаторами, которые полностью отвечают за качество отбора статей.

МАССАЧУСЕТСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ (MIT) И ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ ГЛАВНОГО ГОСПИТАЛЯ В БОСТОНЕ

В Линкольновской лаборатории MIT под руководством проф. Арменти выполняется проект ЛИСТАР. Назначение проекта заключается в разработке методов, алгоритмов и программ для дистанционной индексации документов, вводимых в НПС системы МЕДЛАРС. Предполагается, что после реализации проекта индексаторы, находящиеся в различных городах США и других стран, смогут с помощью пультов управления с пишущими машинками, связанных телефонными каналами связи с вычислительным центром NLM, производить ввод поисковых образов документов в ЭВМ МЕДЛАРС, выбирать необходимые термины из словаря MeSH, хранящегося в памяти ЭВМ, а также производить поиск, формировать и корректировать информационные массивы.

Работа над проектом ЛИСТАР была начата в 1964 г. В настоящее время этот проект находится в стадии завершения. Ведется его экспериментальная проверка и доработка с использованием ЭВМ типа IBM-360

(модель 67), находящейся в Линкольновской лаборатории в Лексингтоне. Для связи абонентов с ВЦ используются пульты с клавиатурой (терминалы) типа IBM-2741, IBM-1050, IBM-2260, усовершенствованная система дистанционного отображения и телетайпы.

Интересными с точки зрения методов построения информационных систем являются принципы организации информационных массивов и язык взаимодействия между ЭВМ и пользователями, разработанные в системе ЛИСТАР. Имеется возможность любому пользователю формировать свои массивы, обладающие переменным числом записей переменной структуры.

Характеристики (описания) всех массивов хранятся в специальном массиве, называемом Главным массивом. В нем указывается состав и структура полей внутри записей, типы значений (буквенные, цифровые, смешанные, специальные). Для взаимодействия между ЭВМ и абонентами используется язык команд, составленных из слов английского языка. Состав команд может дополняться отдельными абонентами. В системе ЛИСТАР предусмотрена возможность инструктирования со стороны ЭВМ неопытного абонента о порядке работы с данной системой и использования ее команд. Для этого достаточно нажать специальную клавишу на пульте и напечатать код непонятной команды. После этого ЭВМ напечатает необходимые пояснения с примерами. Подобный метод машинного инструктирования широко применяется во многих американских автоматизированных системах.

Разработку принципиально новой информационно-поисковой системы универсального назначения предусматривает проект ИНТРЕКС. Основными чертами этой системы, которая в настоящее время находится в стадии опытной эксплуатации, являются использование естественного языка, не ограниченного заранее составленным тезаурусом, для индексации документов и запросов и систем наглядного отображения телевизионного типа для выдачи текстовой и графической информации абонентам.

В отделе информационных наук Рокландского государственного госпиталя в Нью-Йорке под руководством проф. Н. Кляйна, доктора Е. Ласка, доктора Г. Логемана разработана централизованная информационная система для обработки данных о психических больных (MSIS). Система MSIS предназначена для обслуживания семи штатов — Коннектикут, Майн, Массачусетс, Нью Гэмпшир, Нью-Йорк, Род Айленд, Вермонт, и позволяет следить за состоянием здоровья психических больных на всех стадиях психиатрического обслуживания и обобщать опыт этой работы. Для каждого вводимого в НПС сообщения НВЦ автоматически выдает рапорт абоненту, приславшему это сообщение. Данные в НПС представляют три вида событий, связанных с пациентами: перемещение (прием, перевод, выписка и др.), применяемое лечение (включая диагностику, анализы, выписку лекарств и т. д.), изменение состояния пациента.

После полного завершения системы типовая запись истории болезни будет содержать данные, заполняемые при приеме пациента (возраст, пол, адрес, жалобы, психическое состояние, сведения о предшествующем медикаментозном лечении, замечания по состоянию больного, данные общего медицинского и неврологического обследований, лабораторные анализы), а также информацию, поступающую после выписки больного. Первичная информация для ЭВМ записывается на стандартных бланках в виде позиционных отметок, воспринимаемых оптическим читающим устройством, данные с бланков могут также перфорироваться. Формы бланков построены по принципу постановки отметок в списке вопросов (анкет) с заранее указанным набором возможных ответов. Заполняющее лицо обязано только отметить соответствующие позиции в анкете (вопроснике). Заполнение формы может

производиться врачами, сестрами, воспитателями, учителями, социальными работниками и другими лицами, связанными с больными.

MSIS предусматривает учет данных как для больных, находящихся в больницах, так и для остальных психических больных. Данные в НВЦ могут посылаться по почте или передаваться по телефонным линиям связи через терминалы, находящиеся в каждом штате.

Первоначально введенная в ЭВМ запись болезни будет постоянно дополняться и следовать за больным на всех стадиях его лечения, а также при перемещении больного в другие медицинские учреждения или палаты. При поступлении больного в новое медицинское учреждение туда будет высылаться печатное сообщение о его истории болезни. Кроме того, НВЦ будет выдавать сводки о перемещении больных между разными медицинскими учреждениями, об использовании разных видов психиатрического обслуживания, и о потребностях в медицинском обслуживании для разных районов, или групп населения.

Для выработки принципов использования и построения системы привлекаются ведущие медицинские специалисты, которые образуют соответствующие комиссии: комиссию по формам исходных бланков, заполняемых медиками, комиссию по отчетности, связанной со статистическими формами отчетов, необходимых штатам. Все формы унифицированы для всех участвующих в MSIS штатов; однако в формах предусмотрены резервные поля для записи специфических данных, если они потребуются какому-либо штату. Информация в MSIS может использоваться не только для обеспечения процесса лечения данных больных, но и для планирования психиатрического обслуживания, изучения эпидемиологических вопросов, анализа распределения потоков пациентов и эффективности разных терапевтических средств и т. д. Возможные различия в данных, поступающих из разных мест, должны анализироваться с целью выяснения причин различий и их устранения. Данные о лечении больных в соединении с общими данными о медицинском обслуживании населения позволят уточнить картину заболеваний.

Комиссией по отчетности были рассмотрены и утверждены формы приемных бланков, бланков выписки и бланков описания состояния пациентов. На первом этапе не предусматривается ввод в базу данных сообщений с бланков рапортов обследования психического состояния. Эти данные должны с бланков (в виде отметок) вводиться в НВЦ и выдаваться обратно в описательной форме.

Разрабатывается единая шкала для оценки психического состояния, но пока для этой цели отсутствует специальная форма бланка; не градуированное описание психического состояния дается в описательной форме.

Одним из основных принципов MSIS является ответственность каждого участвующего в системе медицинского учреждения за точность его массивов информации. Система позволяет любую порцию данных, введенную в систему, исключить, скорректировать или дополнить. При этом в большинстве случаев используются те же самые формы, которые заполнялись для ввода исходной информации.

После каждой передачи данных отправителю возвращается сообщение с анализом ошибок (если они есть) и их исправлениями. Несмотря на то, что рабочие записи, относящиеся к пациентам, будут исправлены, в ЭВМ сохраняются следы о проведенных проверках и исправлениях, так что в любой нужный момент может быть найдена и выдана как правильная, так и неправильная информация, а также все проведенные исправления.

Накопление данных и их обработка проводятся в едином НВЦ в Рокленде, с которым все участвующие в системе медицинские учреждения связаны каналами связи. Каждое такое учреждение имеет

у себя пункт связи, называемый терминалом. Этот терминал имеет следующее оборудование:

— устройство чтения/печати с перфокарт IBM-2780, модель 1;
— устройство оптического чтения с помеченных страниц IBM-1232,
— устройство перфорации (перфоратор карточный) IBM-534, модель 3, соединенное с устройством 1232;
— автономный перфоратор IBM-029.

Все терминалы в семи штатах связаны с НВЦ в Рокленде с помощью устройства передачи данных типа Telephone Company 201-A-1.

Это оборудование дает возможность каждому терминалу воспринимать и декодировать с помощью оптического читающего устройства отметки с листов (бланков) с данными и передавать их в НВЦ, а также получать информацию из НВЦ с высокой скоростью по телефонным линиям. Обмен между НВЦ и терминалом начинается с того, что программа в ЭВМ, называемая «ТО-монитор» (телеобработка), выбирает данный терминал.

Среднее время выбора (ожидания) 20 минут. Передаваемые из терминала данные должны быть предварительно отперфорированы на перфокартах. После того, как все карты, введенные в приемный карман читающего устройства, будут прочитаны, терминал заканчивает передачу и в ответ из НВЦ получает соответствующий рапорт. В НВЦ поступившие сообщения сортируются по отправителям и по типу данных (например, приемные данные, данные изменения состояний, местоположения, служебные данные и т. д.)

Затем ЭВМ выполняет обработку каждого типа данных для каждого отправителя, включая проверку данных на их соответствие между собой и с предыдущими данными, а также на наличие других ошибок. Затем ЭВМ выдает отправителю рапорт анализа ошибок, в котором указывается, принимает ли НВЦ или нет эти данные и список ошибок, если они есть. Результаты анализа ошибок вместе с исходными присланными данными запоминаются в контрольном массиве, называемом файлом сообщений.

Для каждого участника MSIS (учреждения) в составе базы данных этого участника хранится необходимая специфическая информация (и правила), например номера и число отделений, фамилии персонала. Сведения, запрашиваемые медицинскими учреждениями, вырабатываются НВЦ в две стадии:

а) запрос приходит в виде специальной форматной карты и вызывает программу, называемую монитором рапортов, которая осуществляет анализ ошибок и производит запись сообщений.

б) вырабатывается фактический рапорт, содержащий запрашиваемые сведения, и об этом факте делается отметка в файле сообщений.

В НВЦ имеется два центральных процессора IBM-360/50 и IBM-360/44; кроме того имеется машина IBM-360/30. Последняя ЭВМ используется только для расчетов по заказам отдела психической гигиены штата Нью-Йорк. Дублирование ЭВМ в НВЦ обеспечивает надежность функционирования и возможность работы в реальном масштабе времени. Кроме того, в НВЦ имеются отдельные частности этого оборудования, которые могут использоваться в будущем для наращивания вычислительной системы. Обе основные ЭВМ в НВЦ работают с использованием многопрограммной операционной системы, называемой MFT (многопрограммная работа с фиксированным набором задач). При этом режиме работы МОЗУ делится на блоки переменного размера: некоторые из них закреплены для обработки передач данных, некоторые — за программами. Небольшая часть МОЗУ (≈50 К) выделена для высокоприоритетного использования и для форматирования рапортов, большая часть МОЗУ (≈126 К) используется для размещения обрабатываемой информации, поступающей от терминалов.

MFT обеспечивает выполнение определенного класса работ различного приоритета. В MSIS установлен следующий стандартный ряд классов работ:

Класс	Типы
1	Операции по восстановлению базы данных.
2	Ввод и обработка данных от терминалов
3	Запросы к базе данных в Роклендском госпитале.
4	Разработка новых программ запросов (заявлений) от участников.
5	Обработка для исследовательского центра в Рокленде
6	Выработка рапортов, вызванная монитором рапортов, для терминалов
7	Выработка рапортов для Роклендского госпиталя
8	Работа, связанная с новыми программами участников.

При разработке MSIS рассматривался возможный круг задач и выбирались в большинстве случаев не частные, а общие решения. Для конкретных задач рассматривались классы подобных задач с целью упростить будущее развитие подобных систем, в том числе сделать возможным наращивание системы в интересах использования ее отдельными участниками кооперации. Одним из общих подходов в построении системы явилось использование для взаимодействия людей с ЭВМ метода типовых вопросников, содержащих для каждого вопроса ряд заранее предусмотренных вариантов ответов. Наиболее удобной для врачей является текстовая форма ответов.

Интересна применяемая в MSIS методика ввода медицинской информации в ЭВМ непосредственно с бланков с помощью оптического читающего устройства, которое определяет местоположение и наличие соответствующей отметки на документе и кодирует эту информацию в форму, удобную для хранения в ЭВМ.

Для интерпретации данных, вводимых с бланков, в НВЦ разработана универсальная программа Scanfins, представляющая собой универсальную программу обработки табличных документов. Настройка подобных программ на работу с конкретным документом производится при помощи таблицы описания этого документа.

При разработке форм документов, предназначенных для чтения с помощью оптического читающего устройства, необходимо учитывать особенности работы этого устройства, связанные с его разрешающей способностью.

Содержание любого документа, независимо от того, введен он с помощью оптического читающего устройства или с помощью перфорации, подвергается в ЭВМ логическому контролю, заключающемуся в проверке либо соответствия отдельных полей документа существующим ограничениям (например, дата 30 февраля невозможна), либо соответствия между документом и предыдущими записями.

Программа контроля состоит из ряда модулей анализа ошибок, которые могут использоваться независимо для разных типов документов. Факты обнаружения ошибок, так же как факты обработки документов или факты выдачи рапортов, фиксируются в универсальном файле сообщений, в котором регистрируются все виды взаимодействия между центральной базой данных и абонентами MSIS.

Этот файл используется для анализа ошибок, выдачи рапортов об ошибках, для выдачи сообщений всем терминалам об итогах их дневной деятельности, а также для определения так называемой ревизионной подсистемой в случае обнаружения ошибок вида дальнейшей работы: нужно ли сразу исправлять соответствующие поля в базе данных или необходимо разбираться в появившихся ошибках специалистам и выяснять их причины.

Важной особенностью MSIS является возможность каждому участнику кооперации добавлять в систему свои собственные программы, а также создавать свои массивы данных, которые могут обрабатываться общими и специальными модулями программ MSIS. Для введения этих массивов используется язык DL/I, разработанный фирмой IBM.

Система DL/I предотвращает возможность взаимных помех и искажений данных, связанных с одновременным выполнением специальных программ, введенных в систему отдельными участниками, и основных программ, общих для всех участников системы. DL/I практически допускает неограниченное расширение объема и состава медицинских записей; количества и видов документов, вводимых в систему.

Везде, где возможно, в MSIS применяются универсальные формы входных и выходных документов, подходящие для всех участников кооперации.

В некоторых случаях НВЦ создает специальные модификации программ для отдельных штатов. Для того чтобы учесть эту специфику в массивах данных, соответствующих отдельным штатам, предусматриваются соответствующие данные, называемые «профиль участника», к которым относятся, например, фамилии администрации госпиталей, структура госпиталей и т. д. Все участники MSIS кроме использования описанной выше информационной системы накопления и обработки медицинских записей имеют возможность полностью использовать ЭВМ, имеющиеся в НВЦ, для своих собственных вычислений в режиме разделения времени с дистанционным обращением. Доступ к ЭВМ осуществляется с помощью устройств 2780 и телефонных линий, используемых как для передачи данных, так и для передачи программ. Это позволяет каждому участнику кооперации развивать свою часть общей системы и вести научные исследования, которые требуют применения больших вычислительных машин.

Необходимо отметить, что система MSIS предусматривает также статистическую обработку информации.

Анализ принципов построения централизованной автоматизированной системы MSIS для обработки данных по психиатрии позволяет сделать следующие выводы.

1. Использование мощной вычислительной машины (например, типа IBM-360, модель 50), работающей в режиме разделения времени, делает эффективной централизованную обработку медицинских данных для многих больниц, расположенных на значительном расстоянии от центра обработки (1000 км и более).

2. Централизованная обработка является не только более выгодной экономически, но и позволяет осуществить унификацию форм документации, методов обработки и обеспечить сопоставимость информации, получаемой из различных больниц, что очень важно с точки зрения статистического анализа.

3. Целесообразно построение централизованных систем обработки информации с раздельным ведением информационных массивов для различных участников кооперации и полной ответственностью этих участников за ввод и корректировку первичной информации. С этой целью в системе значительная роль отводится программам контроля вводимой информации и выдачи сообщений об ошибках, получению подтверждений изменений и полной регистрации всех актов взаимодействия между участниками и системой.

4. При разработке подобных систем должны быть предусмотрены наиболее удобные для людей формы ввода и вывода информации. В отношении ввода такой формой является простановка отметок в заранее подготовленных стандартных бланках, читаемых автоматическими устройствами. В отношении вывода наиболее удобной является печать информации в виде текста на естественном языке.

5. Весьма перспективным является взаимодействие между информационными системами и людьми в виде диалога, направляемого ЭВМ на основе заранее составленной (и наращиваемой) древовидной структуры вопросника с возможными вариантами ответов.

УДК 681.3-52

ОБЩИЕ ПРИНЦИПЫ И СТРУКТУРА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ

К. В. ТАРАКАНОВ

Каждая конкретная АСУ независимо от того, в интересах какого органа она создается, предназначена для сокращения сроков сбора систематизированной информации, необходимой для управления, сокращения сроков обработки информации, максимальной разгрузки административно-управленческого аппарата, уменьшения срока выполнения трудоемких и однообразных вычислительных операций, повышения точности расчетов, обеспечения возможности решения всех стоящих перед данным органом управления задач.

На первой стадии работ по автоматизации в каждом звене управления создавались автономные органы управления, которые решали определенный комплекс задач, отражающих характер функционирования того или иного звена управления. Основная стратегия при этом сводилась к разработке достаточного количества задач и решению их на автономных средствах вычислительной техники.

Анализ процессов создания математического обеспечения показывает, что такая организация работы приводит к дублированию, к созданию идентичных программ. Известно то обстоятельство, что разработка комплексов расчетных задач не является главным содержанием автоматизации.

Всякая организованная деятельность людей по управлению прежде всего связана с переработкой информации, которую и нужно автоматизировать. Единственным правильным решением является идея концентрации усилий при создании информационных систем. При решении автономных расчетных задач лишь облегчается вычислительная работа органов управления. Основная же задача состоит в том, чтобы глубоко проникнуть в область процессов управления, изменить характер этих процессов с помощью средств вычислительной и организационной техники и средств связи. Для этой цели создается математическое обеспечение.

Математическое обеспечение автоматизированных систем управления разделяется на общее и специальное. Под общим математическим обеспечением понимают ту его часть, которая является приложением к самой технике и меняется только в зависимости от изменения самой техники. Специальное математическое обеспечение более тесно связано с характером автоматизируемого процесса управления и больше подвержено изменениям, чем общее математическое обеспечение, — это часть математического обеспечения, которая формализует деятельность объекта управления. Общее же математическое обеспечение формализует, в сущности, структуру, характер работы всех технических средств автоматизированной системы управления.

Можно рассмотреть несколько принципов, которых нельзя не придерживаться при создании системы математического обеспечения. Для этого прежде всего необходимо сформулировать конечную цель функционирования всей системы математического обеспечения, вид, форму и со-

держание выходных документов, затем решить, какие информационные процессы и модели создавать, какие результаты отображать и какими правилами руководствоваться при принятии решения по отображенным результатам.

Создаваемое математическое обеспечение должно непрерывно согласовываться с принципами построения автоматизированной системы управления Специальное математическое обеспечение, как было отмечено, создается с целью изменения структуры управления. Но структура управления также должна быть приспособлена к системе математического обеспечения, т. е. должно быть обеспечено взаимное согласование создаваемого математического обеспечения и изменяющейся структуры управляемого объекта, иначе говоря, оба эти элемента должны находиться в некотором итерационном процессе.

При формулировании конечной цели создания специального математического обеспечения чрезвычайно важно сформулировать содержание выходных документов, установить их перечень, так как в зависимости от их формы и содержания решаются два вопроса: какой процесс необходимо создавать и какова должна быть форма представления всей входной информации.

При формулировании конечных целей создания специального математического обеспечения возможны два подхода Первый подход будет иметь место в том случае, если система специального математического обеспечения будет создаваться ответственными лицами, отвечающими за функционирование объекта управления.

Практика показывает, что в тех органах управления, где ответственные лица непосредственно берутся за решение этого вопроса, там сравнительно легко идут процессы внедрения автоматизации и создания специального математического обеспечения. Тогда задача решается значительно легче, так как могут быть сформулированы цели, остается только придать определенный смысл функционированию системы и разработать комплекс программ Где же эта работа поручается лицам второстепенным, не имеющим возможности принимать ответственные решения, там, как правило, она идет очень вяло, без видимых конечных сроков завершения.

Возможен в принципе и другой подход, обусловленный трудностью формулирования конечной цели. Действительно, нельзя требовать от ответственных лиц обязательного формулирования конечной цели Она может быть очень неясной. И тогда возможен такой подход конечная цель формулируется коллективом ученых, разрабатывающим специальное математическое обеспечение Дается оценка конечной цели процесса автоматизации, оценивается возможность реализации конечной цели и определяются связанные с этим трудности Затем конечная цель автоматизации утверждается.

При разработке специального математического обеспечения требуется согласование с внутренней структурой самой автоматизированной системы управления Например, по выходным документам должны быть согласованы требования не только по форме, но и по содержанию, при этом обязательно накладываются определенные требования на операционные системы, на технические средства, которые обеспечивают выдачу конечных документов

Одним из важных моментов является одноразовый ввод и одноразовая подготовка к вводу информации Это требует согласования с одноразовым вводом информации при создании автоматизированных систем управления Имеются задачи с огромным объемом входной и выходной информации, но расчеты по ним, приходящиеся на единицу информации, незначительны Это особенно характерно для решения различных экономических задач. Могут быть задачи и совершенно противоположного характера, имеющие небольшой объем входной информации и значи-

тельный объем расчетов В этих случаях должно соблюдаться требование одноразового ввода информации В тех задачах, в которых приходится иметь дело с подготовкой информации большого объема, когда основное время ЭВМ расходуется на работу с различными массивами информации, на их сортировку, выборку, отождествление, сравнение и другие процедуры логического плана, исходные данные должны быть введены один раз и храниться в системе, с тем чтобы система в процессе работы могла обращаться к этим исходным данным.

Специальное математическое обеспечение предусматривает постоянное накопление массивов информации в системе. Информация должна накапливаться, как правило, на магнитных лентах При этом процесс обновления информации можно организовать независимо от процесса решения задачи. Это можно делать постоянно или с определенной организованной периодичностью по мере поступления новых исходных данных При накоплении массивов информации всегда должна быть возможность обмена данными между машинами и системами, а также данными, хранящимися на различных машинах в различных системах. Обмен может быть простым — обмен магнитными лентами, и сложным — обмен информацией по каналам связи.

Создание специального математического обеспечения требует специальной организационно-технической службы, предназначенной для подготовки первичной информации, придания ей вида, удобного для обновления массивов информации, создания нормативной базы и для выполнения других функций Эта служба должна являться связующим звеном между специальным математическим обеспечением и техническими средствами системы управления Должна быть обеспечена возможность наращивания системы специального математического обеспечения как информационными процессами, так и специально разрабатываемыми функциональными программами и всевозможными обслуживающими программами, должна быть предусмотрена необходимость накопления массивов информации, обновления массивов информации, система должна проводить переработку рабочих программ так, чтобы можно было изменить задачи, конечные цели, методы решения этих задач.

При создании автоматизированных систем управления, как правило, придерживаются тех же принципов, что и при разработке математического обеспечения Более того, когда отсутствует специальное математическое обеспечение, автоматизированная система управления должна создаваться во взаимодействии с процессом разработки специального математического обеспечения Нельзя создавать только специальное математическое обеспечение, не учитывая всех возможностей построения автоматизированной системы управления.

Три элемента АСУ специальное математическое обеспечение, комплекс технических средств и средства автоматизации программирования должны создаваться одновременно Одновременное создание этих трех элементов АСУ на практике вызывает очень серьезные трудности При незавершенной автоматизированной системе и при отсутствии достаточно эффективных средств автоматизации программирования разработка специального математического обеспечения не может привести к желаемому результату или будет связана с большой затратой времени и значительными трудовыми затратами Следует иметь в виду и то обстоятельство, что внесение любых изменений в конструкцию ЭВМ, даже небольших и незначительных, приведет к изменению большинства программ операционных систем и специального математического обеспечения. В этих условиях итерационный процесс взаимной увязки трех элементов АСУ очень сложен Конечно, предпочтительным является такой порядок разработки АСУ, когда сначала создается комплекс технических средств АСУ, вместе с общим математическим обеспечением,

а затем разрабатывается с использованием имеющихся средств автоматизации программирования специальное математическое обеспечение. Такой вариант работ не исключен, поскольку по трудоемкости наибольшую часть всего процесса разработки АСУ составляет создание специального математического обеспечения Известны данные по трудозатратам при создании в целом всех этих средств из мировой практики: 60% трудозатрат приходится на разработку всей математики и 40% — на создание технических средств

Рассматривая вопрос об автоматизации программирования, следует иметь в виду программирующие системы. Под программирующей системой мы понимаем иерархически построенную систему с чрезвычайно крупными блоками типа операторов на верхнем уровне. По мере снижения уровня программирующей системы блоки по своему функциональному назначению становятся более специализированными и доводятся до элементарных событий.

Программирующая система, в том числе система комплексирования и система документирования, создается в зависимости от характера практических задач. Она нацелена на автоматизацию процессов программирования и в итоге на создание специального математического обеспечения А с другой стороны, она имеет самую непосредственную связь с общим математическим обеспечением

При разработке специального математического обеспечения предполагается создание информационной системы как основы всего математического обеспечения Автоматизированные информационные системы предназначаются для накопления, хранения, обновления, поиска, обработки и выдачи по запросам сведений различного характера. Наряду с техническими средствами в состав информационных систем входят комплекс специальных программ, обеспечивающих ее функционирование, информация, массивы сообщений, содержащих сведения учитываемые в системе, и различные служебные массивы словари, таблицы и т. п В дальнейшем понятие автоматизированных систем будем ограничивать этим комплексом программ и информации.

Автоматизированные информационные системы должны разрабатываться на основе использования общего информационного поля для решения различных задач. Практика работы над информационными системами неоспоримо подтвердила необходимость создания комплекса программ системного применения, а не разработку частных информационных задач. Это обусловлено однотипностью процедур, выполняемых при решении достаточно широкого класса информационных задач, удобством централизованной подготовки и централизованного хранения информации в общем информационном поле В связи с этим понятие информационной задачи относится к внешнему проявлению информационной системы и связывается непосредственно с ее структурой Основным структурным элементом массива сведений, хранящихся в памяти системы, является сообщение В сообщении указывается наименование объекта и его признаки, характеристики

Наиболее удобным для человека и универсальным является естественный язык Однако сложность синтаксической и семантической структуры естественных языков с точки зрения их машинной обработки не позволяет рассчитывать на их использование в ближайшем будущем. В связи с этим общение человека с системой должно осуществляться с использованием входных и выходных сообщений на формализованном естественном языке

Анализ реальных информационных систем приводит к заключению о нецелесообразности создания универсальных автоматизированных систем Опыт работы в этой области свидетельствует о том, что следует идти по пути создания ограниченного числа базовых автоматизированных информационных систем, специализированных в дальнейшем в соответствии с конкретными требованиями определенных классов информационных задач. Здесь имеется в виду, что информационные системы, возможно, целесообразно (это надо еще детально изучить) создавать по потребителю

Есть потребитель, который применяет задачи типа план-заказ, план-распределение ресурсов, задачи по оперативному учету продукции — это одного рода задачи Есть задачи, связанные с обеспечением информацией руководства различного уровня Это совсем другое дело, здесь нет, в сущности, никаких учетных задач. Возможно создание систем по потребителю Вопрос о потребном перечне таких систем целесообразно проработать А в качестве базовых автоматизированных систем как по потребителю, так и по своей внутренней структуре предлагается иметь базовые автоматизированные системы фактографического типа и базовые автоматизированные системы документального типа, т е. два типа автоматизированных информационных систем.

Каждая из этих систем должна быть построена иерархически и, главным образом, по модульному принципу Под модулем мы понимаем не просто блок системы, а такой элемент информационной системы, который является универсальным по входу и по выходу, т. е. который выполняет определенные функции и может быть вписан в любую информационную систему, в которой такая функция имеется Модульный принцип создания информационной системы будет гарантировать от возможных переработок программ в случае даже значительных изменений в постановках задач Необходимо создать для этого библиотеку модулей по типу библиотеки известных процедур интерпретирующей системы.

Документальные и фактографические системы отличаются не только содержанием хранимых в них сведений, — различие в них, в основном, в процессах обновления и поиска информации. При разработке этих систем основное внимание должно быть обращено на создание базовой автоматизированной фактографической информационной системы, как наиболее сложной по структуре и перекрывающей большую часть потребностей в информационном обеспечении работ различных органов управления Информационная система должна строиться в основном на базовой автоматизированной фактографической информационно-поисковой системе. Это главное ее содержание.

Автоматизированная фактографическая информационно-поисковая система является интерпретатором, осуществляющим перевод сообщений, запросов с входного информационного языка на машинный информационный язык и обратно и выполняющим предписания относительно характера обработки информации

Перевод запроса с входного языка на машинный, информационный — это первый и наиболее сложный этап его обработки При этом распознаются отдельные элементы запроса, слова и словосочетания и анализируется его логическая структура. На следующем этапе устанавливаются смысловые связи между понятиями, используемыми в сообщениях, а также проводится выборка сообщений, удовлетворяющих условия запроса

Автоматизированная информационная система должна иметь подсистему автоматического редактирования буквенно-цифровой информации. Эта подсистема предназначена для подготовки к выдаче информации на печатающие устройства типа АЦПУ, рулонный телетайп, а также на устройство наглядного отображения коллективного или индивидуального пользования. Данные, необходимые для обеспечения подготовки информации к выдаче на эти устройства, записываются во входном сообщении на специальном алгоритмическом языке автоматического документирования Перевод на машинный язык и подготовка информации к выдаче осуществляются с помощью специального транслятора-интерпретатора. Подсистема должна обеспечивать выдачу информации в тек-

стовой, табличной, регламентной и нерегламентной форме. В последнем случае основная информация по структуре выдаваемых документов определяется в процессе поиска информации. Таким образом, базовая автоматизированная система должна обеспечивать выдачу практически произвольных форм документов

В автоматизированной информационной системе может функционировать подсистема подготовки информации для выдачи ее на устройства наглядного отображения. Подсистема включает группу специальных программ, реализующих поиск информации, и программы преобразования информации, обеспечивающие редактирование полученной в результате поиска информации в виде, необходимом для ее выдачи на устройства наглядного отображения

Важной подсистемой также является автоматизированная словарная служба, которая предназначается для автоматического пополнения словарей. Подсистема должна обеспечивать работу в автономном режиме, режиме пополнения словарей и автоматизированного установления связи между элементами словарей, а также должна обеспечивать автоматизацию трудоемких работ по построению словарей применительно к вариантам практического использования базовой системы в интересах автоматизации и конкретных органов управления. Оперативно в состав информационных систем должна быть включена подсистема программного администрирования, позволяющая персоналу, обслуживающему машину, воспользоваться этой информацией

Итак, при разработке базовых автоматизированных информационных систем должны учитываться следующие обстоятельства:

1. Необходимо разрабатывать два типа базовых информационных систем по их внутреннему содержанию (фактографические и документальные), а также несколько систем или одну, если это окажется возможным, специализированную по потребителю.

Базовые информационные системы должны учитывать общие потребности решения информационных задач автоматизированных органов управления. Частные требования должны обеспечиваться специальными программными средствами, согласованными с базовой системой. Имеется в виду, что к базовой системе должны приспосабливаться специальные программы, учитывающие особенность функционирования того или иного органа.

2. Базовые системы должны создаваться с ориентацией на формализованный входной язык. Входной язык должен обеспечивать возможность расширения его изобразительных средств.

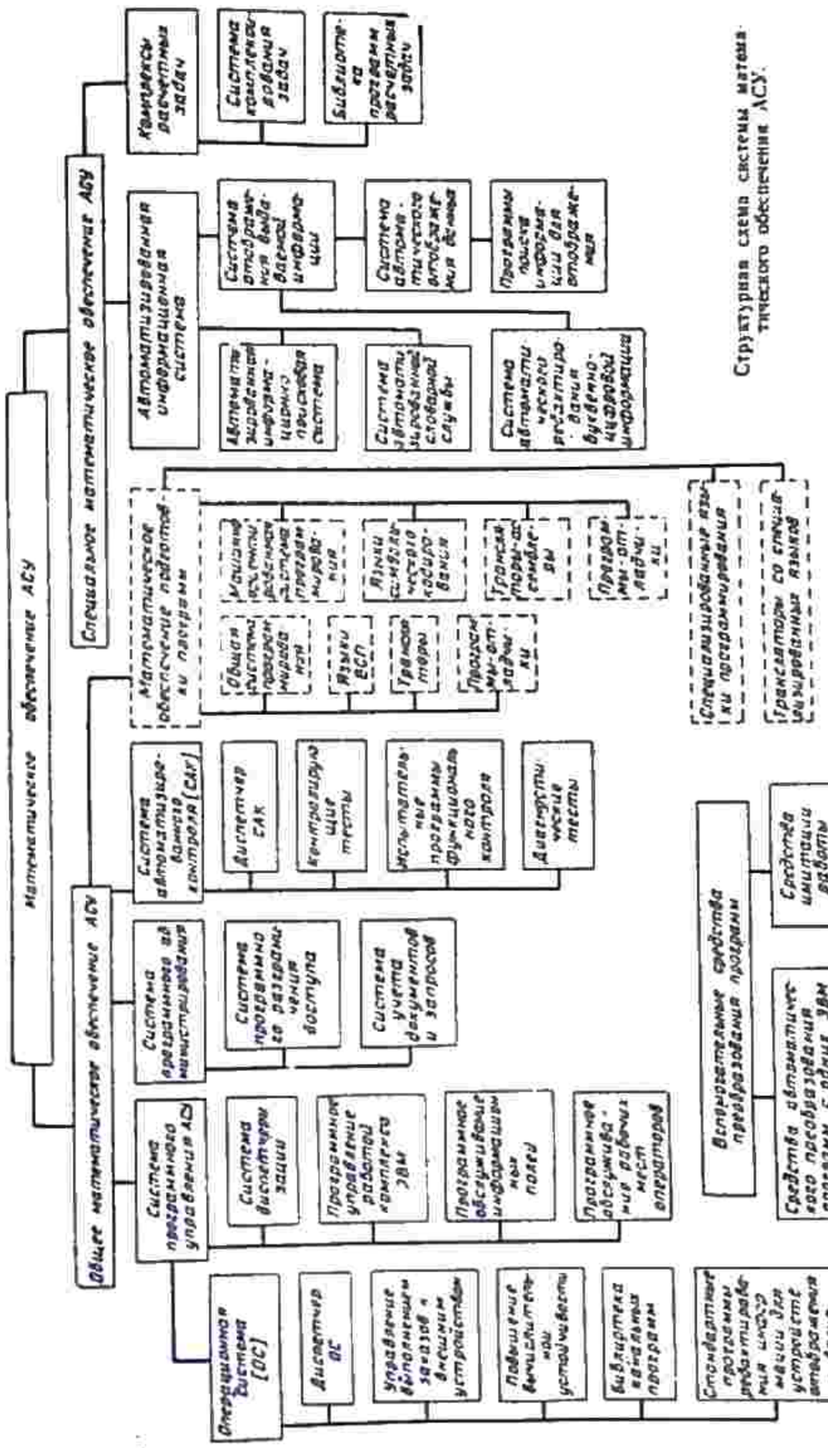
3. Система должна строиться на базе типовых функциональных блоков или программных модулей.

4. Для представления информации в памяти машины должны использоваться ассоциативные методы и такая структура массивов, которая обеспечивает быстрый поиск информации и удобство ее обновления.

5. Массивы информационных сведений и словари должны представляться в памяти машины в виде модулей, блоков, допускающих свободное перемещение по полю оперативной памяти.

6. При создании базовых информационных систем основное внимание должно быть уделено разработке базовой фактографической информационной системы как системы, в наибольшей мере учитывающей потребности решения информационных задач в высших органах управления.

7. При разработке базовых автоматизированных информационных систем должны широко использоваться все доступные средства автоматизации программирования и прежде всего языки символического кодирования и соответствующие трансляторы-ассемблеры.



Структурная схема системы математического обеспечения АСУ.

УДК 681.3.51

К ВОПРОСУ ОЦЕНКИ ВЕРОЯТНОСТИ ОШИБКИ В ВЫДАВАЕМЫХ ВЫЧИСЛИТЕЛЬНЫМ ЦЕНТРОМ ДАННЫХ

В. В СЕМАКОВ

Вероятность появления ошибки в данных, выдаваемых вычислительным центром, является одним из основных показателей качества решения задачи. Ошибка в выдаваемых данных появляется в тех случаях, когда отклонение величины выданного показателя от его достоверного значения превышает заданную меру точности [1].

Определение вероятности ошибки в выдаваемых данных является весьма важным вопросом при проектировании вычислительных центров, предназначенных для обработки экономической информации.

Установим основные частные составляющие ошибки выдаваемых данных, имея в виду, что вычислительный центр, предназначенный для обработки экономической информации, является человеко-машинной системой. Будем считать, что обработка информации на вычислительном центре выполняется в соответствии с блок-схемой, приведенной на рис. 1. В принятом технологическом процессе имеется пять основных этапов.

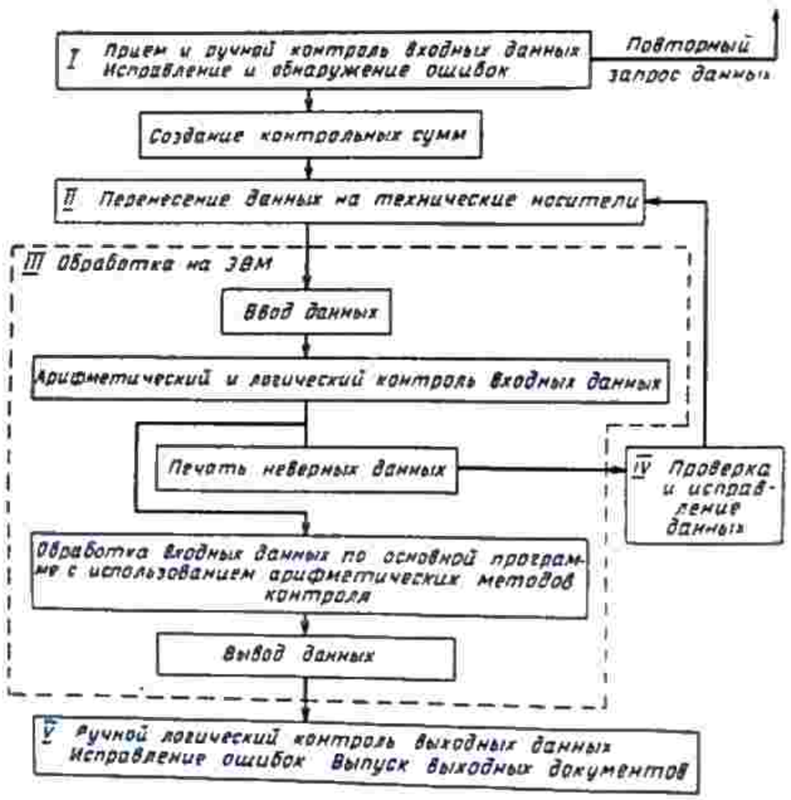


Рис. 1. Укрупненная схема технологического процесса обработки данных на вычислительном центре.

На первом этапе осуществляется ручной логический контроль и исправление обнаруженных ошибок. На этом этапе целесообразно использовать простые методы логического контроля, а именно — проверку наличия необходимых показателей, отсутствия резких отклонений в величинах показателей («диких» показателей) и т. д. Для исправления обнаруженных ошибок может потребоваться повторный запрос данных у отправителя (заказчика).

На втором этапе выполняются работы по перенесению данных на технические носители (перфокарты, перфоленты), а также работы по контролю перфорации (верификация, счетный контроль и т. д.) и исправлению пробитых перфокарт.

На третьем этапе после ввода данных в ЭВМ осуществляется арифметический и логический контроль входных данных. Обработке данных по основной программе производится после исправления обнаруженных ошибок или «выброса» ошибочных данных (для последующего их исправления). Арифметический контроль осуществляется на основе проверки контрольных сумм. Логический контроль входных данных может осуществляться: сопоставлением показателей отчетного периода с такими же показателями предшествующего периода; проверкой предельных величин показателей; по показателям, находящимся в определенном соотношении с другими показателями, проверкой этих соотношений и т. п. При обработке входных данных по основной программе для уменьшения ошибок из-за сбоев в работе ЭВМ используются арифметические методы контроля, например метод двойного счета.

На четвертом этапе производится проверка данных, выданных ЭВМ в результате арифметического и логического контроля. Для исправления обнаруженных ошибок может потребоваться повторный запрос данных.

На пятом этапе могут использоваться различные методы логического контроля. Этот контроль осуществляется операторами или экономистами. Для устранения обнаруженных ошибок может потребоваться повторное полное или частичное решение задачи.

Подэтап «создание контрольных сумм» обеспечивает проведение машинного арифметического контроля введенных в ЭВМ данных с целью обнаружения ошибок, внесенных в процессе перфорации и ввода данных в ЭВМ.

Для принятой схемы технологического процесса обработки данных вероятность выходной ошибки определяется следующими основными частными составляющими.

1. Вероятностью ошибки во входных данных из-за неправильности исходных документов и искажений при передаче данных по каналам связи с учетом использования методов повышения достоверности передачи данных (P_{01}).

2. Вероятностью внесения ошибки во входные данные на этапе подготовки технических носителей информации с учетом использования методов их контроля: верификации, счетного контроля и других арифметических методов контроля на ЭВМ (P_{02}).

3. Вероятностью появления ошибок в процессе обработки данных по основной программе ЭВМ с учетом используемых методов контроля (двойным счетом и других арифметических методов контроля) (P_{03}).

4. Вероятностью появления ошибки в выдаваемых данных, обусловленной сбоями в работе устройств вывода данных на ЭВМ (P_{04}).

Приведенные четыре группы составляющих ошибок, определяющих ошибку выходных данных, являются независимыми случайными ошибками. Заметим, что в процессе передачи данных по каналам связи, машинной обработки или выдачи данных из ЭВМ из-за отказов в работе аппаратуры могут иметь место систематические ошибки. Однако эти ошибки обнаруживаются, после устранения отказов аппаратуры произ-

водится повторная передача данных, повторное решение задачи (или ее части) и систематические ошибки устраняются.

Анализируя приведенные составляющие ошибок, видим, что ошибки на этапах подготовки технических носителей информации и в процессе обработки данных по основной программе могут быть уменьшены за счет применения арифметических и логических методов контроля, в то время как ошибки во входных данных из-за неправильности исходных документов и ошибки, обусловленные сбоями в работе устройств вывода, могут быть уменьшены только за счет использования логических методов контроля.

Ошибка в выходных данных определяется не только приведенными выше составляющими: на нее влияют ошибки, обусловленные сбоями в работе устройств ввода данных ЭВМ, а также ошибки в условно-постоянной информации, используемой для решения задач. Анализ работы ВЦ показал, что при нормально работающих устройствах ввода, при соответствующей организации контроля ввода данных в ЭВМ и при правильном ведении условно-постоянной информации вероятность появления ошибок из-за сбоев в работе устройств ввода данных и вероятность ошибок в условно-постоянной информации будет (на каждый знак, символ) меньше 10^{-8}. Как будет показано ниже, вероятность ошибки во входных данных, поступающих для обработки на центр, может быть принята в среднем за 10^{-4} на каждый знак. Поэтому при оценке вероятности ошибок в выходных данных указанными двумя составляющими можно пренебречь.

Выходная ошибка будет уменьшаться за счет исправления данных в результате использования ручных и машинных логических методов контроля на соответствующих этапах технологического процесса обработки данных. В зависимости от конкретных условий решения задачи и от того, на каком этапе обнаружена ошибка, для устранения обнаруженных ошибок производится повторный запрос необходимых данных, сличение данных на техническом носителе с входным документом, повторное решение задачи или части задачи на ЭВМ и т. д. В некоторых случаях возможна автоматическая коррекция ошибок при использовании специальной программы ЭВМ.

Эффективность логических методов контроля может быть выражена следующими показателями:

1 Степенью уменьшения вероятности ошибок во входных данных за счет использования на этапе их приема ручных логических методов контроля (R_{01})

2 Степенью уменьшения вероятности ошибок во входных данных в результате применения машинных логических методов контроля на первом подэтапе обработки данных на ЭВМ (R_{02}).

3 Степенью уменьшения ошибок в выходных данных на основе использования ручных логических методов контроля на этапе выпуска документов (R_{03}).

Заметим, что вероятности ошибок определяются статистически как частоты, полученные из опыта при многочисленных испытаниях. Естественно, что эти вероятности могут существенно изменяться от различных факторов: квалификации персонала, конструктивных особенностей ЭВМ, используемых программных методов контроля работы ЭВМ и т. п. Ниже приведены некоторые данные по вероятностям появления ошибок в предположении, что операторы и инженерно-технический персонал ВЦ имеют среднюю квалификацию, а для решения задач используются современные ЭВМ при соответствующей организации контроля их работы.

Имея в виду приведенную на рис. 2 схему, выведем расчетные формулы для оценки вероятности ошибки в данных, выдаваемых ВЦ. При этом предполагаем, что на различных этапах (или подэтапах) технологического процесса обработки данных используются различные логи-

ческие методы контроля, независимые между собой. Для удобства вычисления величин P_{01}, P_{02}, P_{03}, P_{04} определяем как вероятности появления ошибок по знакам (символам), принимая, что появление ошибок для различных знаков (символов) имеет равную вероятность. Значения величин R_{01}, R_{02}, R_{03} зависят от следующих основных факторов: от вероятности ошибок в данных, поступающих для обработки на соответствующих этапах технологического процесса; от сложности решения задач и специфики входных и выходных данных; от используемых методов контроля и конкретных условий их реализации. Производим оценку вероятности появления ошибки в данных, полученных в результате решения определенной задачи или группы задач при условии $P_{01}=\text{const}$ и использования конкретных методов контроля. При перечисленных предположениях можно принять, что величины R_{01}, R_{02}, R_{03} имеют постоянные значения.

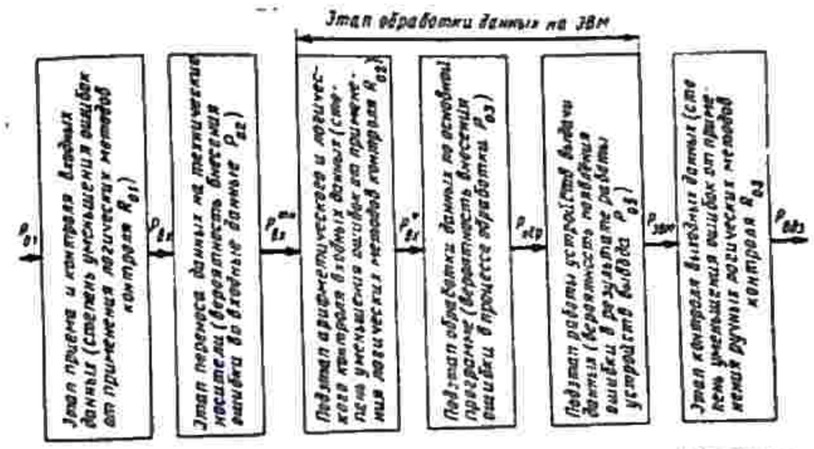


Рис. 2. Схема процесса оценки вероятности ошибки в выдаваемых ВЦ данных.

На этапе приема входных данных после выполнения операций по логическому ручному контролю и устранению ошибок в поступивших данных вероятность ошибки может быть определена из следующего выражения:

$$P_{вх}=\frac{P_{01}}{R_{01}}. \qquad (1)$$

В результате выполнения операций на этапе подготовки технических носителей информации вероятность ошибки во входных данных, поступающих для дальнейшей обработки, определяется следующим выражением:

$$P_{вх}^{тн}=1-(1-P_{вх})(1-P_{02}). \qquad (2)$$

Имея в виду формулу (1) и учитывая, что на практике выполняются условия $P_{вх}\ll 1$ и $P_{02}\ll 1$, получаем

$$P_{вх}^{тн}\approx\frac{P_{01}}{R_{01}}+P_{02}. \qquad (3)$$

Логические методы контроля позволяют обнаруживать ошибки, возникающие как при подготовке исходных документов и передаче данных по каналам связи, так и при обработке данных на ВЦ. Поэтому после этапа машинного логического контроля входных данных с учетом операций по исправлению обнаруженных ошибок вероятность ошибки во

входных данных, полученных в результате такой обработки, определяется следующим выражением:

$$P^*_{вх} \approx \frac{P_{o1}}{R_{o1}R_{o2}} + \frac{P_{o2}}{R_{o3}}. \quad (4)$$

В результате выполнения основной программы обработки входных данных на ЭВМ с применением арифметических методов контроля для уменьшения случайных ошибок, оценка вероятности ошибки в данных, обработанных таким образом, может быть получена по формуле

$$P_{обр} \approx 1 - (1 - P^*_{вх})(1 - P_{o3}) \approx \frac{P_{o1}}{R_{o1}R_{o2}} + \frac{P_{o2}}{R_{o3}} + P_{o3}. \quad (5)$$

С учетом ошибок, возникающих из-за сбоев в работе устройств вывода, вероятность ошибки в данных, полученных (отпечатанных, отперфорированных и т. п.) в результате обработки на ЭВМ,

$$P_{ЭВМ} \approx \frac{P_{o1}}{R_{o1}R_{o2}} + \frac{P_{o2}}{R_{o3}} + P_{o3} + P_{o4}. \quad (6)$$

После уменьшения ошибок в выдаваемых данных за счет использования ручных логических методов контроля, оценка вероятности ошибки в выходных данных может быть получена по формуле

$$P_{вых} \approx \frac{P_{o1}}{R_{o1}R_{o2}R_{o3}} + \frac{P_{o2}}{R_{o2}R_{o3}} + \frac{P_{o3}}{R_{o3}} + \frac{P_{o4}}{R_{o3}}. \quad (7)$$

На различных этапах технологического процесса обработки информации могут быть использованы одинаковые методы логического контроля (например, такие методы, как проверка предельных величин показателей или сопоставление показателей отчетного периода с показателями предшествующего периода). На практике для логического ручного контроля выходных данных обычно используются методы, которые уже применялись при ручном или машинном логическом контроле. При таком условии методы логического контроля, применяемые после выдачи данных из ЭВМ, практически будут эффективными только по уменьшению ошибок, возникающих при машинной обработке и выводе данных из ЭВМ. Поэтому при условии применения для контроля выходных данных ЭВМ логических методов, одинаковых по своему принципу действия с методами, используемыми для контроля входных данных, вероятность ошибки в данных, выдаваемых вычислительным центром, следует определять по формуле

$$P_{вых} \approx \frac{P_{o1}}{R_{o1}R_{o2}} + \frac{P_{o2}}{R_{o3}} + \frac{P_{o3}}{R_{o3}} + \frac{P_{o4}}{R_{o3}}. \quad (8)$$

Необходимо отметить, что формулы для определения вероятности ошибок в выходных данных получены в предположении наиболее сложного технологического процесса обработки данных (наличие этапов передачи исходных данных по каналам связи и переноса данных на технические носители информации; использование логических ручных методов контроля на этапах приема и выпуска данных, а также логических машинных методов контроля входных данных). Если в технологическом процессе обработки данных на ВЦ отсутствуют некоторые этапы или подэтапы обработки, приведенные на схеме рис. 2, то в формулах (7) и (8) соответствующие вероятности принимаются равными нулю, а соответствующие степени уменьшения ошибок — равными единице.

Наиболее существенными составляющими ошибки, в значительной степени определяющими ошибку входных данных, поступающих для обработки на ЭВМ, являются: ошибки из-за неправильности заполнения

исходных документов и ошибки при передаче данных по каналам связи. Вероятность ошибки, обусловленная этими составляющими, может быть принята в среднем равной 10^{-4} на каждый знак [2].

При перенесении данных с исходных документов на перфоносители вероятность искажения знака в среднем имеет порядок 10^{-3} [2]. Для уменьшения ошибок, вносимых при перфорации, обычно применяется верификация или счетный контроль. Метод верификации позволяет обнаруживать до 93% допущенных при перфорации ошибок, а метод счетного контроля — до 99% [3]. Таким образом, вероятность искажения знака во входных данных на этапе подготовки технических носителей информации (P_{o2}) при использовании верификаций имеет величину порядка $7 \cdot 10^{-5}$, а при использовании счетного контроля 10^{-5}.

Таким образом, без применения логических методов контроля входных данных на ВЦ вероятность появления ошибок (по знакам) на входе ЭВМ будет порядка 10^{-4}. Если не принять специальных мер по уменьшению ошибок в процессе машинной обработки, то выходные данные будут иметь недопустимо низкую достоверность.

При обработке экономической информации принимается допустимой вероятности ошибки в выходных данных в пределах $10^{-5} \div 10^{-7}$ [3]. Следовательно, необходимо ввести в технологический процесс обработки данных методы контроля, на основе которых можно уменьшить ошибку примерно в 10—1 000 раз.

В случае использования правильно построенных программ обработки данных с контролем сбойных ошибок и при условии нормальной работы ЭВМ, а также устройств вывода данных, вероятности появления ошибок, обусловленных сбоями в ЭВМ и устройствах вывода, весьма малы (менее 10^{-5}). Поэтому для обеспечения выдачи данных с приемлемой достоверностью в ряде случаев можно ограничиться использованием на ВЦ логических ручных методов контроля на этапе приема данных и логических машинных методов контроля правильности входных данных при помощи ЭВМ.

При принятых условиях организации логического контроля формула (7) принимает следующий вид:

$$P_{вых} \approx \frac{P_{o1}}{R_{o1}R_{o2}} + \frac{P_{o2}}{R_{o3}} + P_{o3} + P_{o4}. \quad (9)$$

Ручной логический контроль исходных данных и устранение ошибок, обнаруженных при этом контроле, целесообразно осуществлять не на ВЦ, а в той организации, которая выдает эти данные для обработки. Для этого должны быть выделены специальные экономисты-контролеры, досконально знающие особенности исходных данных.

В случае выполнения ручного логического контроля данных вне вычислительного центра и осуществления на ВЦ операций по переносу данных с документов на перфоносители, оценка ошибки в выходных данных может производиться по формуле

$$P_{вых} \approx \frac{P_{o2}}{R_{o3}} + \frac{P_{o1}}{R_{o3}} + P_{o4} + P_{o5}. \quad (10)$$

При правильной организации машинного логического контроля входных данных ошибка может быть уменьшена в 10—100 раз. Указанное уменьшение ошибок R_{o2} будет получено в случае использования для контроля различных логических методов и, в частности, метода проверки соответствия одного показателя другому показателю, для чего может потребоваться введение во входные данные избыточной информации.

В качестве примера приведем некоторые сведения об эффективности машинного контроля входных данных при решении на ЭВМ «Минск-22» в ВЦ ЦСУ БССР задач посылочной торговли. Логический контроль правильности данных, входящих в состав заказов населения, осуществляется в ЭВМ следующим образом: 1) проверяется, не превышают

ли расходы (тарифы) заказа максимально возможной величины (четырнадцати), 2) номенклатурные номера контролируются по количеству знаков, 3) проверяется соответствие номенклатуры заказываемого изделия его цене (производится сравнение с ценой справочника в памяти ЭВМ). При обработке 825 заказов населения, имеющих объем порядка 60 000 реквизитов (оснований и реквизитов-признаков), в результате логического контроля было обнаружено 104 ошибочных данных, которые исключили из машинной обработки (для внесения соответствующих исправлений). В выходных данных счетов-фактур, которые выдаются ЭВМ ежедневно при обработке 800—1 000 заказов, наблюдаются единичные ошибочные данные (ошибка в одном или двух реквизитах). Следовательно, логические машинные методы контроля заказов населения обеспечивают уменьшение ошибок в 50—100 раз [4]

Приведенные выше формулы позволяют определить вероятность появления ошибок в выходных данных по знакам. Наиболее существенное значение имеют ошибки в цифровых данных. Поэтому особое внимание следует уделять устранению ошибок при печати (перфорации) цифр.

Вероятность появления ошибки в выходных показателях может быть определена по формуле

$$P_{вдд} \approx 1 - (1 - P_{вдз})^{n_{вв}}, \qquad (11)$$

где $n_{вв}$ — средневзвешенное количество цифр выдаваемых показателей, искажение которых приводит к нарушению заданной меры точности.

Рассмотрим пример определения вероятности ошибки в выдаваемых ВЦ данных применительно к рассмотренному выше случаю решения на ЭВМ «Минск-22» задач посылочной торговли. Обработка заказов населения осуществляется в соответствии со схемой рис. 1, однако на ВЦ не осуществляется ручной логический контроль входных и выходных данных. В этом случае для определения ошибок в выходных данных следует использовать формулу (10). Заметим, что выходные данные печатаются в двух экземплярах на алфавитно-цифровом печатающем устройстве ЭВМ «Минск-22». На ВЦ для уменьшения ошибок перфорации применяется верификация; для уменьшения ошибок входных данных, введенных в ЭВМ, — логические методы контроля (указанные в примере) и метод контрольных сумм, для уменьшения ошибок от сбоев в работе ЭВМ — метод двойного счета. В данных условиях принимаем $P_{вх} = 10^{-4}$; $R_{вп} = 50$, $P_{вп} = 7 \cdot 10^{-6}$; $P_{оп} = 10^{-4}$; $P_{оо} = 5 \cdot 10^{-6}$.

Следует отметить, что величина $P_{вп}$ получена из следующих соображений. Вероятность искажения знака на этапе подготовки технических носителей с использованием для контроля верификации в среднем равна $7 \cdot 10^{-6}$. Ошибки перфорации будут уменьшаться за счет использования после ввода данных в ЭВМ метода контрольных сумм. Повторные методы контроля уменьшают ошибку примерно на один порядок. Поэтому получаем $P_{вп} = 7 \cdot 10^{-6}$.

При принятых величинах $P_{вх}$, $P_{вп}$, $R_{вп}$, $P_{оп}$, $P_{оо}$ по формуле (10) получаем вероятность искажения знака в выходных данных $P_{вдз} \approx 7 \times 10^{-6}$.

При решении данной задачи средневзвешенное количество цифр в выдаваемых показателях имеет величину, примерно равную 3, причем искажения недопустимы во всех разрядах отпечатанных цифр. Поэтому в соответствии с формулой (11) получаем вероятность появления ошибки в выходных показателях $P_{вдп} \approx 2 \cdot 10^{-5}$. Полученная вероятность соответствует частоте ошибок, получаемых на ВЦ при решении задач посылочной торговли.

На основе приведенного анализа можно сделать вывод, что при обработке экономической информации на вычислительном центре необходимо после ввода данных в ЭВМ предусматривать специальный под-

этап машинной обработки — логический и арифметический контроль входных данных. Для повышения эффективности контроля на этом подэтапе следует применять несколько различных логических методов контроля.

ЛИТЕРАТУРА

1. Боярский А. Я. Экономика и мера точности. В сб. «Статистика и электронно-вычислительная техника в экономике». Вып. II. Изд-во «Статистика», 1968.
2. Кулик В. Т. Алгоритмизация объектов управления. Изд-во «Наукова думка», 1968.
3. «Методические материалы по типовому составу технического задания на проектирование вычислительных центров». Изд-во «Статистика», 1970.
4. Смирнова А. В., Семаков В. В. Использование ЭВМ для автоматизации трудоемких операций посылочной торговли. В сб. «Электронно-вычислительная техника и программирование», № 4. Изд. Главного управления вычислительных работ ЦСУ СССР, 1971.

УДК 681.3.06

О РАСЧЕТЕ ВРЕМЕНИ ИСПОЛНЕНИЯ МАШИННЫХ ПРОГРАММ

В. А. ГЕРМАН

ВВЕДЕНИЕ

При решении различного рода задач, связанных с выбором вычислительных устройств для реализации конкретного алгоритма, возникает необходимость в оценке времени исполнения алгоритма на ЦВМ. При использовании ЦВМ в реальном масштабе времени иногда, кроме средних оценок времени исполнения, полезно знать и дисперсию времени исполнения алгоритма и другие вероятностные характеристики. Зная частоты исполнения отдельных блоков алгоритма, можно ставить и решать задачи оптимальной сегментации программ, оптимального распределения блоков алгоритма между несколькими вычислительными устройствами и ряд других.

В данной работе для исследования алгоритма используется его модель, являющаяся стохастическим графом, вершины которого соответствуют отдельным блокам алгоритма, а дуги обозначают возможные переходы от одного блока к другому. Каждой вершине X_i приписана некоторая неслучайная величина c_i — продолжительность исполнения блока, а дуге — величина p_{ij}, являющаяся вероятностью соответствующего перехода. Без ограничения общности будем считать, что существует только одна начальная вершина, с которой всегда начинается исполнение алгоритма, и только одна конечная вершина, при достижении которой исполнение алгоритма заканчивается. Ниже рассматривается задача нахождения средних частот и их дисперсий для каждого блока алгоритма и задача определения среднего времени исполнения алгоритма и дисперсии времени исполнения.

Частично подобные задачи решались и раньше. Так, в работе [1] использовалось адекватное представление стохастической граф-модели дискретной марковской цепью с поглощающими состояниями. Средние частоты каждого блока определялись методом последовательного возведения в степень матрицы переходных вероятностей. Однако следует признать такой метод неэффективным для исследования сложных циклических программ с большим числом блоков, так как его реализация требует большой памяти и большого объема вычислений.

Задача определения среднего времени и дисперсии решалась Рамамурти [2] с помощью производящих функций и правил Мэзона для

ориентированных графов. Метод, изложенный в работе [2], может эффективно использоваться только для моделей с малым числом состояний, например для синтеза оптимальных по скорости микропрограмм.

В работах Мартина и Эстрина [3, 4] используется процедура перехода от циклической структуры граф-модели к ациклической. При этом значительно упрощается расчет среднего времени исполнения алгоритма. Однако такая процедура перехода изменяет вероятностные связи между блоками, и с ее помощью не представляется возможным подсчитать частоты исполнения блоков и дисперсию времени исполнения всего алгоритма.

В данной работе также используется некоторое преобразование графа, позволяющее достаточно просто рассчитывать временные и частотные параметры граф-модели алгоритмов.

Предлагаемые способы расчета имеют, на наш взгляд, две существенные особенности. Первая — состоит в том, что решение всех вышеперечисленных задач объединяется общей схемой расчета. Вторая особенность заключается в широком использовании в расчетах структурных свойств граф-моделей. Учитывая то обстоятельство, что описываемые алгоритмы предназначены для машинной обработки граф-моделей большой размерности, все «структурные» операции сводятся к действиям над соответствующими булевыми матрицами.

Работа состоит из трех разделов.

В первом разделе выводятся формулы для расчета временных и частотных параметров граф-модели. Во втором разделе вводятся в рассмотрение типовые фрагменты граф-моделей для упрощения расчетов и излагается метод декомпозиции больших граф-моделей. И, наконец, в заключительном разделе дается описание алгоритма расчетов.

1. ОСНОВНЫЕ ФОРМУЛЫ

Рассмотрим ориентированный связный граф $G(X, U)$, где $X=\{X_i\}$, $i=\overline{1, N+1}$ — множество вершин графа, а $U=\{u_{ij}\}$, $i, j=\overline{1, N+1}$ — множество ориентированных дуг, $u_{ij}=1$, если есть дуга, соединяющая вершины X_i и X_j, и $u_{ij}=0$ в противном случае. Всегда будем считать X_1 начальной вершиной, а X_{N+1} — поглощающей вершиной. Если каждой имеющейся дуге поставить в соответствие вероятность перехода p_{ij}, то тем самым будет задана конечная марковская цепь с состояниями $X=\{X_i\}$, $i=\overline{1, N+1}$ и матрицей перехода $P=\{p_{ij}\}$, $i, j=\overline{1, N+1}$. Каждая вершина X_i соответствует определенному блоку исследуемой программы. Кроме того, будем считать заданным вектор продолжительности исполнения блоков $c=\{c_i\}$, $i=\overline{1, N}$, в котором c_i — продолжительность исполнения блока с номером i.

Исследуемый процесс характеризуется тем, что при начальном состоянии X_1 система через конечное число шагов попадает в состояние X_{N+1}. При этом состояние X_i «исполняется» S_i раз. Среднее и дисперсию случайной величины S_i обозначим через $\bar{S}_i$ и $D(S_i)$. Время исполнения всего алгоритма можно представить в виде

$$T=\sum_{i=1}^{N} c_i S_i. \qquad (1)$$

Среднее и дисперсию величины T будем соответственно обозначать через $\bar{T}$ и $D(T)$. Используя (1), можно $\bar{T}$ и $D(T)$ выразить в общем виде.

$$\bar{T}=\sum_{i=1}^{N} c_i \bar{S}_i$$

и

$$D(T)=M\{(S_i-\bar{S}_i)^2\}.$$

Здесь и далее $M\{l\}$ обозначает математическое ожидание случайной величины l. Введем двоичную переменную ε_r^i, так что $\varepsilon_r^i=1$, если мы на r-м шаге попали в состояние X_i, и $\varepsilon_r^i=0$ в противном случае. Тогда

$$S_i=\sum_{r=1}^{\infty} \varepsilon_r^i.$$

Усредняя левую и правую части этого равенства, получаем

$$\bar{S}_1=1+\sum_{r=1}^{\infty} p_{11}^{(r)},$$

$$\bar{S}_i=\sum_{r=1}^{\infty} p_{1i}^{(r)}, \quad i=\overline{2, N}, \qquad (2)$$

где $p_{1i}^{(r)}$ — вероятность прихода в состояние X_i на r-м шаге, если начальное состояние системы X_1.

Преобразуем (2):

$$\bar{S}_i=\sum_{r=1}^{\infty} p_{1i}^{(r)}=\sum_{r=1}^{\infty}\sum_{j=1}^{N} p_{1j}^{(r-1)} p_{ji}=\sum_{j=1}^{N} p_{ji}\left(\sum_{r=1}^{\infty} p_{1j}^{(r-1)}\right)=\sum_{j=1}^{N} p_{ji}\bar{S}_j. \qquad (3)$$

Таким образом, $\bar{S}_i$ является решением системы уравнений

$$\bar{S}_1=1+\sum_{j=1}^{N} p_{j1}\bar{S}_j,$$

$$\bar{S}_i=\sum_{j=1}^{N} p_{ji}\bar{S}_j, \quad i=\overline{2, N}. \qquad (4)$$

Выведем формулу для подсчета дисперсии частоты попадания в состояние X_j. Так, где это не вызовет путаницы, мы будем опускать индекс j. Проведем несложные выкладки:

$$D(S)=M\{(S-\bar{S})^2\}=M\left\{\left(\sum_{r=1}^{\infty}\varepsilon_r\right)^2\right\}-\bar{S}^2=M\left\{\sum_{r=1}^{\infty}\varepsilon_r^2\right\}+$$

$$+2M\left\{\sum_{r=1}^{\infty}\sum_{l>r}\varepsilon_r\varepsilon_l\right\}-\bar{S}^2=\bar{S}-\bar{S}^2+2\sum_{r=1}^{\infty}\sum_{l>r}M\{\varepsilon_r\varepsilon_l\}. \qquad (5)$$

В терминах марковской цепи $M\{\varepsilon_r\varepsilon_l\}$ является вероятностью попадания на r-м шаге в состояние X_j при условии выхода из состояния X_1, умноженной на вероятность того, что выйдя из состояния X_j, можно вернуться в него же через $(l-r)$ шагов.

$$D(S)=\bar{S}-\bar{S}^2+2\sum_{r=1}^{\infty}\sum_{l>r} p_{1j}^{(r)} p_{jj}^{(l-r)}.$$

Преобразуя двойную сумму, получаем

$$D(S_j)=\bar{S}_j-\bar{S}_j^2+2\bar{S}_j R_j, \qquad (6)$$

где

$$R_j=\sum_{r=1}^{\infty} p_{jj}^{(r)}.$$

В тех случаях, когда система, выйдя из состояния X_j, никогда туда не

возвращается, $R_j=0$ и формула (6) упрощается:

$$D(S_j)=S_j-S_j^2. \tag{7}$$

С помощью результатов теории рекуррентных событий (в частности, уравнений восстановления) можно получить другую формулу для расчета дисперсии частоты. Ниже будем рассматривать невозвратные состояния (по классификации [6, стр. 380]), вероятность возвращения в которые отлична от нуля. В противном случае $R_j=0$ и приходим к формуле (7).

Можно считать, что любое невозвратное состояние определяет некоторое рекуррентное событие. Переход системы в состояние X_j определяет событие E_j. Основная идея перехода к рекуррентным событиям состоит в том, что некоторые параметры процесса, например число возвращений в состояние X_j, могут быть найдены путем анализа ряда независимых событий, хотя в целом процесс состоит из ряда зависимых испытаний.

Действительно, если система, отправляясь из состояния X_j, вновь попадает в состояние X_j, то процесс начинается как бы заново, и никакой зависимости от предыстории нет. С каждым рекуррентным событием можно связать две последовательности чисел, определенные при $h=1, 2, \ldots$

$b_j^{(h)}=$ Вер $\{E_j$ впервые наступило на h-м шаге при начальном состоянии $X_i\}$.

$f_j^{(h)}=$ Вер $\{E_j$ впервые наступило на h-м шаге при начальном состоянии $X_j\}$.

События «E_j впервые наступило на h-м шаге при $h=1, 2, \ldots$» несовместимы, поэтому

$$f_j=\sum_{h=1}^{\infty} f_j^{(h)}<1 \tag{8}$$

и

$$B_j=\sum_{h=1}^{\infty} b_j^{(h)}<1, \tag{9}$$

где f_j можно трактовать как вероятность возвращения в состояние X_j при выходе из состояния X_j, а B_j есть вероятность того, что, исходя из X_i, система когда-либо достигнет X_j. Так как мы рассматриваем марковские цепи с поглощением, соотношение (8) переходит в строгое неравенство. В [6] доказана теорема о том, что для невозвратных событий E_j (т. е. для которых $f_j<1$) имеет место соотношение

$$S_j=B_j(1-f_j)^{-1}. \tag{10}$$

Остается связать между собой f_j и R_j. Величина R_j есть среднее число возвратов в состояние X_j. При этом число возвратов ξ_j для рекуррентного события E_j имеет геометрическое распределение

$$\text{Вер}\{\xi_j=k\}=f_j^k(1-f_j). \tag{11}$$

Отсюда связь устанавливается формулой

$$R_j=M\{\xi_j\}=f_j(1-f_j)^{-1}. \tag{12}$$

Введенные нами параметры B_j, R_j и f_j позволяют для каждого состояния нарисовать упрощенный граф (рис. 1), из которого определяются средняя частота и дисперсия частоты для данного блока. Все ветви, показанные на рис. 1, в общем случае представляют собой совокупность

путей из одной вершины в другую. Система, отправляясь из состояния X_i, попадает с вероятностью B_j в состояние X_j, либо минует его с вероятностью $(1-B_j)$, попадая в поглощающее состояние. Попав в состояние X_j, система либо с вероятностью $(1-f_j)$ переходит в поглощающее состояние, либо с вероятностью f_j через некоторое число шагов возвращается в X_j и т. д. Как следует из формул (10), (12) и (6), для подсче-

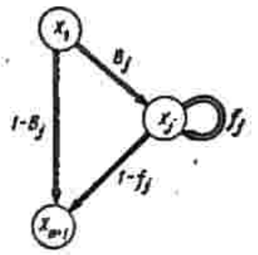


Рис. 1. Упрощенный граф для состояния X_i.

та дисперсии частоты достаточно знания двух параметров из трех: S_j, B_j, f_j (либо R_j).

Перейдем к расчету дисперсии времени исполнения всего алгоритма:

$$D(T)=M\{T^2\}-T^2=\sum_{i=1}^{N} c_i^2 M\{S_i^2\}+$$
$$+2\sum_{i=1}^{N-1}\sum_{j>i} c_i c_j M\{S_i S_j\}-T^2. \tag{13}$$

Распределение для S_i имеет вид:

$$\text{Вер}\{S_i=k\}=B_i(1-f_i)f_i^{k-1},\ k=1, 2, \ldots \tag{14}$$

Поэтому первая сумма из (13) может быть преобразована:

$$\sum_{i=1}^{N} c_i^2 M\{S_i^2\}=\sum_{i=1}^{N} c_i^2 B_i(1-f_i)\left(\sum_{k=1}^{\infty} k^2 f_i^{k-1}\right)=$$
$$=\sum_{i=1}^{N} c_i^2 S_i\frac{1+f_i}{1-f_i}; \tag{15}$$

Для преобразования второй суммы из (13) рассмотрим $M\{S_iS_j\}$:

$$M\{S_iS_j\}=M\left\{\left(\sum_{r=1}^{\infty} s_r^i\right)\left(\sum_{r'=1}^{\infty} s_{r'}^j\right)\right\}=\sum_{r,r'} M\{s_r^i s_{r'}^j\}=$$
$$=\sum_{r,r'>r} M\{s_r^i s_{r'}^j\}+\sum_{r,r'>r'} M\{s_r^i s_{r'}^j\}. \tag{16}$$

Последние преобразования сделаны с учетом того, что при $i\neq j$ и $r=r'$ $M\{s_r^i s_{r'}^j\}=0$, так как система на данном шаге не может находиться в двух состояниях одновременно. Затем каждая сумма из (16)

преобразуется по одному и тому же правилу. Например,

$$\sum_{\substack{r,\,r'\\ r'>r}} M\{a_r^l a_{r'}^l\} = \sum_{r=1}^{\infty} p_{ii}^{(r)} \left(\sum_{r'=r+1}^{\infty} p_{ij}^{(r'-r)} \right) =$$

$$= \sum_{r''=1}^{\infty} p_{ij}^{(r'')} \sum_{r=1}^{\infty} p_{ii}^{(r)} = q_{ij} S_i, \tag{17}$$

где $q_{ij} = \sum_{r=1}^{\infty} p_{ij}^{(r)}$.

Величина q_{ij} будет являться средней частотой попадания в состояние X_j, если X_i является начальным состоянием.

Таким образом,

$$M\{S_i S_j\} = q_{ij} S_i + q_{ji} S_j, \quad i \neq j. \tag{18}$$

Подставляя (18) и (15) в (13), получаем

$$D(T) = \sum_{i=1}^{N} c_i^2 S_i \frac{1+f_i}{1-f_i} + 2 \sum_{i=1}^{N-1} \sum_{j>i} c_i c_j (q_{ij} S_i + q_{ji} S_j) - \bar{T}^2. \tag{19}$$

Правая часть из (19) может быть записана более компактно. Действительно, если необходимо подсчитать $D(T)$ для многих вычислительных устройств, которые характеризуются различными c_i, целесообразно вести расчеты в матричной форме. Введем матрицу Q, элементы которой q_{ij} при $i \neq j$ определены в (17), а $q_{ii} = (1+f_i)/2(1-f_i)$. Теперь можно переписать (19) в следующем виде:

$$D(T) = 2\mathbf{c}Q\tilde{\mathbf{c}} - \bar{T}^2, \tag{20}$$

где

$$\mathbf{c} = \{c_i\}, \text{ а } \tilde{\mathbf{c}} = \{c_i S_i\}.$$

Как видно из (20), для конкретной граф-модели алгоритма достаточно один раз рассчитать элементы Q, а затем их использовать многократно с заданным набором векторов $\mathbf{c}$ и $\tilde{\mathbf{c}}$.

В случае одноразового подсчета $D(T)$ может оказаться предпочтительнее другая форма записи. Для этого обозначим через $\bar{t}_i$ среднее время исполнения алгоритма, если в качестве начального принят блок X_i. В частности $\bar{t}_1 = \bar{T}$. Опуская промежуточные выкладки, связанные с преобразованием (19), получаем

$$D(T) = 2 \sum_{i=1}^{N} \left(\bar{t}_i - \frac{c_i}{2} \right) c_i S_i - \bar{T}^2. \tag{21}$$

Таким образом, нами получены все необходимые формулы для решения поставленных задач, однако применение стандартных приемов для решения задач большой размерности приводит к известным трудностям. Действительно, если для определения средних частот исполнения блоков программы достаточно решить одну систему линейных уравнений (4) N-го порядка, то для расчета дисперсии времени исполнения алгоритма необходимо уже N раз решать систему (4), поочередно принимая за начальные все блоки алгоритма. Именно поэтому в последующих разделах основное внимание уделяется разработке специальных приемов, позволяющих сократить объем вычислений.

2. МЕТОД ДЕКОМПОЗИЦИИ БОЛЬШИХ ГРАФОВ

Обычно граф-модели алгоритмов характеризуются слабой связностью, т. е. число ветвей графа мало по сравнению с N^2, где N — число вершин графа. Последнее обстоятельство позволяет сократить объем необходимых расчетов. В предыдущем параграфе для каждого состояния марковской цепи были введены параметры B_i, R_i, f_i, S_i. На примере нескольких типичных фрагментов граф-схем мы покажем, как можно рассчитывать параметры отдельных состояний на основе параметров соседних состояний. Так, для фрагмента № 1 (рис. 2), если определены

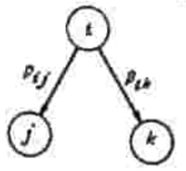


Рис. 2. Фрагмент № 1.

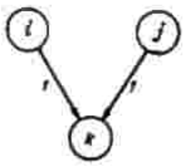


Рис. 3. Фрагмент № 2.

S_i, R_i и R_j, то S_j, S_k и R_k можно найти, воспользовавшись следующими соотношениями:

$$R_i = \sum_{h=1}^{\infty} p_{ii}^{(h)} = \sum_{h=1}^{\infty} p_{ij} p^{(h-1)} + \sum_{h=1}^{\infty} p_{ik} p_{kk}^{(h-1)} =$$

$$= \sum_{h=1}^{\infty} p_{ii}^{(h)} + \sum_{h=1}^{\infty} p_{kk}^{(h)} = R_j + R_k, \tag{22}$$

$$S_j = S_i p_{ij}, \; S_k = S_i p_{ik}.$$

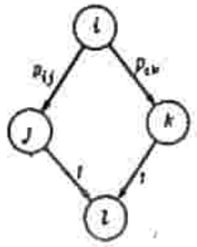


Рис. 4. Фрагмент № 3.

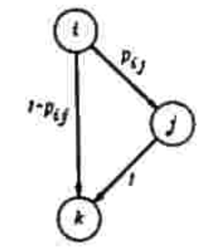


Рис. 5. Фрагмент № 4.

Не повторяя преобразований, аналогичных (22), сразу запишем соответствующие соотношения для остальных фрагментов (рис. 3, 4, 5).

Фрагмент № 2	Фрагмент № 3	Фрагмент № 4
$R_k = R_i + R_j,$	$S_i = S_i, \quad R_i = R_i,$	$S_k = S_i,$
$S_k = S_i + S_j,$	$R_k = R_i p_{ik},$	$R_i = R_k,$
	$R_j = R_i p_{ij}.$	$R_j = R_i p_{ij}.$

(23)

Аналогичные упрощения могут быть сделаны и при расчетах $\bar{t}_i$ и q_{ij}.

Более мощным средством сокращения расчетов является метод разложения исходного графа на подграфы. На исходной граф-модели выделим такие подмножества вершин, которые имеют одну входную и одну выходную вершины, и будем называть их подпрограммами. Пример такой подпрограммы изображен на рис. 6. Метод декомпозиции

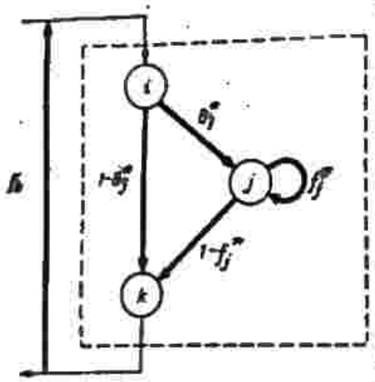


Рис. 6. К методу декомпозиции граф-схем.

заключается в том, что расчет параметров состояний проводится в два этапа. На первом этапе проводится расчет укрупненной схемы. На этом этапе подпрограмма представлена только одной своей вершиной — начальной. Предположим, на первом этапе определяются величины S_i и R_i. На втором этапе каждая подпрограмма рассчитывается отдельно. Вершина X_i выступает в роли начального состояния, а вершина X_k принимается за поглощающее состояние. Пусть для некоторого внутреннего оператора подпрограммы получены на втором этапе такие результаты: S^*_j, R^*_j (результаты второго этапа будем отмечать звездочкой). Покажем, как вычислить параметры S_j и R_j. Как и на рис. 1, ветви, показанные на рис. 6, представляют в общем случае некоторую совокупность путей из одной вершины в другую. Так, например, B^*_{ij} — вероятность попадания в состояние X_j при движении от начала подпрограммы при условии невыхода из подпрограммы.

По аналогии графа рис. 6 с фрагментом № 4 сразу запишем, что $S_i=S_k$ и $R_k=R_i$. Для расчета средних часто результат очевиден:

$$S_j=S_iS^*_j. \tag{24}$$

Подсчитаем вероятность невозвращения в состояние X_j при выходе из состояния X_j, рассматривая всю программу целиком:

$$(1-f_j)=(1-f^*_j)\sum_{h=0}^{\infty}(1-f_h)f_h^h(1-B^*_j)^h=\frac{(1-f^*_j)(1-f_h)}{1-f_h(1-B^*_j)}. \tag{25}$$

Отсюда

$$f_j=1-\frac{(1-f^*_j)(1-f_h)}{1-f_h(1-B^*_j)}. \tag{26}$$

Подставляя (26) в (12) и опуская промежуточные выкладки, получаем

$$R_j=R^*_j+R_iS^*_j. \tag{27}$$

Итак, мы показали, что расчет исходной граф-модели может быть проведен путем расчета нескольких графов меньшей размерности.

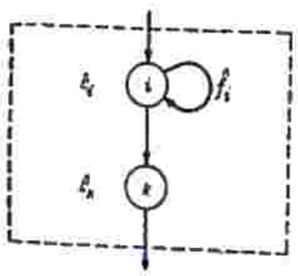


Рис. 7. Пример эквивалентной схемы для подпрограмм.

Необходимо остановиться на применении метода декомпозиции для расчета дисперсии времени исполнения всего алгоритма. Этот метод может эффективно применяться для расчета элементов матрицы Q. К сожалению, расчетные формулы разд. 1 неприменимы для простой схемы расчета, когда сначала рассчитываются отдельно подпрограммы, а затем укрупненная граф-модель, так как время исполнения подпрограммы становится уже случайной величиной. Возможны различные приближенные методы для вышеуказанной схемы расчета. Например, после расчета подпрограммы в дальнейших расчетах она представляется

эквивалентной схемой (рис. 7), где $\hat{c}_i$, $\hat{c}_k$ и $\hat{\tau}_i$ подбираются так, чтобы сохранилось среднее значение и дисперсия времени исполнения подпрограммы. В этом случае $\hat{c}_i$ и $\hat{c}_k$ — детерминированные величины, поэтому применимы формулы (20) или (21). Однако методика подбора эквивалентных схем и оценка точности предложенной аппроксимации выходят за рамки настоящей статьи.

3. ОПИСАНИЕ АЛГОРИТМА РАСЧЕТОВ

Циклическая структура граф-модели не позволяет, двигаясь по графу от начальной вершины, последовательно вычислять средние частоты исполнения каждого блока алгоритма по формулам (4). Можно воспользоваться некоторыми правилами преобразования графов [7], приводящими к эквивалентным в смысле сохранения значений S_i графам. Пример такого преобразования изображен на рис. 8. Коэффициенты передачи исходного графа (рис. 8,*а*) равны соответствующим переходным вероятностям p_{ij}. Выбор X_1 в качестве начального состояния приводит к необходимости «подключить» к вершине X_1 источник единичной интенсивности. Величина сигналов, установившихся в узлах граф-модели, соответствует средним частотам исполнения блоков алгоритма, благодаря адекватности граф-модели рис. 8,*а* и системы линейных уравнений (4). Как и в теории линейных графов сигналов [7], под передачей A_{ij} от узла X_i к узлу X_j будем понимать величину сигнала в узле X_j, появляющегося в результате внешнего единичного сигнала, приложенного к узлу X_i. Удаление ветви u_{62} исходного графа (рис. 8,*а*) приводит к появлению собственной петли у вершины X_2 и новой ветви u_{32}. Коэффициенты передач новых ветвей рассчитываются по формулам:

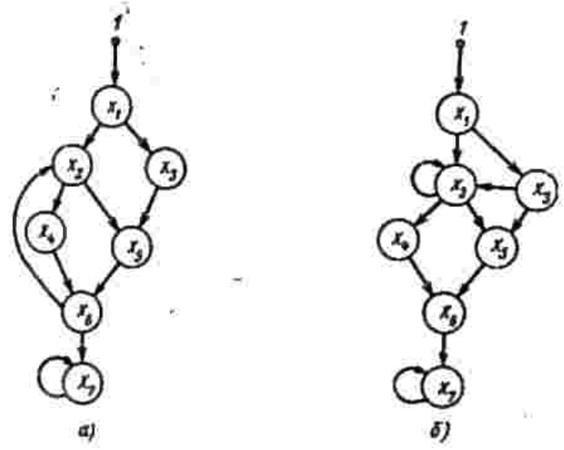


Рис. 8. Исходный (*а*) и преобразованный (*б*) графы.

$$p_{22}=A_{32}p_{62}, \tag{28}$$
$$p_{22}=p_{23}A_{52}p_{62}.$$

Преобразованный граф на рис. 8,*б* допускает последовательное вычисление всех S_i. Однако следует заметить, что подобные преобразования графов, совершенно элементарные с точки зрения ручной обработки, нелегко реализовать в виде алгоритма для автоматической обработки граф-схем большой размерности. Подобное положение вещей довольно типично для операций над графами.

С целью построения простого алгоритма введем между вершинами графа некоторое отношение, которое мы будем называть отношением доминирования. Если X_i доминирует над вершиной X_j, то будем назы-

вать X_i доминантой X_j, а X_j — субдоминантой X_i. Отношение доминирования определим индуктивно: X_i является доминантой X_j, если:

1) ранее между этими вершинами не было введено отношение доминирования и

2) из X_i можно достичь X_j непосредственно или проходя только через субдоминанты X_j.

Отношение доминирования описывается доминантной матрицей $D=\{d_{ij}\}$, у которой $d_{ij}=1$, если X_i является доминантой X_j, и $d_{ij}=0$ в противном случае. Для определенности будем полагать, что X_i всегда является доминантой самой себя, т. е. $d_{ii}=1$.

Как следует из определения, введенное отношение обладает свойствами транзитивности и антисимметричности [8]. Для графа, не имеющего циклов, отношение доминирования совпадает с понятием достижимости вершин, которое вычисляется путем построения транзитивного замыкания графа [9]. Действительно, если в ориентированном графе без циклов X_j достижимо из X_i, то обратного быть не может и, следовательно, X_i является доминантой X_j. Отсюда следует вывод, что доминантная матрица является матрицей, описывающей транзитивное замыкание некоторого графа без циклов. Другими словами, построение доминантной матрицы на основе исходного графа дает нам возможность преобразовать исходный граф в граф без циклов (кроме, быть может, собственных петель у вершин).

В качестве примера представлена доминантная матрица, построенная для графа рис. 8,*а*:

$$D=\begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}.$$

Вообще говоря, одному исходному графу может соответствовать несколько различных доминантных матриц. Это объясняется тем, что каждая из матриц соответствует определенному преобразованию исходного графа в граф без циклов, а такое преобразование в общем случае не единственно. Алгоритм построения доминантной матрицы на основе матрицы смежности U и алгоритм преобразования исходного графа приведены в приложении. Здесь мы только перечислим основные этапы алгоритма преобразования.

Шаг 1. Строится доминантная матрица D.

Шаг 2. Определяются ветви, подлежащие размыканию.

Шаг 3. Для каждой ветви, определенной на шаге 2, находится подмножество вершин, дающих новые ветви графа.

Шаг 4. По формулам, аналогичным (28), после расчета соответствующих передач вычисляются коэффициенты передач петель и новых ветвей.

После преобразования графа производится расчет средних частот исполнения каждого блока последовательным решением уравнений системы (4) и подсчитывается среднее время исполнения всего алгоритма.

Как указывалось выше, параметры R_i и q_i можно определить, приняв состояние X_i за начальное. Следовательно, выбирая каждый раз за начальную очередную вершину графа и производя расчет средних частот исполнения блоков, мы получили бы возможность рассчитать все необходимые параметры. Такая процедура, конечно, возможна, однако расчеты можно провести более экономичным путем. Обратимся

к рис. 9,*а*, где в отличие от рис. 8,*а* сигнал единичной интенсивности приложен не к вершине X_1, а к X_4, т. е. X_4 принята за начальную вершину. Если за основу преобразования взять матрицу D, то полученный граф (рис. 9,*б*) будет отличаться от соответствующего графа на рис. 8,*б* отсутствием единичного сигнала в вершине X_1 и появлением новой ветви от единичного источника к вершине X_2. Коэффициент передачи новой ветви рассчитывается аналогично (28):

$$p_{02}=p_{01}A_{14}p_{42}=A_{14}p_{42},$$

где нулевым индексом обозначен единичный источник.

Таким образом, достаточно сделать только одно преобразование графа, и при перемещении источника от одной вершины к другой учитывать только появление новых ветвей, исходящих из источника. Остается добавить, что расчет средних частот графа рис. 8,*б* начинается не с начальной вершины X_4, а с вершины X_1.

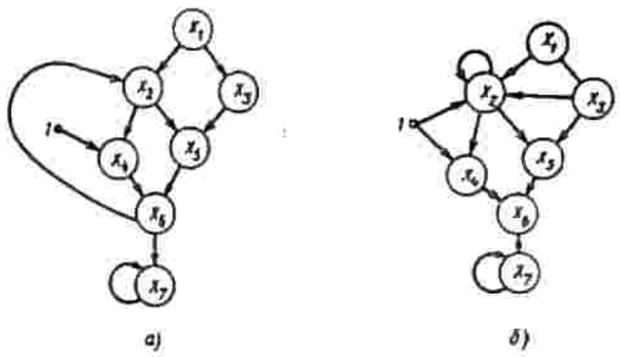


Рис. 9. Переход к новой начальной вершине.

Во втором разделе был описан метод декомпозиции сложных граф-схем. Основой его является разбиение исходной граф-модели на подграфы, названные нами подпрограммами. Доминантная матрица исходной граф-модели дает простой способ выявления всех подпрограмм исходной программы. Основная идея способа состоит в том, что если объединить множества доминант и субдоминант для начальной и конечной вершин подпрограммы, то полученные множества совпадают друг с другом. Отсюда становится ясным алгоритм выявления всех подпрограмм исходной программы.

Шаг 1. Строится матрица $W=D\vee D^T$, где D^T обозначает матрицу, полученную в результате транспонирования D.

Шаг 2. Из матрицы W выбираются совокупности одинаковых строк.

Шаг 3. В каждой из выбранных совокупностей строк определяются индексы начальной и конечной вершин подпрограммы. Начальной вершиной является вершина, которая будет доминантой всех остальных вершин в данной совокупности (определяется по D), а конечной вершиной является вершина, которая будет субдоминантой всех остальных вершин (определяется по D^T).

Шаг 4. Определяем состав каждой подпрограммы. Если X_i — начальная вершина, а X_k — конечная, то индексы блоков, входящих в подпрограмму, определяются единицами в вектор-строке $\mathbf{v}$.

$$\mathbf{v}=d_i\wedge d_k^T,$$

где d_i и d_k^T — i-я и k-я строки матриц D и D^T соответственно.

Шаг 5. Так как состав подпрограмм определяется по преобразованному графу, то необходимо проверить, не нарушается ли структура подпрограммы при учете «оборванных» ветвей. Структура подпрограммы будет нарушаться в том случае (т. е. она не является подпрограммой в принятом смысле), если одна из вершин, инцидентных «оборванной» ветви, входит в подпрограмму, а другая не входит.

В данном разделе мы остановились только на вопросах, вызывающих принципиальную трудность в расчетах, полагая, что составление полного алгоритма расчетов не вызовет особых затруднений

Заключение. В данной работе были описаны методы расчета основных параметров граф-моделей машинных программ: частоты исполнения отдельных блоков программы и времени исполнения всей программы. Анализ производился в предположении, что исходные параметры модели (время исполнения операторов, переходные вероятности) заданы. В действительности, подобные априорные сведения могут содержать элемент произвола, кроме того, марковская модель программ является лишь некоторым приближением к реальности. В силу этого автор полагает, что дальнейшие работы должны развиваться по двум направлениям. Во-первых, необходимо разработать методику определения исходных параметров моделей программ. При этом функционирование модели должно зависеть от качества заданных исходных данных. Другими словами, следует избегать переусложнения модели, когда высокая сложность модели не отвечает низкой точности исходных данных. Второе направление работ связано с построением других классов моделей, более точно соответствующих реальным процессам при исполнении программ на ЦВМ.

Приложение

Дадим описание вычисления доминантной матрицы D на основе матрицы смежности U. Матрица D строится построчно. Введем вектор $\beta=\{\beta_i\}$, который указывает на то обстоятельство, что i-я строка уже построена, если $\beta_i=1$. Покажем, как построить j-ю строку D, которую мы будем обозначать через d_j.

Среди множества вершин X_i, для которых $\beta_i=1$, выделим те, которые являются доминантами вершины X_j. Обозначим полученное множество через Z_j^-. У вершин, входящих в Z_j^-, удалим входящие ветви. Эта операция соответствует обнулению i-х столбцов матрицы U, если $X_i\in Z_j^-$. Полученная таким образом U_j^- выступает в качестве матрицы смежности при построении вектора достижимости [5] для вершины X_j. При этом могут быть использованы известные алгоритмы [5, 9] построения транзитивного замыкания графа. Построенный вектор достижимости и является строкой d_j. Последовательность построения векторов такова. Сначала строится d_1, затем рассматриваются в любой последовательности вершины, которые могут быть достигнуты из X_1 за один шаг, и для них вычисляются строки матрицы D. Затем переходим к вершинам, которые могут быть достигнуты из X_1 за два шага, и т. д. Некоторый произвол в выборе вершин может приводить к построению различных матриц D, но для нашей задачи это не имеет особого значения, так как любая матрица D обеспечивает необходимое преобразование графа. Как легко убедиться, процедура построения матрицы D соответствует данным выше определениям доминантной матрицы.

Для определения дуг, подлежащих удалению, строим матрицу $M=\{m_{ij}\}$, элементы которой вычисляются по формуле

$$m_{ij}=u_{ij}\wedge\bar{d}_{ij}. \qquad (\text{П.1})$$

Если $m_{ij}=1$, то дуга u_{ij} подлежит удалению. Удалив из графа G все дуги u_{ij}, для которых $m_{ij}=1$, получим редуцированный граф G^*. Каждой удаляемой ветви u_{ij} поставим в соответствие множество существенных вершин $\Phi[i, j]$, которое образует существенный подграф дуги. Вершина X_k называется существенной относительно дуги u_{ij}, если:

1) существует путь в G^* из X_j в X_k;

2) существует путь G^* из X_k в X_i.

Вершину X_i также будем относить к существенным вершинам.

Ветви u_{lk}, для которых X_k — существенная вершина, а X_l не является существенной вершиной, назовем входными ветвями существенного подграфа, а X_l будем именовать входом существенного подграфа. Так, для ветви u_{62} (см. рис. 8,*а*) существенными вершинами являются X_6, X_3 и X_2, а u_{35} является входной ветвью существенного подграфа ветви u_{62}. Как показано на рис. 8,*б*, размыкание ветви u_{62} компенсируется петлей при вершине X_3 с передачей $p_{62}\Lambda_{35}$ и для входной ветви u_{35} строится новая ветвь u_{32} с передачей $p_{35}\Lambda_{35}p_{62}$. Передачи Λ_{35} и Λ_{36} вычисляются только на путях, соединяющих существенные вершины.

Для каждой удаляемой ветви определяем существенный подграф, вершинам которого соответствуют ненулевые элементы вектора $\varphi[i, j]$:

$$\varphi[i, j]=d_i\wedge d_j^T, \qquad (\text{П.2})$$

что непосредственно следует из данного выше определения и из того обстоятельства, что D соответствует транзитивному замыканию графа G, у которого удалены «обратные» ветви. Для определения входов в существенный подграф рассуждаем следующим образом. Инвертируем ветви графа G. Как известно [7], этой операции соответствует транспонирование U. Множество вершин, куда можно попасть за один шаг, двигаясь по инвертированному графу из существенных вершин, обозначим $\Phi_1[i, j]$. Тогда множество входов $\Phi_2[i, j]$ существенного подграфа определяется следующим образом:

$$\Phi_2[i, j]=\Phi_1[i, j]\setminus\Phi[i, j]\setminus X_j. \qquad (\text{П.3})$$

От операций с множествами (П.3) перейдем к операциям с булевыми векторами. Входам существенного подграфа соответствуют ненулевые элементы вектора $\varphi_2[i, j]$:

$$\varphi_2[i, j]=\bigvee_{X_k\in\Phi^*[i, j]}u_k^T\wedge\overline{(\varphi[i, j]\vee\delta[i])}, \qquad (\text{П.4})$$

где через $\delta[i]$ обозначен вектор, i-я составляющая которого равна 1, а все остальные — нулю.

Порядок удаления ветвей из графа устанавливается с помощью матрицы D. Желательным является такой порядок удаления ветвей, при котором существенный подграф удаляемой ветви является графом без циклов (допускаются лишь собственные петли у вершин), что обеспечивает простой расчет передач внутри существенного подграфа.

Исходя из этого, устанавливаем следующее правило предпочтения: ветвь u_{ij} удаляется из графа раньше, чем u_{pq}; если X_j является субдоминантой X_p.

ЛИТЕРАТУРА

1. Král I. One way of estimating frequencies of jumps in a program. Comm. ACM, 1968, v. 11, № 7.
2. Ramamoorthy C. V. Discrete system representation and analysis by generating functions of abstract graphs. IEEE Int. Conv. Rec. Pt. 6, 1965.

3. Martin D., Estrin G. Models of computational systems. IEEE Trans. EC, 1967, v. 16, Febr.
4. Martin D., Estrin G. Evaluation of vertex probabilities in graph models of computations J. of. ACM, 1967, v. 14, № 2.
5. Ramamoorthy C. V. Analysis of graphs by connectivity considerations J. of ACM, 1966, v. 13, № 2.
6. Феллер В. Введение в теорию вероятностей и ее приложения, т. 1. Изд-во «Мир», 1964.
7. Мэзон С., Циммерман Г. Электронные цепи, сигналы и системы. Изд-во иностранной литературы, 1963.
8. Оре О. Теория графов. Изд-во иностранной литературы, 1968.
9. Мартынюк В. В. Экономное построение транзитивного замыкания бинарного отношения. ЖВМ и МФ, 1962, № 4.

УДК 681.3.04—033

ПРОБЛЕМНО-ОРИЕНТИРОВАННЫЙ ЯЗЫК И СИСТЕМА ГЕНЕРИРОВАНИЯ ПРОГРАММ ДЛЯ ЗАДАЧ ОБРАБОТКИ ДАННЫХ

В. А. ПАВЕЛЬЕВ, Д. А. СТЕПАНЧЕНКО

Расширение сферы применения вычислительных машин приводит к созданию языков и систем программирования, предназначенных для решения отдельных классов задач. Такие языки и системы программирования часто называют проблемно-ориентированными [1].

К числу проблемно-ориентированных языков можно отнести RPG (Report Program Generator) *), предназначенный для класса задач обработки данных, требующих обработки больших объемов информации и представления результатов в виде печатных документов (всевозможных отчетов, справок, таблиц) [2].

Первый вариант языка был разработан фирмой IBM для серии малых вычислительных машин IBM-1401 и получил широкое распространение среди непрофессиональных программистов для решения экономических задач. Дальнейшее развитие этот язык получил при разработке математического обеспечения для ЦВМ третьего поколения IBM/360. В настоящее время он реализован на многих ЦВМ других типов (Univac 9000, RCA Spectra, ICL System/4). Усовершенствованный вариант языка RPG-II принят в качестве единственного языка программирования в фирменном математическом обеспечении вычислительных машин серии IBM System/3.

Система RPG предполагает определенную организацию процесса решения задачи и информации. Процесс решения задачи должен состоять из таких этапов, как выбор необходимой информации, ее обновление, формирование итоговых сведений разных уровней обобщения, оформление и выдача результатов в форме отчетов. При использовании RPG данные для обработки должны объединяться в комплекты данных, называемые фондами данных, или файлами. Фонды делятся на порции информации, называемые «записями».

Обычно фонды данных располагаются на внешних запоминающих устройствах ЦВМ и допускают переписывание в оперативную память (из их состава) только целых записей. Каждая запись в свою очередь состоит из элементов информации, называемых полями данных **). Исходная информация к задаче организована в виде так называемых входных фондов, соответственно выходная — в виде выходных фондов.

*) RPG читается «ар пи джи».

**) Обычно в состав записи включают поля данных, связанные между собой каким-то образом, например сведения об одном и том же объекте.

В качестве носителей информации (информационных хранилищ) могут использоваться хранилища как с последовательным доступом (магнитные ленты, перфокарты), так и с непосредственным доступом (магнитные диски).

Возможность использования таких типов оборудования для хранения информации приводит к целесообразности использования различных способов организации фондов данных для обеспечения более эффективных способов выбора информации. Просмотр записей (с целью выбора нужных) в хранилище с последовательным доступом производится в порядке их расположения в фонде, для выбора необходимой порции информации с диска должен быть задан ее начальный адрес на диске. В RPG задание адресов записей фонда с прямым доступом осуществляется с помощью так называемого фонда адресов записей. Использование внешней памяти с непосредственным доступом приводит к целесообразности использования в RPG и такой организации фондов данных, при которой записи одного фонда связаны адресами ассоциативных связей с записями других фондов, фонды, организованные таким образом, в RPG, называются Ц (цепочечно)-адресующими и Ц—адресуемыми соответственно.

Характерным для задач, удобно описываемых на RPG, является использование табличных данных, представляющих собой последовательности значений функции, или последовательности пар, образованных из значений аргументов и функций. Данные, представленные так, образуют табличный фонд. RPG рекомендуется для программирования задач обработки данных, характерной особенностью которых является процесс обновления, заключающийся в получении некоторого нового фонда данных на основании корректировки первичного фонда. Это осуществляется путем внесения в его информацию изменений, содержащихся в других фондах, называемых вторичными. Изменения сводятся к включению или исключению некоторых записей или модификации отдельных полей в записях. Для удобства описания этих действий некоторые поля данных объявляются сопоставимыми. При совпадении данных, находящихся в сопоставимых полях, осуществляется корректировка соответствующих записей основного фонда информацией соответствующих записей вторичного фонда.

Обновление фонда, расположенного на носителе с последовательным доступом, допускает в получаемом фонде появление записей, размеры полей которых изменены по сравнению с полями соответствующих записей первичного фонда. Обновление же фонда данных, расположенного на носителе с произвольным доступом, в аналогичном случае повлечет к необходимости корректировки соответствующего фонда начальных адресов.

В случае обновления фонда, расположенного на носителе с произвольным доступом, без изменения расположения полей записей, допускается получение результирующего фонда на месте первичного. Фонд данных в этом случае называется обновляемым.

Для более эффективного функционирования некоторых процессов обработки данных целесообразно использование фондов, записи которых упорядочены по заданному признаку; такие фонды называются упорядоченными. Так, например, в случае процесса обновления вторичные фонды данных, расположенные на носителях с последовательным доступом и участвующие в процессе, должны быть упорядочены в соответствии с порядком следования записей в основном фонде.

Характерные для задач обработки данных, описываемых с помощью RPG, является и то, что в процессе вычислений помимо частных результатов часто требуется получение и выдача итоговых, обобщающих данных разных уровней обобщения, а также заголовков. Это возможно благодаря наличию классификации исходных данных.

Формируемые и выдаваемые на печать отчеты и справки чаще всего выдаются в виде таблиц заранее заданной формы. Поэтому при выдаче результатов должно осуществляться их редактирование в соответствии с требованиями выходных форм. Язык RPG ориентирован на задачи класса обработки данных. В названии языка отражена основная цель задач из рассматриваемого класса — выдача печатных отчетов для справок. Результаты также могут быть оформлены в виде фондов и получены, например, на магнитной ленте с целью использования в других задачах.

RPG позволяет легко и быстро описывать задачи обработки данных благодаря наличию в нем ряда средств, предназначенных для таких задач. Можно выделить две основные особенности системы RPG. Первая заключается в том, что RPG-программа пишется на специальных бланках, называемых формами спецификации. Форма каждого типа используется для определенного типа информации, например описания фондов или описания вычислений. Заполнение форм происходит построчно. Каждая строка состоит из колонок, сгруппированных в графы в соответствии с типом описываемой в них информации, указанным в «шапке» формы. Такая форма записи RPG-программ позволяет записывать их коротко и просто. Например, вместо записи: FILE TYPE = = INPUT (тип файла = входной) программист должен в колонке с названием «Тип файла» поставить символ I.

Вторая особенность RPG основывается на использовании базовой блок-схемы при программировании задач рассматриваемого класса. Это позволяет некоторые части программы составлять автоматически генератором по некоторым унифицированным правилам. Например, команды ввода — вывода не существуют в языке RPG и формируются в стандартной форме самим генератором.

Благодаря такому принципу построения системы целый ряд аспектов программирования может быть легко описан в RPG. Некоторые из них указываются ниже.

1. RPG допускает удобное обращение с табличными данными, которое в процессе записи вычислений задается с помощью одной команды «Поиск в таблице».

2. Процесс обновления основного фонда очень просто реализуется в RPG с помощью метода сопоставления записей, принадлежащих двум фондам, один из которых основной, а второй содержит данные о внесении изменений (т. е. является вторичным).

3. Введение так называемых индикаторов уровней управления, отражающих структуру данных, позволяет легко получать итоговые данные при переходе от одного уровня данных к другому.

4. Для описания форматов выходных данных введены так называемые редактирующие слова, с помощью которых в выводимых на печать данных производится подавление незначащих нулей и включение различных пунктуационных символов.

Однако существует целый ряд приемов программирования, которые не могут быть описаны на языке RPG. К их числу можно отнести такие, как оперирование с двоичными числами, ввод и вывод через пишущую машинку, описание повторяющихся элементов данных и обработку их с помощью модификации адресов, управление устройствами ЦВМ (например, отмотка магнитной ленты). Однако есть возможность выполнения этих операций, для этого RPG-программа может вызвать необходимые программные модули, написанные на другом языке, например на автокоде.

Как уже говорилось, RPG-программа записывается на формах 6 типов, задавая тем самым соответствующие спецификации программы:

1) описания фондов,

2) детализации фондов,
3) счетчика строк,
4) входных данных,
5) вычислений,
6) выходных данных.

Не все спецификации должны быть обязательно заданы в RPG-программе. Спецификация детализации фондов задается только в случае присутствия в программе фондов данных специальных типов (например, табличного) для задания дополнительных характеристик. Спецификация счетчика строк задается только в том случае, когда требуется дополнительная информация о некоторых выдаваемых фондах.

Каждая из указанных спецификаций в записи RPG-программы содержит некоторые общие для всех типов форм характеристики.

На каждой строке форм указывается номер страницы и номер строки на странице *). Эта информация используется в процессе генерирования программы для проверки правильности следования перфокарт программы. Крайняя правая позиция в номере строки оставляется свободной при первоначальном написании программы для обеспечения возможности простого способа включения дополнительных строк в случае необходимых изменений без перенумерации всего массива.

Каждая строка формы содержит указание о типе спецификации из числа допустимых шести типов. Звездочка в определенной позиции строки говорит о том, что вся строка должна восприниматься как комментарий. На каждой строке отведено поле для идентификации программы, которой принадлежит данная строка спецификации или соответственно перфокарта. Эта информация игнорируется программой RPG.

Для записи RPG-программы используется алфавит, состоящий из букв, цифр и специальных знаков. Из них строятся некоторые элементарные конструкции языка: цифровые и алфавитно-цифровые литералы, константы, редактирующие слова, идентификаторы.

В качестве элементов спецификаций в некоторых из допустимых в языке форм задаются индикаторы. Индикаторы служат для управления последовательностью вычислений и контроля.

В языке допускаются индикаторы, выполняющие следующие функции: индикаторы останова, индикаторы способов управления уровнями вычислений, индикаторы состояния поля, индикаторы обрабатываемой записи, индикаторы сопоставимости записей, индикаторы результатов вычислений, индикаторы переполнений страниц при печати.

Спецификация описания фондов. Форма спецификации описания фондов используется для задания информации о тех фондах, которые должны обрабатываться генерируемой программой.

Эта информация включает такие характеристики, как название фонда, тип фонда, структура фонда, способ обработки. Устройства ввода — вывода, связанные с описываемым фондом, задаются названием устройства в соответствии с таблицей допустимых типов устройств.

В качестве типа фонда указывается один из трех возможных: входной, выходной или обновляемый (в последнем случае фонд является входным и выходным одновременно).

Если описываемый фонд является входным, то указывается его тип из числа допустимых для входных фондов: первичный, вторичный, фонд адресов записей, Ц-адресуемый, табличный.

Для входного фонда указывается также способ его упорядоченности (в возрастающей или убывающей последовательности) по некоторому признаку. Этот признак используется генерируемой программой для

*) Формы, используемые для записи RPG-программы, содержат восемьдесят колонок для удобства использования 80-колоночных перфокарт при перфорации.

контроля упорядоченности фонда: в случае обнаружения нарушения упорядоченности включается соответствующий индикатор и выполнение программы приостанавливается до тех пор, пока этот индикатор не будет выключен.

В спецификациях, задающих формат фонда, указывается, постоянную или переменную длину имеют записи фонда и задается длина записи в первом случае и максимально возможная длина — в последнем случае.

Если описываемый фонд является выходным фондом для печати, то указываются используемые индикаторы для указания переполнения страницы, относящиеся к данному фонду.

Дополнительно ряд спецификаций должен быть задан в случае, когда организация фонда допускает произвольное обращение. Задаваемый в этом случае способ обработки фонда может быть либо последовательным, либо выборочным. В последнем случае адрес записи получается либо с помощью фонда адресов записей, либо с помощью Ц-адресующего фонда.

Если описываемый фонд является фондом адресов записей, то указывается длина поля, отведенного для каждого адреса.

Если описываемый фонд является Ц-адресующим, табличным или фондом адресов записей, то должно быть указание на наличие дополнительной информации о нем в спецификации детализации фондов, а для выходного фонда в случае необходимости — в спецификации счетчика строк.

Спецификация детализации фондов. Форма детализации фондов используется для задания дополнительной информации о Ц-адресующих фондах, фондах адресов записей и табличных фондах, используемых в RPG-программе.

Для указанных фондов задаются их названия, а также названия соответствующих им фондов: а) при описании Ц-адресующего фонда задается название Ц-адресуемого фонда, б) при описании фонда адресов записей задается название фонда, содержащего эти записи; в) при описании табличного фонда задается название фонда, получаемого после обновления таблицы.

Для Ц-адресующего фонда задается номер поля цепной адресации, присвоенный ему в спецификации входных данных, а также указание об упорядоченности последовательности записей.

Для табличного фонда задается либо название таблицы, содержащей аргументы, либо название таблицы, содержащей значения функции, либо и то и другое, а также длины полей, содержащих соответствующие компоненты таблицы, и способ упорядочения последовательности соответствующих элементов таблиц и их количество в записи и в таблице. Если элементами таблиц являются числа, то специфицируется способ задания чисел.

Спецификация счетчика строк. С помощью спецификаций, задаваемых в этой форме, программист имеет возможность изменить носитель информации для выходных данных без изменения исходной RPG-программы. Такая необходимость может возникнуть, например, в случае, если результаты работы должны быть выданы на магнитную ленту вместо печати.

Данная спецификация используется для задания информации о тех строках, выдача которых зависит от специальной управляющей перфоленты и поэтому информация об этих строках без описания их в данной спецификации не будет передана на магнитную ленту.

Спецификация входных данных. Эта форма спецификации определяет характеристики входных данных для RPG.

Информация, заполняющая эту форму, идентифицирует описываемые записи и устанавливает связь этих записей с другими записями

в данном фонде, а также определяет поля входных записей, используемых в составляемом отчете.

Для каждого описываемого входного фонда задается его название, а также порядок следования в нем записей. Если записи должны следовать в фонде в определенном порядке, то должен быть задан желаемый порядок их следования, при этом указывается случай, когда запись данного типа не обязательно должна присутствовать или количество записей такого типа может быть произвольным. Нарушение указанного в описании следования записей приводит к останову при выполнении программы и включению соответствующего индикатора останова.

Для каждой записи может быть задан один или несколько идентифицирующих кодов, каждый из которых в записи задается позициями тех символов, которые служат для различения записей по их типам. Между кодами идентификации записи могут существовать отношения *и, или*. Для каждой записи специфицирован индикатор, включаемый каждый раз, когда запись данного типа обрабатывается. Этот индикатор может использоваться в других формах спецификаций.

Другая группа спецификаций этой формы относится к описанию полей записи. При этом задается название поля, местоположение его в записи, тип данных, содержащихся в поле (числовые или алфавитно-цифровые). Если описываемое поле является управляющим (определяющим уровень управления), то ему ставится в соответствие индикатор из числа индикаторов уровней управления. Указанный в этой спецификации индикатор уровня управления затем используется в спецификациях вычислений и выходных данных.

Специфицируются также такие характеристики, как сопоставление или адресация записей этого поля к записям другого фонда. В случае, если одно и то же поле занимает различное местоположение в различных записях, то отождествление таких полей осуществляется заданием соответствующего индикатора, специфицирующего поля записей, находящихся в отношении *или*.

Спецификация индикаторов состояния входного поля позволяет для каждой записи определить, положительным, равным нулю или отрицательным является соответствующее поле еще до начала обработки этой записи.

Если в процессе выполнения программы будет прочитана запись, не определенная на форме спецификации входных данных, то включается индикатор останова и выполнение программы приостанавливается.

Спецификация вычислений. Спецификации этой формы обеспечивают RPG информацией о том, когда, какого рода должны быть выполняемые вычисления и какие проверки должны быть выполнены над результатами этих вычислений. Операции должны быть перечислены в порядке их выполнения рабочей программой. При этом могут производиться либо частные вычисления, либо итоговые. Выполнение частных вычислений осуществляется для каждой очередной записи. Характер и выполнение итоговых вычислений специфицируется заданием индикатора соответствующего уровня управления. Изменение уровня управления при некотором конкретном уровне приводит к включению всех индикаторов уровней управления ниже данного.

Для каждой операции могут быть заданы также индикаторы, обусловливающие выполнение данной операции. Некоторое вычисление выполняется, если все указанные индикаторы находятся в требуемом состоянии.

Указанные индикаторы могут определять тип записи, для которой должно быть выполнено вычисление, или условия, которые возникли при предыдущих вычислениях либо заданы индикаторами уровней управления, либо индикаторами сопоставимых записей. Каждое вычисле-

контроля упорядоченности фонда: в случае обнаружения нарушения упорядоченности включается соответствующий индикатор и выполнение программы приостанавливается до тех пор, пока этот индикатор не будет выключен.

В спецификациях, задающих формат фонда, указывается, постоянную или переменную длину имеют записи фонда и задается длина записи в первом случае и максимально возможная длина — в последнем случае.

Если описываемый фонд является выходным фондом для печати, то указываются используемые индикаторы для указания переполнения страницы, относящиеся к данному фонду.

Дополнительно ряд спецификаций должен быть задан в случае, когда организация фонда допускает произвольное обращение. Задаваемый в этом случае способ обработки фонда может быть либо последовательным, либо выборочным. В последнем случае адрес записи получается либо с помощью фонда адресов записей, либо с помощью Ц-адресующего фонда.

Если описываемый фонд является фондом адресов записей, то указывается длина поля, отведенного для каждого адреса.

Если описываемый фонд является Ц-адресующим, табличным или фондом адресов записей, то должно быть указание на наличие дополнительной информации о нем в спецификации детализации фондов, а для выходного фонда в случае необходимости — в спецификации счетчика строк.

Спецификация детализации фондов. Форма детализации фондов используется для задания дополнительной информации о Ц-адресующих фондах, фондах адресов записей и табличных фондах, используемых в RPG-программе.

Для указанных фондов задаются их названия, а также названия соответствующих им фондов: а) при описании Ц-адресующего фонда задается название Ц-адресуемого фонда; б) при описании фонда адресов записей задается название фонда, содержащего эти записи; в) при описании табличного фонда задается название фонда, получаемого после обновления таблицы.

Для Ц-адресующего фонда задается номер поля цепной адресации, присвоенный ему в спецификации входных данных, а также указание об упорядоченности последовательности записей.

Для табличного фонда задается либо название таблицы, содержащей аргументы, либо название таблицы, содержащей значения функции, либо и то и другое, а также длины полей, содержащих соответствующие компоненты таблицы, и способ упорядочения последовательности соответствующих элементов таблиц и их количество в записи и в таблице. Если элементами таблиц являются числа, то специфицируется способ задания чисел.

Спецификация счетчика строк. С помощью спецификаций, задаваемых в этой форме, программист имеет возможность изменить носитель информации для выходных данных без изменения исходной RPG-программы. Такая необходимость может возникнуть, например, в случае, если результаты работы должны быть выданы на магнитную ленту вместо печати.

Данная спецификация используется для задания информации о тех строках, выдача которых зависит от специальной управляющей перфоленты и поэтому информация об этих строках без описания их в данной спецификации не будет передана на магнитную ленту.

Спецификация входных данных. Эта форма спецификации определяет характеристики входных данных для RPG.

Информация, заполняющая эту форму, идентифицирует описываемые записи и устанавливает связь этих записей с другими записями

в данном фонде, а также определяет поля входных записей, используемых в составляемом отчете.

Для каждого описываемого входного фонда задается его название, а также порядок следования в нем записей. Если записи должны следовать в фонде в определенном порядке, то должен быть задан желаемый порядок их следования, при этом указывается случай, когда запись данного типа не обязательно должна присутствовать или количество записей такого типа может быть произвольным. Нарушение указанного в описании следования записей приводит к останову при выполнении программы и включению соответствующего индикатора останова.

Для каждой записи может быть задан один или несколько идентифицирующих кодов, каждый из которых в записи задается позициями тех символов, которые служат для различения записей по их типам. Между кодами идентификации записи могут существовать отношения *и, или*. Для каждой записи специфицирован индикатор, включаемый каждый раз, когда запись данного типа обрабатывается. Этот индикатор может использоваться в других формах спецификаций.

Другая группа спецификаций этой формы относится к описанию полей записи. При этом задается название поля, местоположение его в записи, тип данных, содержащихся в поле (числовые или алфавитно-цифровые). Если описываемое поле является управляющим (определяющим уровень управления), то ему ставится в соответствие индикатор из числа индикаторов уровней управления. Указанный в этой спецификации индикатор уровня управления затем используется в спецификациях вычислений и выходных данных.

Специфицируются также такие характеристики, как сопоставление или адресация записей этого поля к записям другого фонда. В случае, если одно и то же поле занимает различное местоположение в различных записях, то отождествление таких полей осуществляется заданием соответствующего индикатора, специфицирующего поля записей, находящихся в отношении *или*.

Спецификация индикаторов состояния входного поля позволяет для каждой записи определить, положительным, равным нулю или отрицательным является соответствующее поле еще до начала обработки этой записи.

Если в процессе выполнения программы будет прочитана запись, не определенная на форме спецификации входных данных, то включается индикатор останова и выполнение программы приостанавливается.

Спецификация вычислений. Спецификации этой формы обеспечивают RPG информацией о том, когда, какого рода должны быть выполняемые вычисления и какие проверки должны быть выполнены над результатами этих вычислений. Операции должны быть перечислены в порядке их выполнения рабочей программой. При этом могут производиться либо частные вычисления, либо итоговые. Выполнение частных вычислений осуществляется для каждой очередной записи. Характер и выполнение итоговых вычислений специфицируется заданием индикатора соответствующего уровня управления. Изменение уровня управления при некотором конкретном уровне приводит к включению всех индикаторов уровней управления ниже данного.

Для каждой операции могут быть заданы также индикаторы, обусловливающие выполнение данной операции. Некоторое вычисление выполняется, если все указанные индикаторы находятся в требуемом состоянии.

Указанные индикаторы могут определять тип записи, для которой должно быть выполнено вычисление, или условия, которые возникли при предыдущих вычислениях либо заданы индикаторами уровней управления, либо индикаторами сопоставимых записей. Каждое вычисле-

ние может быть специфицировано заданием типа операции, двух полей операндов и поля результата.

В поле операнда могут быть представлены либо название некоторого поля данных, либо литерал. Спецификации поля результата, определяющие его длину, точность, с которой должен быть вычислен результат, необходимость округлений, задаются один раз, если одно и то же поле используется в нескольких вычислениях.

При использовании операций сравнения или поиска в таблице может быть специфицирован индикатор поля результата для управления выполнением последовательности операций.

В число операций, допустимых в языке RPG, входят операции следующих типов: арифметические, пересылки, проверки и сравнения, условного перехода и обращения к подпрограмме, включения и выключения индикаторов, работы с таблицами, операции программ преобразования адресов для работы с ЗУ с произвольным обращением.

Спецификация выходных данных. Эта спецификация заполняется информацией о выходных фондах, состоящей из описания их записей и полей.

Для каждой описываемой записи фонда, задаваемого своим названием, специфицируется тип записи. Записи могут быть заголовками, частными или итоговыми результатами. Содержимое этих записей может представлять собой константы, данные входных записей или результаты вычислений. Спецификации записей позволяют указать необходимое число пустых строк, которые должны быть выданы *до*, а также *после* выдачи описываемой записи. Задаваемые в спецификациях этой формы индикаторы служат для управления выдачей записей и формированием полей. Индикаторы могут находиться в отношении *и*, а также *или*. Каждое поле, описываемое в данной форме, задается либо своим названием, либо константой. Имеется возможность ведения автоматической нумерации страниц путем задания в качестве названия поля идентификатора PAGE (страница).

Выдаваемые на печать поля могут быть отредактированы с помощью редактирующего слова, обеспечивающего подавление незначащих нулей, проставление при печати необходимых символов, знаков чисел и т. д.

Перевод RPG-программ на машинный язык осуществляется с помощью транслятора компилирующего типа. Все рабочие программы, полученные с помощью этого компилятора, имеют базовую блок-схему, которая отвечает основным требованиям программ для решения задач определенного класса систем обработки данных. Благодаря указанной особенности компиляторы подобного типа называют иногда генераторами.

Полный процесс компиляции RPG-программы можно разбить на несколько последовательных этапов, каждому из которых соответствует свой блок компилятора. Переходом от одного этапа к другому управляет координирующий блок, постоянно находящийся в оперативном запоминающем устройстве ВМ, в то время как остальные блоки компилятора вызываются по мере необходимости.

Блок синтаксического контроля проверяет правильность синтаксиса конструкций исходной RPG-программы, строит многочисленные таблицы, используемые на других этапах компиляции, и записывает исходную программу, а также сведения о выявленных ошибках в некоторый рабочий фонд для использования этих данных на последнем этапе компиляции.

Блок обработки таблиц осуществляет обработку и связывание таблиц, полученных на первом этапе. Блок генерации машинных команд создает рабочие модули в машинном коде в соответствии с таблицами, построенными на первом и втором этапах, и с учетом той базовой

блок-схемы, которая является общей для всех генерируемых RPG-программ. Блок окончательной обработки завершает формирование рабочей программы и производит печатный документ, так называемый «Листинг», в котором представлены как текст исходной программы, так и некоторые сгенерированные машинные команды, относящиеся к спецификациям вычислений, а также выявленные ошибки.

Анализ рассматриваемого языка показывает, что количество различных характеристик, необходимых для спецификации RPG-программы, значительно увеличивается при обработке фондов сложных структур, возникающих, как правило, при использовании ЗУ с непосредственным доступом, например магнитных дисков. Кроме того, использование RPG непрофессиональными программистами требует от них знакомства с некоторыми характеристиками машинных языков программирования, например упакованной формой представления чисел или сблокированными записями на магнитной ленте.

Однако система программирования RPG оказывается весьма эффективной для рассмотренного класса задач обработки данных, при этом сравнительный анализ записей алгоритма задачи данного класса на языках КОБОЛ и RPG показывает, что размеры и время составления исходной RPG-программы существенно меньше таковых для программы той же задачи, составленной на языке КОБОЛ [3].

ЛИТЕРАТУРА

1. «Современное программирование». Сб. статей. Перев. с англ., под ред. Задыхайло И. Б., вып. 2. Изд-во «Советское радио», 1967.
2. RPG Reference Manual. International Computers Ltd, 1967.
3. Leslie H. The Report Program Generator. Datamation, 1967, № 6, p. 26—28.

УДК 681.3.06

ОБ ИСПОЛЬЗОВАНИИ ГРАММАТИЧЕСКИХ СРЕДСТВ В ИПС ДЛЯ БОЛЬШИХ МАССИВОВ ДОКУМЕНТОВ

А. И. КИТОВ, Е. К. ГРАЧЕВА

Информационно-поисковые системы (ИПС), предназначенные для хранения и оперативного поиска документов в информационном фонде порядка 1 млн. документов, с принципом построения по автономным специализированным направлениям (подмассивам), ориентированным на определенный круг потребителей и источников информации, наряду с общими выдвигают ряд дополнительных требований к информационному языку системы (ИЯ), предназначенному для описания документов и запросов [1]. ИЯ системы должен обеспечивать:

— необходимую полноту и точность описания содержания документов (или запросов);

— достаточно высокую эффективность разделения документов (запросов) по тематическим подмассивам;

— высокие показатели полноты и точности выдачи ответов на запросы;

— возможность автоматического аннотирования документов, выдаваемых в ответ на запрос.

Одним из возможных способов достижения вышеперечисленных требований является введение в состав информационного языка специальных грамматических средств.

В качестве грамматических средств предлагается использовать систему смысловых связей или логико-семантических отношений (ЛСО) [2].

ЛСО включают:

— указание направлений связей между дескрипторами поискового образа;

— указание видов отношений между отдельными дескрипторами внутри поискового образа.

Состав ЛСО в ИПС для больших массивов документов должен быть не только достаточно полным для обеспечения вышеперечисленных требований, но и существенно ограниченным из-за требований минимального дополнительного расхода памяти и минимального увеличения времени поиска документов.

В таблице представлены виды отношений реализуемой библиографической ИПС для медицины с информационным фондом системы порядка 1 млн. документов.

Указатель вида отношения	Виды отношений	Форма представления при печати (для *B*)	Пример
0	*B* является свойством или признаком *A*	Сокращенное прилагательное	„Рентгенодиагностическая (*B*) установка (*A*)“
1	*B* является методом или средством выполнения действия, представленного *A*	Существительное в родительном падеже	„Лечение (*A*) при помощи актиномицина (*B*)“
2	*B* — объект действия или место действия, обозначаемого *A*, а также отношение части к целому или принадлежность	То же	„Трансплантация (*A*) сердца (*B*)“
3	*B* — свойство или условие, отсутствующее или отрицаемое у *A*	„ „	„Операция (*A*) без наркоза (*B*)“
4	*B* — назначение предмета или цели процесса, обозначаемого *A*	„ „	„Устройство (*A*) для переливания (*B*)“
5	*B* соединено с *A* связкой *и*	Существительное в родительном или местительном падеже либо прилагательное	„Сердце (*A*) или легкое (*B*)“
6	*B* — причина или явление (событие), предшествующее *A*	Существительное в родительном падеже	„Смерть (*A*) после несчастного случая (*B*)“
7	*B* — условие или фактор, характерный для явления или предмета *A*	То же	„Помощь (*A*) при наличии препарата (*B*)“

Примечание. Подчеркнутые слова являются словесными штампами для соответствующих видов отношений.

Порядковый номер дескриптора *A* указывает направление связи для *B*.

На рис 1 представлен бланк поискового образа документа (ПОДа). Заполненный от руки индексатором и снабженный для удобства читателя разметкой (*УВ* — указание вида отношения согласно таблице; *УН* — указание направления отношения, т. е. номера одного из дескрипторов, описанных в данный бланк; *ЛСО* — логико-семантическое отношение, *КД* — код дескриптора)

Бланк состоит из трех рядов «ячеек», предназначенных для записи дескрипторов В первых двух рядах ячейки снабжены номерами (1-я строка ячейки), которые после вписывания дескрипторов в ячейку становится их номерами Последний ряд ячеек номеров не имеет. В нем

могут быть записаны только дескрипторы, к которым связи не направлены. В нижней части бланка содержание документа представлено в виде грамматически связанной фразы, учитывающей ЛСО между дескрипторами поискового образа документа Заполнение бланка проводится следующим образом. Последовательно рассматриваются предложения, записанные в нижней части бланка. Для каждого из них выписываются в ячейках входящие в него дескрипторы (состоящие из одного или нескольких слов) При этом дескрипторы получают номера. Затем в соответствии с таблицей заполняются ЛСО, причем дескрип-

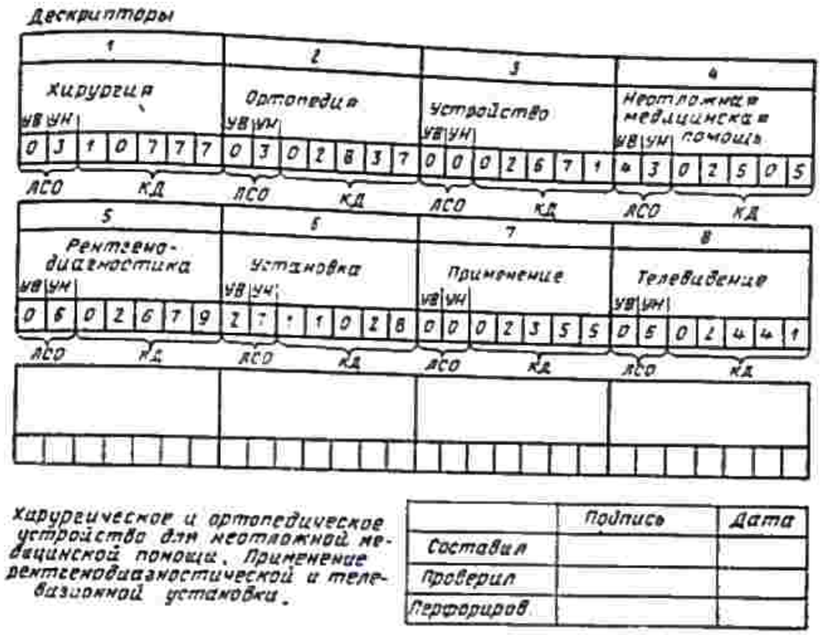


Рис. 1 Бланк поискового образа документа.

тору, являющемуся подлежащим (если подлежащее одно), приписывается $УВ=0$ и, $УН=0$ Если подлежащих несколько и они соединены связками *и* (особый случай, не включенный в таблицу), *УВ* и *УН* заполняются по особым правилам, приведенным на стр 60 Остальные дескрипторы снабжаются аналогичной информацией с помощью таблицы и тезауруса.

Аналогичным образом может быть представлен и поисковый образ запроса (ПОЗ).

Информация ЛСО дескрипторов позволяет представить как ПОД, так и ПОЗ в виде графа, вершинами которого являются дескрипторы, а ребрами — связи, определяемые логико-семантическими отношениями Граф разбивается на уровни значимости, позволяющие обеспечить возможность изменения критерия смыслового соответствия при поиске или при пополнении информационного фонда системы.

В качестве критериев смыслового соответствия ПОДа или ПОЗа тематическому подмассиву может быть использован критерий на полное вхождение в словарь подмассива дескрипторов одного, двух, трех или всех уровней.

В качестве критериев смыслового соответствия ПОЗа ПОДам выбранного тематического подмассива могут быть предложены следующие:

— критерии на полное вхождение дескрипторов соответствующего ПОЗа в ПОД (по одному, двум, трем, четырем уровням) без учета видов отношений между дескрипторами,

— аналогичные критерии с учетом видов отношений между дескрипторами (более жесткий критерий отбора).

Гибкая система выбора критериев смыслового соответствия должна обеспечить достаточно высокие показатели полноты и точности выдачи ответов на запросы данной ИПС.

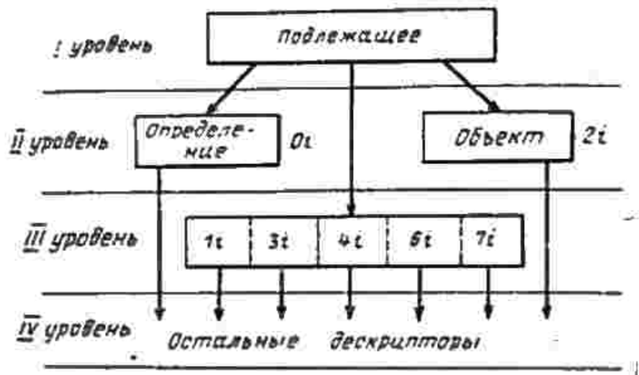


Рис. 2. Схема распределения дескрипторов фразы по уровням значимости.

На рис. 2 представлен один из вариантов разбиения графа ПОДа (ПОЗа) на уровни значимости

К первому уровню относятся лишь подлежащие, ко второму — определения подлежащих (код вида отношения — «0») и объекты (код вида отношения «2»), к третьему — все остальные дескрипторы, имеющие логическую связь с подлежащими, к четвертому — все дескрипторы, не вошедшие в первые три уровня.

Эксперимент, проведенный на массиве в 100 документов, показал, что наиболее эффективным критерием смыслового соответствия ПОЗа ПОДу в смысле полноты и точности является совпадение дескрипторов по первым трем уровням с учетом видов отношений между дескрипторами. В условиях эксперимента к третьему уровню относились определения и объекты для дескрипторов второго уровня; все дескрипторы, не вошедшие в первые три уровня, были отнесены к четвертому уровню. Эти результаты требуют проверки на больших массивах документов и при различного рода запросах.

В настоящей системе поиск может быть многоступенчатым с усилением (или ослаблением) критерия смыслового соответствия до некоторого оптимального, в зависимости от количества документов, подлежащих выдаче на запрос по текущему критерию.

Как уже было сказано, содержание отношений между дескрипторами представлено в таблице. Главные дескрипторы (в некотором смысле подлежащие) имеют код вида отношения, равный нулю. В случае несколько подлежащих, соединенных связкой *и*, код *УВ* полагают либо нулем (только для одного подлежащего), либо порядковым номером того дескриптора, с которым данный связан связкой *и* (для каждого из остальных подлежащих). Код *УН* берут для первого подлежащего равным 0, а для каждого из остальных равным порядковому но-

меру данного дескриптора. Конъюктивная связь (*и*) кодируется специальным образом без использования таблицы.

Кодирование в случае связок *и* для не главных дескрипторов является достаточно простым. Связь достигается повторением одного и того же кода. *УВ* при одном и том же значении *УН*.

При построении фразы принято условие: дескрипторы, связанные между собой отношением *и*, разделять запятыми, а перед последним ставить союз *и*.

До сих пор *УВ* могло быть нулем для главного дескриптора только при *УН*=0. Возможности кодирования и построения формализованных фраз несколько расширяются за счет введения дополнительных грамматических средств и предположения, что главный дескриптор может иметь код *X*0, где *X*≠0. В данном случае этот код интерпретируется как указатель роли.

Ниже приведены значения кодов указателей ролей для главных дескрипторов (*X*=1÷7) и даны соответствующие их представления при печати.

10 — «метод»,
20 — «действие»,
30 — «отрицание»,
40 — «назначение»,
50 — код не используется,
60 — «последствия»,
70 — «условия».

Указатель роли придает своему дескриптору родительный падеж и в предложении предшествует ему. Например, указатель роли 20 для дескриптора «медикамент» дает «действие медикамента».

Заметим, что в описываемой системе определительная связь может быть направлена не только к существительному, но и к прилагательному.

Так, если *A*, *B*, *C* — дескрипторы соответственно с порядковыми номерами 1, 2, 4, причем *B* является определением по отношению к существительному *A*, а *C* является определением по отношению к прилагательному *B*, то будет иметь место именно указанный случай. При индексировании и кодировании следует учитывать, что при построении фразы первым будет поставлено прилагательное, определяющее прилагательное, а затем уже прилагательное, определяющее существительное.

Описанные грамматические средства рассчитаны на введение в ЭВМ минимальной грамматической информации о дескрипторах. При этом используются только: именительный и родительный падежи существительных; сокращенная форма прилагательных; отсутствует понятие рода; ограничен набор вспомогательных слов: «при помощи», «без», «для», «после», «при наличии» и др., а также знаков препинания и союзов «и», «или».

Такие грамматические средства для построения формализованных фраз требуют наличия машинного лексического словаря дескрипторов. Структура лексического словаря дескрипторов представлена на рис. 3.

По коду дескриптора возможно обращение к ячейке, содержащей следующую информацию:

УС — управляющее слово, определяющее порядок выбора лексической информации,

АС — адрес связи с лексической информацией данного дескриптора.

Структура управляющего слова и лексической информации ясна из того же рис. 3.

Реализованный лексический словарь на 10 000 дескрипторов занимает ≈62 тыс. 37-разрядных ячеек памяти (30 зон МЛ по 2 048 ячеек каждая).

Реализация лексического словаря потребовала разработки ряда алгоритмов:

— алгоритма А1 контроля правильности представления исходных данных на перфоленте с последующей выдачей информации на АЦПУ;

— алгоритма А2 пофрагментного формирования и пополнения лексического словаря. Последний алгоритм позволяет формировать лексический словарь, включающий до 16 000 дескрипторов. Под ячейки, содержащие управляющие слова и адреса связи с лексической информацией, отводится 8 зон МЛ (число ячеек должно быть не меньше пред-

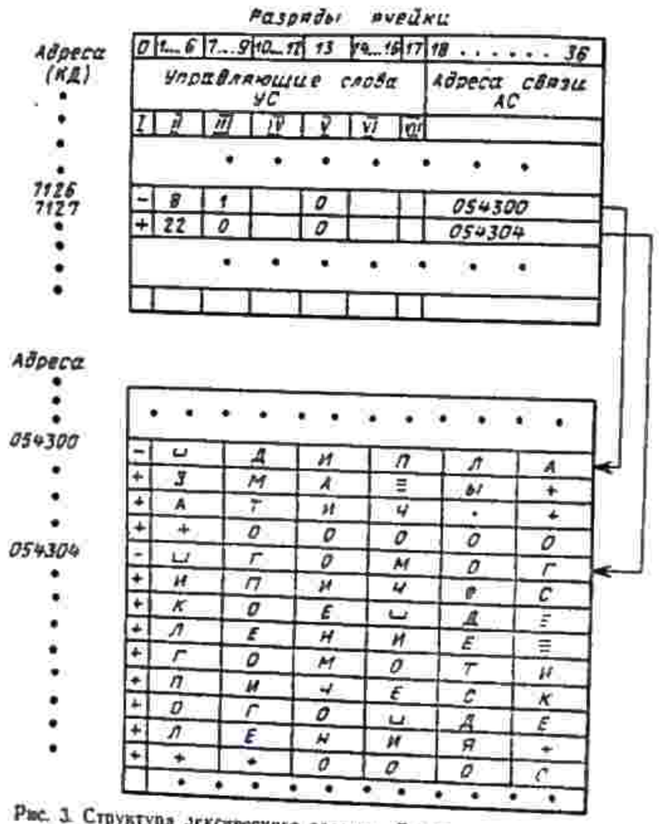


Рис. 3. Структура лексического словаря. Содержимое позиций УС:
I — признак словосочетания, II — число букв в именительном падеже, III — число букв (отнять до основы), V — признак прилагательного, IV, VI, VII — резервные позиции

полагаемого объема тезауруса). Лексические части дескриптора занимают остальные зоны той же МЛ. Максимально возможный объем памяти ЭВМ, занимаемый лексическим словарем, 4 МЛ;

— алгоритма А3 контроля правильности формирования фрагмента словаря. Согласно алгоритму для заданного числа дескрипторов осуществляется обращение к ячейкам, фиксирующим УС и АС, а по значению АС — за лексической информацией каждого дескриптора. При этом на АЦПУ для осуществления визуального контроля по каждому дескриптору выдаются значение кода дескриптора, значения УС и АС, лексическая информация;

— алгоритма А4 внесения исправлений в управляющие слова по заданным кодам дескрипторов и новым значениям управляющих слов;

— алгоритма А5 изменения кодов дескрипторов, ранее введенных

в лексический словарь, по заданным значениям кодов дескрипторов (старых и новых);

— алгоритма «чистки словаря» А6, производящего удаление лексических частей дескрипторов (для которых осуществлена повторная запись новой лексической информации), сдвиг лексической информации для ликвидации свободных полей памяти внутри словаря и соответствующие изменения значений адресов связи для дескрипторов словаря. Машинная реализация алгоритмов представляет собой группу программ, содержащих около трех тысяч машинных команд.

При использовании логико-семантических отношений (смысловых связей) соответствующие ПОДы информационного фонда системы будут содержать информацию, достаточную для осуществления их автоматического аннотирования: в состав ПОДов могут входить при этом до 12

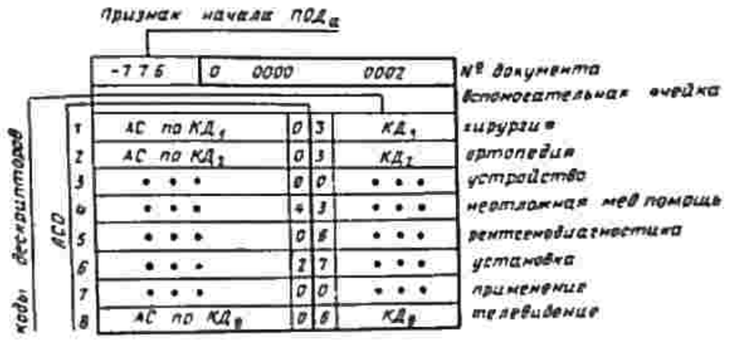


Рис. 4. ПОД, хранящийся в поисковом подмассиве.

дескрипторов, представленных кодами и логико-семантическими отношениями; структура ПОДа представлена на рис. 4. ПОД состоит из ячейки, содержащей признак начала ПОДа и номер документа, вспомогательной ячейки, обычно не занятой и используемой при построении формализованной фразы, и нескольких (до 12) дескрипторных ячеек. Каждая из дескрипторных ячеек имеет следующее распределение разрядов:

0÷16 — разряды, используемые под адреса связи;

17÷22 — разряды, фиксирующие логико-семантические отношения между дескрипторами;

23÷36 — разряды, отведенные под код дескриптора.

Логико-семантические отношения занимают 6 двоичных разрядов:

17÷19 — указатель вида отношения (связи) УВ,

20÷22 — указатель направления связи УН.

Дескрипторы ПОДа пронумерованы ($i=1\div12$); связи могут быть направлены лишь к первым семи дескрипторам ($i=1\div7$).

Реализованный алгоритм автоматического аннотирования документа (составления формализованных фраз по ПОДу) состоит из двух частей.

Первая часть осуществляет упорядочение дескрипторов во фразе путем построения цепного списка, в котором дескрипторы и вспомогательные слова идут в том порядке, в котором они должны располагаться во фразе. При этом для каждого дескриптора указывается, является ли он прилагательным, существительным в именительном падеже, или существительным в родительном падеже. Так, ПОД, представленный на рис. 4, в результате выполнения первой части алгоритма будет преобразован в список, представленный на рис. 5 и определяющий структуру

формализованных фраз аннотации. В этом списке дескрипторы представлены своими кодами.

Вторая часть алгоритма реализует окончательное формирование формализованной фразы в словесном виде. Для этого производится обращение к лексическому словарю, выбираются из него словесные представления дескрипторов ПОДа в форме, определяемой значениями признаков в членах целого списка (признака существительного в именительном падеже, признака существительного в родительном падеже, признака прилагательного, признака вспомогательного слова) и эти словесные представления дескрипторов записываются в порядке следования (связи) членов списка.

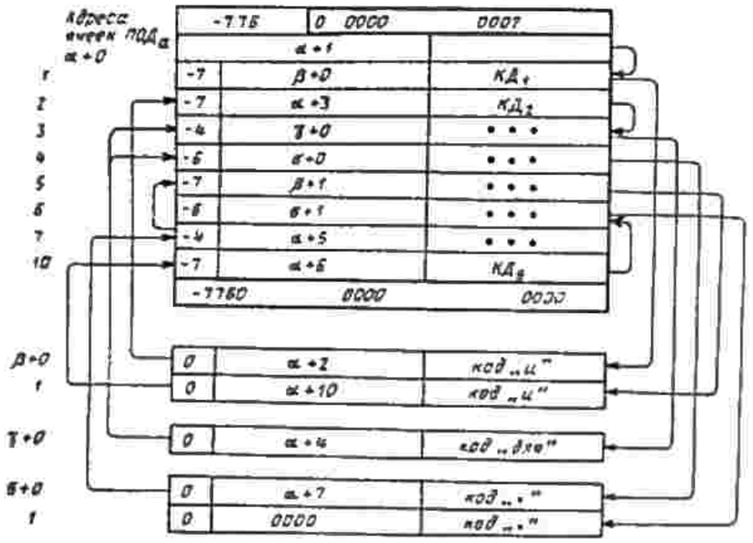


Рис. 5 ПОД, преобразованный в цепной список, определяющий структуру фраз аннотации:
—7 — признак прилагательного,
—6 — признак родительного падежа;
—4 — признак именительного падежа;
0 — признак вспомогательного слова

С помощью алгоритма аннотирования по ПОДу, представленному на бланке рис. 1 и на рис. 4, будет получена следующая аннотация: «Хирургич. и ортопедич. устройство для неотложной медицинской помощи. Применение рентгенодиагностич. и телевизион. установки».

Среднее время выполнения программы построения формализованных фраз для одного ПОДа на машине «Минск-22» составляет 0,6—0,8 м.

На рис. 6 представлена блок-схема алгоритма формирования формализованных фраз по информации ПОДа и лексического словаря.

Количество дескрипторов, входящих в поисковый образ документа, сравнительно невелико (до 12), но даже при наличии таких ограниченных логико-семантических отношений число возможных связей между дескрипторами чрезвычайно велико. В связи с этим для удобства построения списка дескрипторов был использован следующий прием.

В ячейках ПОДа разряды 0—16 отводятся под адрес связи с соседним членом цепочки для данного дескриптора, алгоритм автоматического аннотирования использует эти разряды для вспомогательной информации. Нулевой и первый разряды фиксируют информацию об этапе

обработки данного дескриптора (необработанные дескрипторы имеют код «00», дескрипторы, находящиеся в процессе обработки, — код «10», а обработанные дескрипторы — код «11»). Согласно алгоритму вначале обрабатывается и отмечается кодом «10» первый (по месту в ПОДе) главный дескриптор (управляющий дескриптор), затем для него находятся все зависимые слова, которые также отмечаются кодом «10», после чего отметка главного дескриптора заменяется кодом «11» и он больше обработке не подлежит. На следующем этапе на обработку берется первый дескриптор с кодом «10» и он становится управляющим, т. е. кодом «10» метятся его зависимые дескрипторы, а затем он сам отмечается кодом «11». Далее эта процедура повторяется, причем безразлично, взят ли на обработку дескриптор с кодом «10», относящийся непосредственно к главному дескриптору или к его зависимому. В результате все дескрипторы будут обработаны и помечены кодом «11». Затем среди еще не рассматривавшихся дескрипторов находится новый главный и процедура построения списка следующих фраз повторяется, пока все дескрипторы не будут помечены кодом «11». Параллельно для дескрипторов ПОДа в первый и второй разряды заносятся признаки для работы с информацией лексического словаря:

«00» — соответствует словесному представлению дескриптора в именительном падеже; «10» — в родительном падеже; «11» соответствует словесному представлению дескрипторов в форме прилагательного.

Разряды 4—15 отводятся под адреса связи в списке дескрипторов и вспомогательных слов формируемой фразы. Под вспомогательные слова отведено определенное поле памяти, где расположены адреса связи с их соответствующим словесным представлением: в нулевом и первом разрядах ячеек фиксируется код «00», что является признаком вспомогательного слова. Количество одинаковых вспомогательных слов в формируемых фразах ПОДа ограничено восемью.

Вторая часть алгоритма осуществляет считывание управляющих слов и «лексики» для всех дескрипторов ПОДа в определенное поле памяти и формирование для каждого дескриптора (начиная с начала цепочки) его словесного представления в форме, определяемой информацией в разрядах 1 и 2 соответствующих строк ПОДа и информацией управляющих слов.

Алгоритмы формирования лексического словаря и автоматического аннотирования реализованы на ЭЦВМ «Минск-22» и находятся на стадии внедрения в автоматизированной системе научной медицинской информации.

При индексировании и кодировании смыслового содержания документов, вводимых в информационный фонд системы, наиболее вероятными являются ошибки при кодировании логико-семантических отношений между дескрипторами, что будет являться причиной ошибок при автоматическом аннотировании.

Чтобы избежать выдачи на печать грамматически ошибочных фраз, следует по возможности контролировать правильность построения фразы по ПОДу, а при наличии ошибок выдавать на печать словесную форму представления дескрипторов без учета логико-семантических отношений.

Программа формирования формализованных фраз предусматривает следующий контроль:

— отсутствие подлежащего;

— наличие неупотребляемых кодов;

— ошибки при кодировании УС для дескрипторов в форме словосочетаний;

— разрыв цепочки кодов слов, составляющих формализованную фразу.

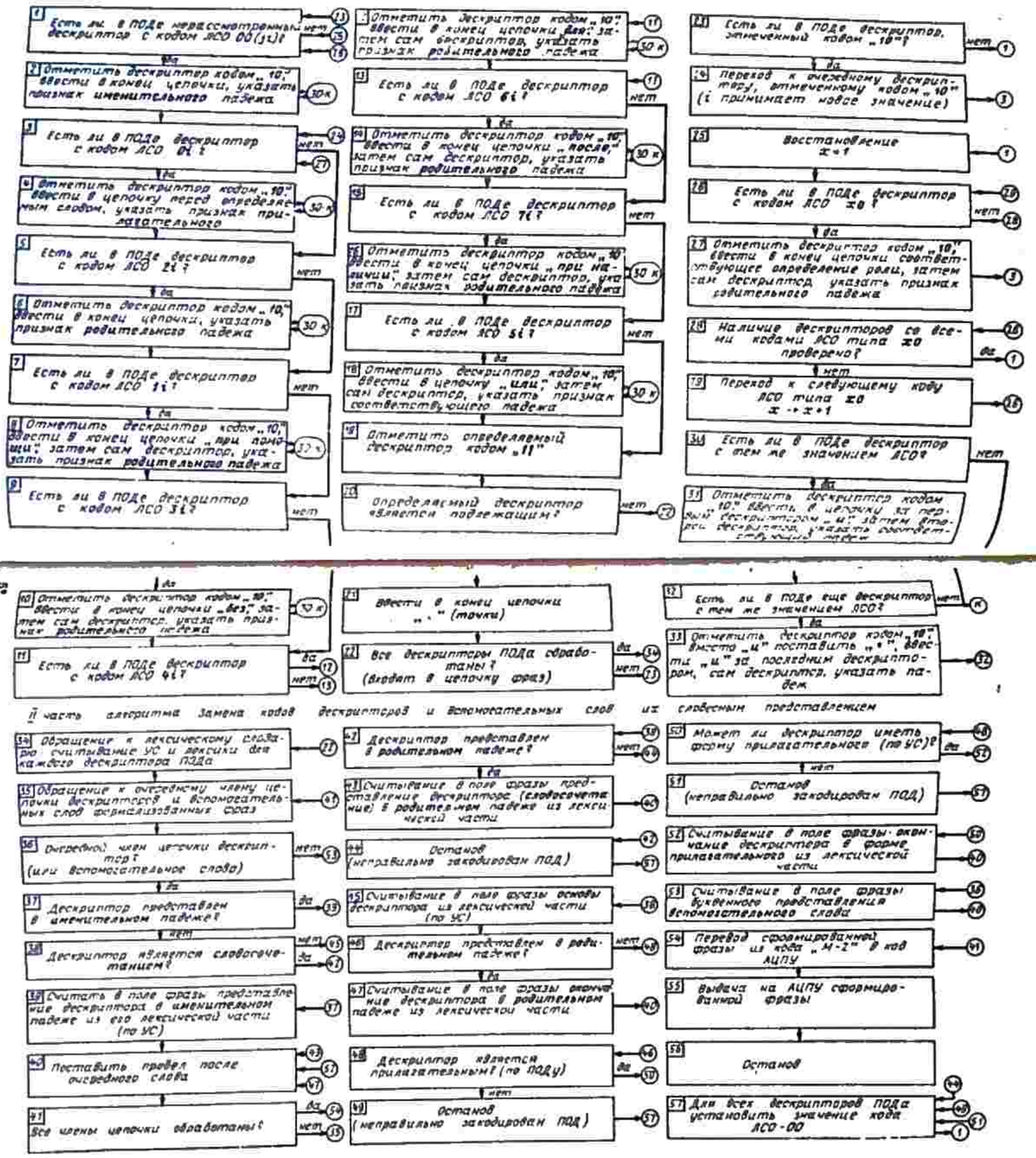


Рис. 6 Блок-схема алгоритма формирования формализованных фраз.

I часть алгоритма Формирование формализованных фраз в виде цепного списка кодов дескрипторов и вспомогательных слов

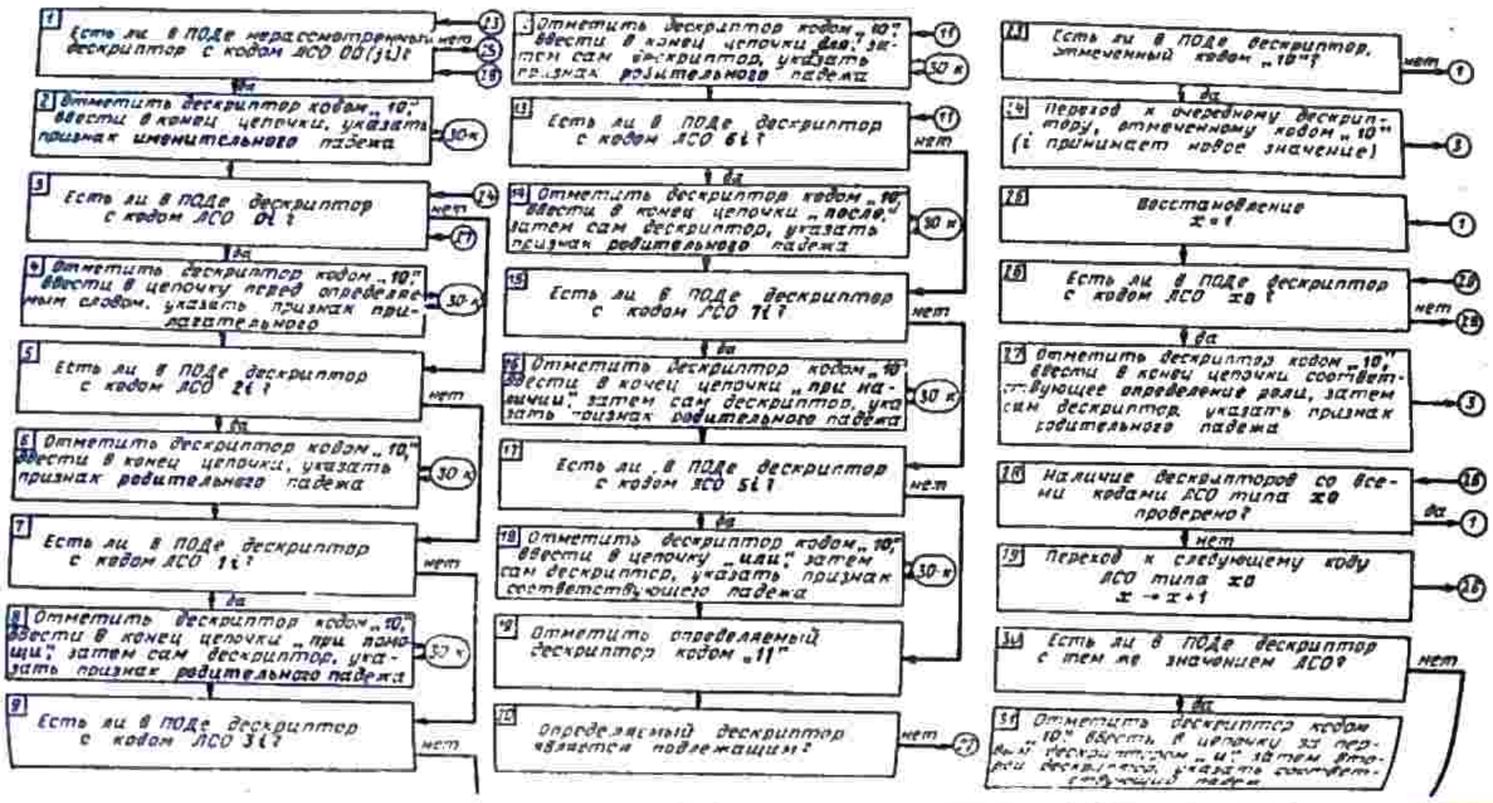


Рис. 6. Блок-схема алгоритма формирования формализованных фраз.

При выявлении одной из перечисленных ошибок управление передается программе, осуществляющей замену в ПОДе текущего документа всех значений логико-семантических отношений кодом «00». При повторном обращении к программе формирования формализованных фраз на печать будут выданы дескрипторы ПОДа в словесном представлении, после каждого дескриптора будет проставлена точка. В этом случае смысловое содержание документа может быть оценено лишь по составу дескрипторов ПОДа

Введение грамматических средств в ИПС для большого массива документов (типа ИПС для медицины) позволяет:

— повысить точность распределения документов по тематическим подмассивам (за счет связей между дескрипторами в ПОДах и их представления в виде графов с разделением дескрипторов в ПОДах по уровням значимости);

— повысить полноту и точность поиска и выдачи ответов на запросы за счет более точного представления смыслового содержания документов,

— облегчить процесс подготовки исходной информации для пополнения информационного фонда системы за счет представления смыслового содержания документа лишь в форме ПОДа (индексирование, перфорация, контроль), при этом более экономично используется память, занимаемая информационным фондом системы (за счет автоматического аннотирования документов по их ПОДам).

Количественные оценки влияния грамматических средств на показатели работы ИПС могут быть получены лишь в процессе ее эксплуатации.

ЛИТЕРАТУРА

1. Китов А. И. Принципы построения ИПС для медицины. В сб. «Цифровая вычислительная техника и программирование», вып. 6. Изд-во «Советское радио», 1973.
2. Китов А. И., Керимов С. К. Использование ассоциативно адресных структур для организации хранения и поиска информации в ЭВМ. «Проблемы создания больших информационно-вычислительных машин и обработки данных на ЭВМ». Всесоюзная научно-техническая конференция. Тезисы докладов. Киев, 1968.
3. Альбани Э., Чеккато С., Маретти И. Математическая лингвистика. Изд-во «Мир», 1964.

УДК 681.3.06

ОРГАНИЗАЦИЯ ПАМЯТИ ДЛЯ ВЫПОЛНЕНИЯ ОБРАЩЕНИЙ ПО ПРИЗНАКАМ

Ф. Д. КОЖУРИН, М. Г. АНТОНЕНКО, Н. Я. ШВЕЦ

1. ВВЕДЕНИЕ

Выполнение обращений по признакам является одной из операций, с которыми часто приходится встречаться при обработке информации.

Большое распространение в практике находят способы организации памяти, характеризующиеся наличием справочного массива, фиксирующего соответствие признак-адрес. Память с произвольным доступом, как известно, не позволяет длительно хранить большие информационные массивы.

Для долговременного хранения больших массивов информации используют память с последовательным доступом, т. е. магнитные ленты. Однако обработка больших массивов информации, записанных на магнитных лентах, занимает много машинного времени.

Широко распространена такая организация массивов, при которой фразы или записи *), из которых состоит массив, могут занимать различное число адресуемых элементов памяти (ячеек), т. е. отличаются по длине.

Применяется также так называемая «плотная» запись информации, при которой начало и конец слов определяются специальными служебными символами (например, пробелами).

При переменной длине фраз для поиска какого-либо признака необходимо производить сравнения по каждому элементу памяти (ячейке), а при «плотной» записи — по каждому символу. Кроме сравнений в последнем случае предварительно необходимо производить выделение символов. Это также увеличивает общее время обработки информации.

В настоящей работе предлагается один метод организации памяти с последовательным доступом для выполнения обращений по признакам, названный методом *отрезков*.

2. НЕКОТОРЫЕ ОПРЕДЕЛЕНИЯ

Процесс обработки информации, размещенной в памяти с последовательным доступом, состоит, как правило, в считывании в оперативную память частей одного или нескольких обрабатываемых массивов и совместной обработке этих частей в оперативной памяти.

Совместная обработка нескольких массивов методически может быть сведена к обработке двух массивов: основного обрабатываемого массива M_0 и массива запросов M_3, содержащего фразы, которые обрабатываются совместно с совпадающими по некоторым признакам фразами массива M_0. Иногда вместо массива M_3 указываются только признаки фраз массива M_0, которые подлежат обработке. Назовем такую обработку массивов *типовой обработкой*.

При типовой обработке информации обращение по признакам производится в два этапа:

1) поиск места нахождения признака на внешнем носителе;

2) поиск признака в оперативной памяти.

Оба пункта выполняются до тех пор, пока не будет найден искомый признак или не выяснится, что он вовсе отсутствует.

В целях сокращения времени обращения по признакам в систему математического обеспечения вычислительных систем целесообразно включить макрооперацию *группового обращения по признакам* в памяти с последовательным доступом.

Сущность макрооперации заключается в использовании специально созданного справочного массива $M_{спр}$, фиксирующего соответствие признак — адрес, для одновременного обращения по максимально возможному количеству признаков.

3. ГРУППОВОЕ ОБРАЩЕНИЕ ПО ПРИЗНАКАМ

В основе группового обращения по признакам в памяти с последовательным доступом лежит понятие отрезка [2].

Отрезком называется максимальная по количеству группа адресов справочного массива $M_{спр}$ (соответствующая фразам массива M_0), признаки при которых сравнились (или поставлены в некоторое заданное соответствие) с признаками массива запросов M_3; каждый из элементов

*) Используемая здесь и ниже терминология заимствована из [2].

(адресов) этой группы должен удовлетворять следующему условию:

$$\max_i |N_{j+i-1} - N_{j+r}| \leqslant (s-1)z, \quad (i=1, 2, \ldots, r; \quad j=1, 2, \ldots, M). \tag{1}$$

где N_j — адрес фразы массива M_0, признак которой совпал (или поставлен в заданное соответствие) с признаком массива M_3; j — порядковый номер фразы массива $M_{сир}$ (при этом учитываются только лишь фразы массива $M_{сир}$, необходимые для поиска), r — количество адресов отрезка (длина отрезка), s — количество фраз массива M_0, помещающихся одновременно в отводимой части оперативной памяти; z — средняя длина фразы, выраженная в адресуемых элементах памяти (в неадресуемых элементах памяти — при «плотной» записи информации), M — количество поисковых признаков массива запросов M_3.

Рассматриваемая организация памяти характеризуется наличием справочного массива $M_{сир}$, фразы которого состоят из признаков фраз «организуемого» (основного) массива M_0 и адресов расположения последних на носителе (с точностью до неадресуемых элементов памяти — двоичных разрядов, триад, тетрад, гексад и т. д. — при «плотной» записи информации).

Массив $M_{сир}$ может храниться как в памяти с последовательным, так и с произвольным доступом. В последнем случае время обращения по признакам будет меньше. Массив $M_{сир}$ формируется путем просмотра основного массива M_0 (массив $M_{сир}$ целесообразно формировать при образовании массива M_0).

Одновременно массив M_0 уплотняется за счет выбрасывания признаков фраз. Массив $M_{сир}$ может храниться как в рассортированном, так и в нерассортированном виде. Массив M_0 остается нерассортированным. Таким образом, отпадает необходимость в сортировке.

Алгоритм реализации макрооперации группового обращения по признакам состоит в следующем.

Последовательно сравниваются признаки массива M_3 ($П_{M_3}$) с признаками $M_{сир}$ ($П_{M_{сир}}$). В случае сравнения первого признака массива M_3 адрес фразы, соответствующей признаку $П_{M_{сир}}$, выносится отдельно в рабочее поле. Затем сравнивается следующий признак $П_{M_3}$. В случае сравнения (т. е. $П_{M_3} = П_{M_{сир}}$) проверяется выполнимость условия отрезков (1). Если (1) выполняется, то адрес фразы массива M_0, соответствующей признаку $П_{M_{сир}}$, выносится в рабочее поле. Происходит сравнение следующего признака $П_{M_3}$ и выполняются остальные операции описанного цикла. Если условие (1) на каком-то этапе не выполняется, то считается, что отрезок найден.

Таким образом, группа адресов, вынесенных в рабочее поле, является максимально возможной, при этом максимально возможным будет и количество признаков, по которым одновременно возможно обращение в памяти с последовательным доступом.

Далее среди всех адресов отрезка выбирается наименьший, который принимается за начальный адрес считывания порции массива M_0. Производится считывание частей массива M_0 и необходимая его обработка в оперативной памяти.

В оперативной памяти выполняются лишь выборки по адресам, получаемым после преобразования адресов на внешнем носителе (памяти с последовательным доступом) в относительные адреса, а затем — в адреса оперативной памяти. Указание адресов расположения информации с точностью до неадресуемых элементов памяти позволяет заранее выбрать константу для выделения символов, слов или групп слов.

Алгоритм формирования отрезка состоит в последовательном присоединении адресов к первому адресу, включаемому в отрезок без всяких сравнений и проверок условий.

При последовательном включении 2-го, 3-го, ..., r-го адреса в отрезок производится проверка условия отрезков нахождением максимального модуля $|N_{j+i-1} - N_{j+r}|$ для всех $i=1, 2, \ldots, r$, а затем непосредственной проверкой условия (1), либо проверкой условия $|N_{j+i-1} - N_{j+r}| \leqslant (s-1)z$ для всех i. Например, при необходимости *) произвести обращение по адресам 300, 150, 610, 520, 1400, 1020, 1500, ..., полагая, что $(s-1)z = 450_8$, включение адресов в отрезки будет производиться следующим образом (рис. 1).

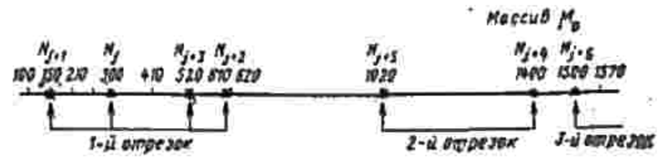


Рис. 1. Иллюстрация понятия отрезка.
Восьмеричными числами обозначены адреса начала фраз.

1-й шаг ($r=1$):

$$i=1, \quad |N_j - N_{j+1}| = 130.$$

Условие (1) удовлетворяется.

2-й шаг ($r=2$):

$$i=1, \quad |N_j - N_{j+2}| = 310,$$
$$i=2, \quad |N_{j+1} - N_{j+2}| = 440.$$

Условие (1) удовлетворяется.

3-й шаг ($r=3$):

$$i=1, \quad |N_j - N_{j+3}| = 220,$$
$$i=2, \quad |N_{j+1} - N_{j+3}| = 350,$$
$$i=3, \quad |N_{j+2} - N_{j+3}| = 70.$$

Условие (1) удовлетворяется.

4-й шаг ($r=4$):

$$i=1, \quad |N_j - N_{j+4}| = 1\,100,$$
$$i=2, \quad |N_{j+1} - N_{j+4}| = 1\,230,$$
$$i=3, \quad |N_{j+2} - N_{j+4}| = 570,$$
$$i=4, \quad |N_{j+3} - N_{j+4}| = 660.$$

Условие (1) не удовлетворяется, поэтому $r=4$ и 1-й отрезок: (300, 150, 610, 520).

Переходим ко 2-му отрезку.

1-й шаг ($r=1$):

$$i=1, \quad |N_{j+1} - N_{j+5}| = 360.$$

Условие (1) удовлетворяется.

2-й шаг ($r=2$):

$$i=1, \quad |N_{j+1} - N_{j+6}| = 100,$$
$$i=2, \quad |N_{j+2} - N_{j+6}| = 460.$$

Условие (1) не удовлетворяется, поэтому $r=2$ и 2-й отрезок: (1 400, 1 020).

*) Необходимые адреса определяются, как и ранее, путем сравнения признаков массивов M_3 и $M_{сир}$.

В 3-й отрезок войдут адреса: (1 500, ..).

Алгоритм образования отрезков также может строиться путем фиксации наибольшего и наименьшего адресов и затем включением новых адресов. При этом требуется в среднем не более двух сравнений.

В зависимости от того «левее» или «правее» от наименьшего адреса (или наибольшего — при этом будет все наоборот) находится включаемый адрес производятся различные проверки:

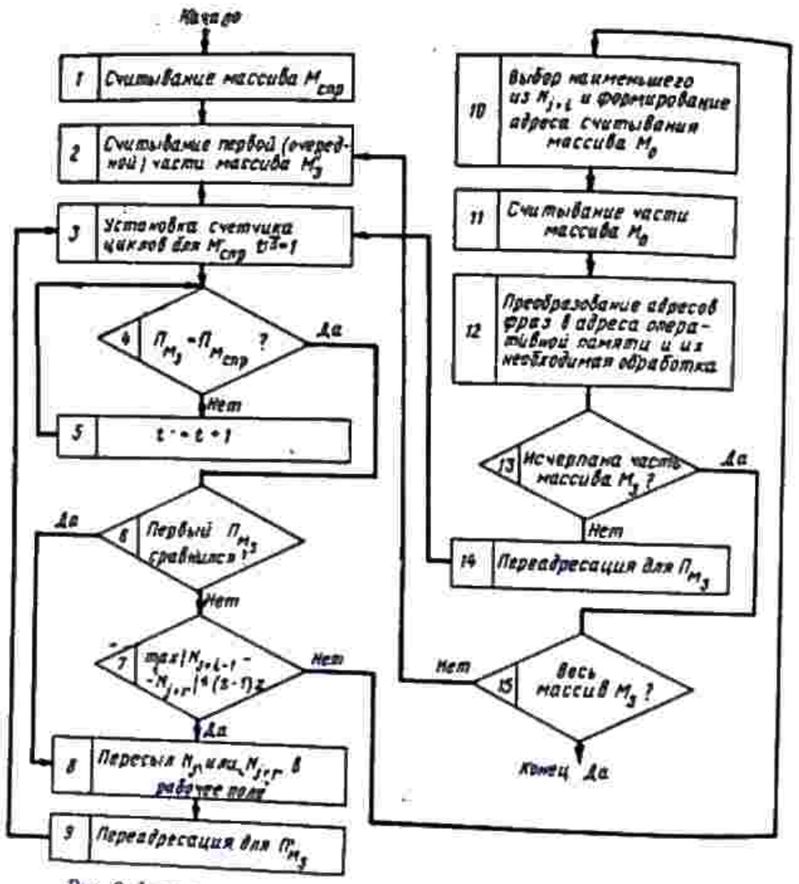


Рис. 2. Блок-схема реализации группового обращения по признакам.

Для простоты изложения массив $M_{спр}$ выбран таким, чтобы он помещался в отведенной для него части оперативной памяти

1) если «левее» и удовлетворяется условие

$$|N_{max} - N_{j+r}| \leqslant (s-1)z,$$

то N_{j+r} включается в отрезок и он принимается за новый адрес N_{min};

2) если «правее» и удовлетворяется условие

$$|N_{min} - N_{j+r}| \leqslant (s-1)z,$$

то N_{j+r} включается в отрезок.

Здесь N_{j+r} принимается за новое N_{max} лишь в том случае, когда N_{j+r} окажется «правее» «старого» N_{max}, на что требуется дополнительная проверка.

Во всех остальных случаях адрес N_{j+r} не включается в данный отрезок.

На рис. 2 представлена блок-схема реализации группового обращения по признакам в памяти с последовательным доступом.

Блок *1* производит считывание массива $M_{спр}$ в оперативную память. (В случае, когда массив $M_{спр}$ большого размера, считывать его эффективно в память с произвольным обращением.) Блок *2* осуществляет считывание в оперативную память частей массива запросов M_3. В блоке *3* выполняется установка счетчика циклов для поиска в массиве $M_{спр}$. Здесь t — значение счетчика. Блок *4* сравнивает заданный признак $П_{M_3}$ с признаком $П_{M_{спр}}$, соответствующим текущему состоянию счетчика циклов t. В случае несравнения осуществляется переход в блок *5*, который изменяет значение счетчика t и возвращает управление в блок *4*. В случае же сравнения управление передается в логический блок *6*, анализирующий «Первый ли $П_{M_3}$ сравнился?» для формирования очередного отрезка. Если «да» — управление передается в блок *9*, где выполняется пересылка N_j в рабочее поле. Если «нет» — управление передается в логический блок *7*, проверяющий условие отрезков. В случае выполнения условия отрезков адрес N_{j+r} включается в отрезок. Управление передается на блок *8*. Блок *9* осуществляет переадресацию на следующий признак $П_{M_3}$ массива запросов и передает управление блоку *3*.

Если для какого-либо очередного N_{j+r}-го адреса условие отрезков не выполняется, он автоматически становится первым адресом следующего отрезка. Управление же при этом передается на блоки *10*, *11*, *12*, реализующие выбор наименьшего из N_{j+1}, формирование адреса считывания части массива M_0, собственно считывание части массива M_0, преобразование адресов в памяти с последовательным доступом в относительные адреса, а затем — в адреса в оперативной памяти, необходимую совместную обработку соответствующих фраз массивов M_3 и M_0.

Блок *13* проверяет, „Исчерпана ли часть массива M_3?" Если „нет" — передает управление на блок *14*, который осуществляет переадресацию для $П_{M_3}$ и передает управление в блок *3*. Если „да" — управление передается логическому блоку *15*.

Блок *15* в случае, если исчерпан массив M_3 (т. е. все запросы удовлетворены), заканчивает работу. Если же массив M_3 не исчерпан — управление передается в блок *2*.

В случае, если структура массива M_0 такова, что некоторому признаку соответствует несколько различных фраз, то этот признак в массиве $M_{спр}$ будет повторен несколько раз с различными адресами. Поиск и выборка нужных фраз при этом осуществляются путем считывания всех различных фраз в соответствии с изложенным методом и сравнения некоторых дополнительных признаков, уточняющих идентификацию фраз.

Введенная макрооперация группового обращения по признакам позволяет эффективно вести обработку нерассортированной информации, используя лишь незначительный (при $z \gg k$, где k — средняя длина признака) объем избыточной памяти для размещения массива $M_{спр}$.

Если массив M_0 рассортирован, групповое обращение по признакам теряет силу, так как первый встретившийся адрес в массиве $M_{спр}$ принимается за начальный адрес считывания порции. Однако предложенная организация памяти с последовательным доступом играет большую роль и в этом случае [4]. Время обращения по признакам будет намного меньше времени обращения при обработке обычного рассортированного массива. Это следует из таких рассуждений.

На обращение по признаку к массиву $M_{спр}$ расходуется времени сравнительно мало (при $z \gg k$).

Очевиден выигрыш времени при поиске фразы в оперативной памяти (он такой же, как и при перассортированном M_0). Выигрыш времени при поиске фраз на внешнем носителе состоит в том, что требуется не просмотр, а прогон информации. Прогон — механическое перемещение носителя. При просмотре кроме тройного прогона (применение контрольного считывания, являющегося обязательным для контроля достоверности информации) требуется дополнительно время на сравнения (а также выделения — при «плотной» записи информации) и поиски в оперативной памяти, которое может быть сравнительно большим при фразах переменной длины и при «плотной» записи.

Предлагаемая организация памяти с последовательным доступом требует наличия многих массивов $M_{спр}$ при необходимости обращения по многим признакам.

Вопросы целесообразности хранения массивов-дублей детально рассмотрены в работе [5]. При организации памяти по методу отрезков хранятся только массивы-дубли $M_{спр}$. При этом недопустимо уплотнение массива M_0 за счет выбрасывания признаков, так как они могут потребоваться для обработки при обращении по другим признакам. При внесении изменений в массив M_0 они одновременно вносятся во все массивы $M_{спр}$.

4. ОБ ЭФФЕКТИВНОСТИ МЕТОДА ОТРЕЗКОВ

Оценку эффективности методов организации памяти необходимо производить по величине дополнительной памяти (т. е. памяти сверх объема, занимаемого основным обрабатываемым массивом) и среднего времени обращения по признаку.

В нашем случае объем дополнительной памяти определится выражением

$$V = M_1(k+\Delta),$$

где M_1 — количество фраз массива M_0; Δ — величина памяти, необходимая для записи адреса фразы на внешнем носителе.

Среднее время обращения по признакам можно определить как отношение фактического времени обращения по всем признакам к числу признаков (массива M_2).

Числитель этого выражения есть величина, которая зависит от статистики распределения признаков в массивах M_0 и M_2 и может быть точно определена лишь в каждом конкретном случае, т. е. и среднее время обращения по признаку можно определить, только зная закон распределения признаков в упомянутых массивах.

Введем понятие коэффициента избыточности, представляющего собой отношение общего количества просматриваемых при обработке информации фраз к количеству фраз, по признакам которых производились обращения:

$$\theta = \frac{\sum_{i=1}^{I} g_i}{\sum_{i=1}^{I} m_i} + 1, \qquad (2)$$

где g_i — количество фраз, которые одновременно считывались в оперативную память, но оказывались пустыми с точки зрения цели данного обращения (обращений — при групповом обращении по признакам);

m_i — количество фраз, оказавшихся непустыми; I — общее количество обращений ко внешней памяти.

Из выражения (2) заключаем, что только при групповом обращении по признакам, позволяющем обращаться по максимально возможному количеству признаков $\left(\sum_{i=1}^{I} m_i = \max \text{ и } \sum_{i=1}^{I} g_i = \min\right)$, коэффициент избыточности θ принимает наименьшее значение, т. е. метод отрезков является *оптимальным* в смысле коэффициента избыточности.

К достоинствам метода отрезков необходимо отнести исключение операций сравнения, выделения и поиска в оперативной памяти.

На основе метода отрезков построены эффективные алгоритмы сортировки больших массивов информации на магнитных лентах, подробное изложение которых приведено в [2].

К недостаткам метода следует отнести необходимость создания и ведения справочного массива $M_{спр}$ и необходимость обращения к нему по признакам.

Более точная оценка эффективности метода отрезков организации памяти с последовательным доступом и сравнительный анализ с существующими методами представляет самостоятельный интерес и является предметом дальнейшего рассмотрения.

Авторы признательны проф. А. И. Китову за ряд замечаний, которые способствовали улучшению статьи.

ЛИТЕРАТУРА

1. Китов А. И. Программирование информационно-логических задач. Изд-во «Советское радио», 1967.
2. Антоненко М. Г., Кожурин Ф. Д. и др. Об одной методике ведения информационных массивов в системах обработки данных. В сб. «Цифровая вычислительная техника и программирование», вып. 6. Изд-во «Советское радио», 1970.
3. Гладун В. П., Кобцева С. А. Методы выполнения обращений по признакам в памяти произвольного доступа. «Кибернетика», 1967, № 2.
4. Кожурин Ф. Д. и др. Метод построения фраз упорядочения и ведения информационных массивов в АСУП и его реализация. Доклад на республиканской научно-технической конференции «Проблемы науки управления и применение вычислительной техники для автоматизации и механизации управленческого труда». Июль, 1968, Киев.
5. Лозинский Л. С. Дублирование массивов в системе обработки данных. «Кибернетика», 1966, № 4.

УДК 681.3.065

О НЕКОТОРЫХ ПРИНЦИПАХ ОРГАНИЗАЦИИ ФАКТОГРАФИЧЕСКИХ ИНФОРМАЦИОННО-ПОИСКОВЫХ СИСТЕМ С ТАБЛИЧНОЙ ФОРМОЙ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ И БИБЛИОТЕЧНЫМ МАТЕМАТИЧЕСКИМ ОБЕСПЕЧЕНИЕМ

М. Б. ГОРИЗОНТОВА

Принципы организации фактографической информационно-поисковой системы (ФИПС) [1, 6] зависят от конкретных условий ее применения. Организация ФИПС и принципы ее функционирования определяются главным образом:

1) способом представления входной и выходной информации;

2) способом хранения информации в запоминающих устройствах машины;

3) способом обработки информации (поиска информации, вычислений и т. д.).

В настоящее время широкое распространение получают ФИПС, в которых информации представляется в табличной форме. Для таких систем существуют некоторые общие пути построения, связанные, в основном, с общностью форм представления информации.

В данной работе излагаются некоторые принципы организации ФИПС, для которых характерна не только определенная форма представления входной и выходной информации, но и вполне определенный принцип организации функционирования системы — библиотечное математическое обеспечение с динамическим распределением памяти. В описываемых системах поиск и обработка информации осуществляется библиотекой стандартных подпрограмм-процедур. Функциональные программные средства обеспечивают автоматическое распределение и экономию оперативной памяти ЭВМ. Системный интерпретатор включает в работу процедурные средства системы, среди которых имеется процедура формирования и выдачи результирующей информации в виде законченных документов.

Состав библиотеки программ-процедур может изменяться, чем достигается ее достаточная эффективность для конкретного класса информационных задач.

1. КОНСТРУКЦИЯ ХРАНИЛИЩА ИНФОРМАЦИИ

Будем считать, что хранилище информации включает в себя произвольное количество объектно-характеристических таблиц (ОХТ) [3]. Попарно различные *объекты* и *характеристики* всех ОХТ объединяются в словари объектов и характеристик, образуя две совокупности поисковых понятий. Кроме того, для каждой ОХТ имеются классификационные словари объектов и характеристик, содержащие следующие сведения:

1) информацию об отношении словарных понятий к этой конкретной таблице (такие сведения позволяют установить связь табличных и словарных понятий);

2) данные о родовидовых связях словарных понятий, относящихся к рассматриваемой таблице (задаются ранг каждого такого понятия и связь, если она существует, с подчиняющим словарным понятием [2]).

3) данные о размещении значений характеристик (эти данные придаются понятиям низшего ранга, для понятий-характеристик указываются номера и расположение во внешней памяти последовательностей значений характеристик, упорядоченных по объектам рассматриваемой таблицы, для понятий-объектов — номера значений характеристик в последовательностях этих значений).

Будем называть номера значений характеристик в последовательностях, относящихся к характеристикам, номерами объектных строк таблицы, а сами последовательности — столбцами значений характеристик таблицы.

Значение характеристики — произвольный набор буквенных, цифровых или буквенно-цифровых кодов, снабженный указателем специфики этого значения (текст, число и т. п.) и признаком конца, в частности в системе возможно хранение и обработка положительных, отрицательных, целых, дробных чисел и числовых интервалов. В целях повышения плотности хранения информации каждый столбец значений характеристики таблицы снабжен информацией о существовании значения характеристики для каждого объекта таблицы [*]. Такая информация позволяет хранить лишь те значения характеристик ОХТ, которые имеют смысл для объектов, введенных в хранилище системы.

При поиске информации выбор классификационного словаря таблицы осуществляется с помощью номера этой таблицы и информации о размещении во внешней памяти классификационных словарей. Номера таблиц определяются с помощью некоторых понятий — параметров таблиц, которые задаются в запросе или сообщении.

[*] В одну таблицу могут быть объединены объекты различной природы, благодаря чему некоторые характеристики для некоторых объектов могут не иметь смысла, например, объектами могут быть «корабль» и «самолет». При этом характеристика «водоизмещение» для самолета бессмысленна (т. е. не существует).

2. ПРОЦЕДУРНЫЕ СРЕДСТВА СИСТЕМЫ

Библиотека стандартных подпрограмм-процедур составляет процедурные средства системы [3, 4]. Она обеспечивает получение ответов на заданные запросы справочного характера и решение некоторого класса информационных задач.

Поисковые процедурные средства системы представлены следующими подпрограммами:

— подпрограммой φ_8 поиска номеров таблиц по заданным их параметрам;

— подпрограммой φ_7, реализующей поиски в словаре понятий и в классификационном словаре некоторой таблицы по заданному поисковому понятию (одинаково используется для объектов и характеристик) данных о размещении значений характеристик;

— подпрограммой φ_6 поиска словарного понятия (характеристики или объекта любого ранга), связанного с понятием низшего ранга, для которого заданы сведения о размещении в хранилище информации (столбца значений характеристики или объектной строки);

— подпрограммой φ_5 поиска значения характеристики, которое соответствует заданному номеру объектной строки в заданном столбце некоторой таблицы;

— подпрограммой φ_4 поиска в столбце некоторой таблицы заданного значения характеристики, которое удовлетворяет одному из условий: «равно», «неравно», «больше», «меньше», «входит в интервал числовых значений характеристики».

Операционные процедурные средства системы включают следующие подпрограммы-операции над информацией:

— подпрограмму φ_1 вызова в оперативную память ЭВМ словаря объектов или характеристик;

— подпрограмму φ_2 вызова в оперативную память ЭВМ классификационного словаря объектов или характеристик по заданному номеру таблицы;

— подпрограмму φ_3 вызова в оперативную память ЭВМ столбца значений характеристики по данным о его размещении во внешней памяти;

— подпрограмму φ_4 теоретико-множественного умножения двух последовательностей объектных строк таблицы, т. е. определения последовательности элементов, присутствующих в каждой из двух заданных последовательностей;

— подпрограмму φ_5 теоретико-множественного сложения в последовательности объектов или характеристик любого ранга, т. е. удаления повторяющихся понятий, и подсчета совпавших понятий;

— подпрограмму φ_6 теоретико-множественного вычитания двух последовательностей номеров объектных строк таблицы;

— подпрограмму φ_7 суммирования числовой информации;

— подпрограмму φ_8 заключительной обработки результирующей информации (информация формируется в виде списка или двумерной таблицы).

Динамический характер системы приводит к тому, что при решении информационных задач может не оказаться в хранилище информации нужных исходных данных.

При формировании ответов на запросы такого рода отказы представляют собой полезную информацию и она доводится до сведения потребителя. При решении же информационных задач важно снять и определенным образом трансформировать частичные отказы. В противном случае может произойти потеря значительной части результирующей информации. Для избежания потери полезной информации каждый процедурный отказ сопровождается формированием специального признака. Этот признак представляет собой комбинацию кодов, отличную от принятых в хранилище информации. Введены специальные признаки, соответствующие отказам следующего содержания

— отсутствие таблицы с заданными параметрами;

— отсутствие в словаре объектов или характеристик заданного понятия;

— отсутствие в классификационном словаре объектов или характеристик таблицы связи с заданным словарным понятием,

— отсутствие в классификационном словаре объектов или характеристик таблицы связи с искомым словарным понятием заданного ранга,

— отсутствие смысла характеристики для заданного объекта,

— отсутствие сведений по характеристике для заданного объекта,

— отсутствие в столбце таблицы значений характеристики, удовлетворяющих одному из задаваемых условий («равно», «неравно», «больше», «меньше» или «входит в числовой интервал»);

— отсутствие результатов (в виде номеров объектных строк) операции теоретико-множественного умножения двух последовательностей номеров объектных строк некоторой таблицы;

— отсутствие результатов операции теоретико-множественного сложения объектов или характеристик любого ранга (в виде понятий);

— отсутствие результатов (в виде номеров объектных строк) операции теоретико-множественного вычитания двух последовательностей номеров объектных строк некоторой таблицы,

— отсутствие результата операции суммирования информации

3. ФУНКЦИОНАЛЬНЫЕ СРЕДСТВА СИСТЕМЫ

Система располагает двумя уровнями библиотек: библиотекой стандартных подпрограмм-процедур и библиотекой рабочих программ [3].

Библиотека рабочих программ — это набор произвольного количества пронумерованных программ решения некоторого класса информационных задач, для которого предназначается система. В библиотеку рабочих программ входит также программа, которая анализирует состояние запроса, организует поиск справочной информации и формирует ее в виде списка или двумерной таблицы. Состав библиотеки рабочих программ можно полностью изменить, расширить или сократить в зависимости от назначения системы.

Рабочая программа — это произвольная комбинация из обращений к библиотечным подпрограммам-процедурам.

Постоянное место хранения библиотек — магнитные ленты. Два уровня библиотек делают систему малозависимой от класса информационных задач и конструкции хранилища информации, так как такие изменения требуют лишь обновления состава библиотеки рабочих программ и коррекции отдельных подпрограмм-процедур.

Система снабжена аппаратом автоматического распределения памяти. В основе организации этого распределения лежит представление словарей объектов и характеристик, классификационных словарей таблиц, столбцов значений характеристик, промежуточной и результирующей информации, полученных при выполнении какой-либо процедуры, в виде массивов информации.

Массив информации — совокупность элементов, однородных по отношению к процедурам, выполняемым над ними. Словари объектов и характеристик, классификационные словари таблиц и столбцы значений характеристик представляют собой упорядоченные массивы информации. Массивы, полученные при выполнении какой-либо логической процедуры или процедуры поиска, как правило, не упорядочены, так как их состав зависит от принятого алгоритма процедуры.

Количество элементов в массиве произвольно. Элементами массивов могут быть поисковые понятия, значения характеристик, номера объектных строк и т. д., а также специальные признаки. Длина элемента (количество его символов) произвольна и фиксируется принятым в системе символом «конец элемента».

Конкретное количество массивов и элементов в каждом из них определяется непосредственно при выполнении рабочей программы.

При автоматическом распределении памяти применяются граничные условия для используемых частей памяти; таблица размещения массивов информации в оперативной памяти ЭВМ, подпрограмма, организующая формирование этой таблицы, заполнение памяти массивами, определение с помощью таблицы размещения требуемых массивов или их отдельных элементов и экономию памяти; блок контроля за переполнением оперативной памяти.

В таблице задается количество элементов в массивах. Удалению из оперативной памяти подвергаются те массивы информации, которые далее не участвуют в выполнении рабочей программы. Номера таких массивов снабжаются признаком «стирания». При стирании корректируется таблица размещения массивов.

Функциональные средства системы обрабатывают три типа символических адресов: адреса запроса, адреса рабочей программы, адреса — номера массивов, и оперируют действительными адресами стандартных ячеек.

Символические адреса запросов задаются в рабочих программах и обеспечивают независимость этих программы от конкретного содержания запросов. Символические адреса рабочих программ позволяют устанавливать эти программы в произвольном месте оперативной памяти ЭВМ. Задание в рабочих программах адресов-номеров массивов освобождает программистов от распределения оперативной памяти ЭВМ.

Функциональные средства системы используют несколько фиксированных участков оперативной памяти: стандартные ячейки, таблицу размещения массивов и рабочее поле.

В стандартных ячейках фиксируется различного рода организационная информация, настраивающая функционирование системы, и исходные данные для выполнения стандартных подпрограмм-процедур. В частности, в них указываются начало запроса, начало и вход в рабочую программу, граничные условия для памяти с автоматическим распределением. Участок стандартных ячеек имеет ограниченный объем.

Каждая стандартная подпрограмма-процедура выполняет следующие функции:

— анализирует тип символических адресов исходных данных и определяет их действительные адреса, используя содержимое некоторых стандартных ячеек и аппарат автоматического распределения памяти;

— оценивает содержание исходных данных, выявляя и трансформируя специальные признаки,

— выполняет собственно процедуру и формирует ее результаты на рабочем поле (в случае наличия исходных данных);

— оформляет содержимое рабочего поля с помощью аппарата автоматического распределения памяти в виде массивов с адресами-номерами, заданными в исходных данных.

Процедуры могут быть многоместны (иметь n входных и m выходных данных).

Особо отметим подпрограммы-операции ψ_5, ψ_7 и ψ_8. Операции ψ_5 и ψ_7 обрабатывают произвольное количество массивов информации с непрерывным накоплением результатов на рабочем поле. Поэтому у этих операций отсутствует оформление содержимого рабочего поля: оно переносится в рабочую программу.

Процедура ψ_8 имеет специфичную организацию, которую мы описываем несколько ниже.

Все стандартные подпрограммы-процедуры самовосстанавливаются.

Исходные данные подпрограмм-процедур задаются в специальной группе стандартных ячеек. В частности, в одной из них задается порядковый номер процедуры.

Оператор обращения к любой подпрограмме-процедуре состоит из следующих команд:

1) задающих порядковый номер процедуры,

2) задающих исходные данные процедуры;

3) обращающихся к системному интерпретатору, который настраивает выполнение процедуры;

4) обеспечивающих возврат на продолжение рабочей программы (используется системная ячейка обратной связи).

Предусмотрено два типа размещения подпрограмм-процедур функционирующей системы: в оперативной памяти и на внешних запоминающих устройствах (в буферной памяти).

В оперативной памяти ЭВМ размещаются системный интерпретатор, подпрограмма автоматического распределения памяти и часто используемые подпрограммы-процедуры.

Остальные подпрограммы-процедуры в начальный момент размещаются в оперативной памяти или на магнитном барабане.

Системный интерпретатор выполняет следующие функции:

— определяет тип размещения (по номеру) подпрограммы-процедуры;

— вызывает в оперативную память подпрограмму-процедуру и передает ей управление (если процедура размещена в оперативной памяти, то лишь передает ей управление);

— оценивает номер предыдущей подпрограммы-процедуры и, если он не совпадает с текущим номером и предыдущая процедура была вызвана в оперативную память, то «стирает» предыдущую процедуру аналогично массиву информации, корректируя при этом табличные данные (при совпадении номеров предыдущей и текущей процедур организует лишь передачу управления текущей процедуре).

4. ОРГАНИЗАЦИЯ ДОКУМЕНТИРОВАНИЯ В СИСТЕМЕ

Процедура заключительной обработки информации ψ_8 обеспечивает автоматизацию формирования двумерных таблиц с произвольной конфигурацией [3].

Классификационные словари таблиц, поисковые и логические операции из процедурных средств обеспечивают полную автоматизацию построения описаний структур справочных таблиц, получаемых при ответах на запросы.

Табличная выдача при решении информационных задач организуется путем задания на специальном языке описания общей («базисной») структуры требуемой таблицы. Это описание задается в рабочей программе в качестве исходных данных оператора обращения к процедуре ψ_8.

Процедура заключительной обработки информации оперирует различными типами адресов, оценивает содержание результирующей информации и определяет оптимальные размеры строк и столбцов таблицы. Процедура представляет собой автономную программу с собственным внутренним управлением и может быть использована в различных фактографических информационно-поисковых системах.

5. ФОРМЫ ЗАДАНИЯ ЗАПРОСОВ И СООБЩЕНИЯ

Запросы справочного характера или на решение информационных задач и сообщения заносятся в специальные бланки. Бланк представляет собой двухполосный формат. Каждая полоса состоит из строк, причем бланки различного назначения имеют их определенное количество.

Левые полосы бланков являются носителями постоянного содержания — различного рода указаний для заполнения правых полос. Эти полосы не кодируются. Правые полосы заполняются потребителем или информатором на формализованном языке.

В запросном бланке обычно указываются следующие данные:

— время задания запроса;

— информация о лице, задавшем запрос;

— порядковый номер рабочей программы, обеспечивающей ответ на запрос;

— параметры таблиц, в которых сосредоточена требуемая информация;

— тип устройства выдачи (алфавитно-цифровое печатающее устройство, рулонный телеграфный аппарат и т. д.),

— тираж выдачи на печатающем устройстве.

Для запроса справочного характера в бланке предусмотрены строки, в которых можно задать произвольное количество поисковых понятий-объектов и характеристик, а также данных для подпрограммы-процедуры ψ_5, обеспечивающей подбор объектов с определенными значениями характеристик.

Для сообщений предусмотрено четыре типа бланков:

1) для задания времени поступления сообщения, информатора, срочности сообщения и параметров таблицы, в которую требуется занести это сообщение,

2) для задания нового поискового понятия и данных о его родовидовых связях (этот тип бланка используется в том случае, когда поисковое понятие в хранилище информации отсутствует);

3) для задания поискового понятия-объекта;

4) для задания словарного номера характеристики, значение которой требуется обновить, собственно значения характеристики с указанием его специфики: текст, число и т. д.

Правила кодирования всех бланков одинаковы.

Функциональные средства системы содержат подпрограмму первичной обработки бланковых записей, в которой предусмотрен контроль правильности запроса. Организация системы позволяет подключить процедуры вторичной обработки бланковых записей в виде кодирования и декодирования поисковых понятий. Тогда станет возможным задавать поисковые понятия на частично формализованном или естественном языке.

6. ОСНОВНЫЕ ПРИНЦИПЫ ОРГАНИЗАЦИИ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ

Функционирование системы состоит в выполнении следующих пяти этапов:

1) перепись с внешних запоминающих устройств в буферную память словарей объектов и характеристик, классификационных словарей таблиц и некоторых стандартных подпрограмм-процедур;

2) первичная обработка запроса или сообщения и установка его в оперативную память с автоматическим распределением;

3) перепись с внешних запоминающих устройств в оперативную память рабочей программы с указанным в запросе номером или ввод ее с перфокарт и ее автоматическая настройка;

4) установка системы в начальное состояние (с магнитных лент переписываются на фиксированные участки оперативной памяти некоторые стандартные подпрограммы-процедуры и функциональные средства системы; интерпретатор, подпрограмма автоматического распределения памяти; в начальное состояние приводятся таблица размещения массивов, рабочее поле и содержимое ряда стандартных ячеек);

5) выполнение рабочей программы.

Предусмотрены различные режимы функционирования системы

1) режим начала функционирования системы,

2) режим продолжения функционирования системы (первый этап не выполняется),

3) режим решения (рабочая программа переписывается с внешних запоминающих устройств в оперативную память ЭВМ),

4) режим отладки (отлаживаемая рабочая программа вводится в оперативную память с перфокарт).

Режимы задаются при вводе в ЭВМ запроса или сообщения. Предусмотрена возможность визуального слежения за функционированием системы перед началом и в конце каждого из этапов на печатающее устройство выдаются стандартные сообщения о том, что представляет собой начинающийся этап, и закончен ли он. Такое слежение позволяет локализовать участок нарушения нормального функционирования системы

Функционирование системы контролируется на трех уровнях:

— этапное слежение за информационным процессом обеспечивает высший уровень контроля;

— контроль среднего уровня осуществляют системный интерпретатор и подпрограмма автоматического распределения памяти (они контролируют переполнение оперативной памяти ЭВМ и нумерацию в рабочей программе массивов информации);

— низший уровень контроля обеспечивают стандартные подпрограммы-процедуры (в них организован контроль непосредственно процедур)

Аварийные состояния системы идентифицируются с помощью выдачи следующих стандартных фраз: «Номер массива задан неверно», «Переполнение оперативной памяти», «Столбец с номером (указывается номер) в хранилище информации отсутствует», «Формируемая таблица превышает допустимые размеры» и т. п.

Для насыщения хранилища информации новыми сведениями в состав библиотеки рабочих программ включена программа «Обновления хранилища информации». Эта программа построена на обращении к поисковым процедурам, так как вырабатываемые ими специальные признаки могут быть использованы для определения состояния хранилища информации. Она реализует следующие виды обновления:

— обновляет словари поисковых понятий;

— обновляет классификационные словари таблиц;

— обновляет столбцы таблиц.

— обновляет постоянные носители информации — магнитные ленты.

7. АВТОМАТИЗАЦИЯ СОСТАВЛЕНИЯ РАБОЧИХ ПРОГРАММ ДЛЯ СИСТЕМЫ. ОТЛАДКА ПРОГРАММ

Библиотечный принцип организации процедурных средств системы, автоматическое распределение памяти и возможность формального описания структуры выходных таблиц создают предпосылки для разработ-

ки проблемно-ориентированного языка программирования информационных задач. Разработки в этом направлении привели к созданию языка «Инфол» и соответствующего транслятора [5]. Объем и время решения транслированных рабочих программ меньше, чем программ, составленных вручную в коде ЭВМ. На составление и отладку транслированных программ требуется существенно меньше времени.

Указанные факты противоречат установившемуся мнению о том, что программы, написанные на алгоритмическом языке, а затем транслированные, обычно «хуже» или во всяком случае «не лучше», составленных вручную в коде ЭВМ. По мнению автора, более высокое качество транслированных программ в данном случае объясняется удачным выбором алгоритмического языка и метода трансляции, а также трудностями программирования вручную для машин третьего поколения.

Наладка системы организуется в два последовательных этапа

1) автономная и совместная отладка на тестовой информации комплекса системных подпрограмм;

2) с помощью специальной рабочей программы отладка хранилища информации (в программе организована выдача информации в табличной форме по каждой ОХТ)

Отладка хранилища целесообразна при ручном кодировании в виде ОХТ первоначального потока сообщений.

Отладка рабочих программ организуется по мере их накопления. Для ведения отладочных работ в функциональные средства системы включены специальные программы, которые определяют ошибки, допущенные в рабочих программах, с точностью до нескольких команд (локальный участок) и сообщают эту информацию программисту. С помощью этих программ можно ознакомиться с состоянием фиксированных участков системы: с содержимым стандартных ячеек, ячейки обратной связи, рабочего поля и таблицы размещения информации в оперативной памяти, с содержимым заполненной массивами информации памяти, а также определить состояние последней выполненной стандартной подпрограммы-процедуры и рабочей программы.

ЛИТЕРАТУРА

1 Криницкий Н. А. О некоторых информационных системах. В сб. «Цифровая вычислительная техника и программирование», вып. 2. Изд-во «Советское радио», 1967.

2 Криницкий Н. А. Таблицы объектно-характеристические. Энциклопедия «Автоматизация производства и промышленная электроника». «Советская энциклопедия», т. 3, 1964.

3 Миронов Г. А., Горизонтова М. Б., Степанченко Д. А. Об организации одной информационно-поисковой системы, предназначенной для выдачи стандартных справок. Всесоюзная конференция 27—30 мая 1968 г. в г. Киеве, ч. III.

4 Горизонтова М. Б., Миронов Г. А. О независимых элементарных поисках и операциях в динамических информационных системах. Всесоюзная конференция 27—30 мая 1968 г. в г. Киеве, ч. II.

5. Степанченко Д. А. Язык для автоматизации программирования стандартных информационных задач «Инфол». Всесоюзная конференция 27—30 мая 1968 г. в г. Киеве, ч. I.

6 Тарасова Н. А., Боз М. М. Фактографические информационные системы на ЭЦВМ (аналитический обзор). Изд. ЦЭМИ АН СССР, 1970.

УДК 681.3.51

ОБ ОДНОМ МЕТОДЕ ОБЪЕДИНЕНИЯ МАССИВОВ

Е. А. МАТЭР, Г. Л. ФЕЛЬДМАН

ВВЕДЕНИЕ

В автоматизированных системах управления одной из наиболее употребительных операций обработки информации является операция объединения массивов с целью подведения различных итогов. Так, например, в автоматизированных системах управления предприятиями

(АСУП) в результате расчетов производных нормативных данных по отдельным изделиям получается ряд упорядоченных массивов, которые необходимо объединять *) для получения показателей по предприятию в целом [1]. Объединяемые массивы, как правило, имеют различные размеры (длины). Число объединяемых массивов в АСУП обычно определяется количеством планируемых изделий.

Для получения упорядоченных массивов больших размеров обычно применяются методы сортировки типа слияния, содержащие внутреннюю и внешнюю стадию сортировки [2, 3]. Эффективность алгоритмов внешней сортировки обычно определяется числом прогонов лент и числом необходимых лентопротяжных механизмов. Эффективность алгоритмов сортировки, как правило, существенно определяет эффективность организации всей системы [1].

Все рассмотрения в данной работе проводятся только для случая использования в качестве внешних носителей информации магнитных лент.

Легко показать, что последовательное объединение массивов M_1, M_2, ..., M_r, расположенных на N зонах магнитной ленты по формуле

$$M = (\ldots((M_1 \cup M_2) \cup \ldots \cup M_r)),$$

является неэффективным, так как может потребовать N прогонов ленты.

Целесообразным является исследование методов объединения, наилучшим образом выбирающих стратегию объединения, с использованием информации об упорядоченности исходных массивов.

Процесс объединения кортежей **) интерпретирован при помощи дерева в работе [4]. Там же предложена стратегия объединения, использующая идею Хаффмана оптимального кодирования из [5] и названная методом минимального дерева. Алгоритм объединения, описанный в работе [6], соответствует построению минимального дерева. Но предложенный алгоритм не учитывает первоначальной упорядоченности.

В настоящей статье рассматривается задача объединения кортежей разной длины, расположенных на магнитных лентах, методом минимального дерева.

Рассматривается последовательность

$$A = \{a_1^{(1)}, a_2^{(1)}, \ldots, a_{n_1}^{(1)}; a_2^{(2)}, \ldots, a_{n_2}^{(2)}; \ldots; a_1^{(r)}, \ldots, a_{n_r}^{(r)}\},$$

состоящая из r кортежей $A_j = \{a_1^{(j)}, a_2^{(j)}, \ldots, a_{n_j}^{(j)}\}$, $j = 1, \ldots, r$, причем:

1) $a_k^{(l)} \geqslant a_i^{(j)}$, если $\begin{cases} k \neq l, & i, j = 1, 2, \ldots, r, \\ k = l, & i \neq j, \end{cases}$

2) $\sum_{i=1}^{r} n_i = N$.

Для простоты будем считать, что кортежи упорядочены в порядке возрастания значений некоторого признака.

Общая задача объединения заключается в том, чтобы получить кортеж $B = \{b_1, b_2, \ldots, b_N\}$ из элементов последовательности A так, чтобы кортеж B был упорядочен в порядке возрастания того же признака, что и кортежи A_j.

Применение метода минимального дерева для внутренней сортировки рассматривается в работе [1]. Указывается, что общее число C операций (сравнения, пересылки), требуемое самим методом, заключено в следующих пределах:

$$N + (m-1)(r-1)\log_m(r-1) \leqslant C \leqslant (m-1) N \log_m r,$$

где N — общее число элементов обрабатываемого массива; r — число объединяемых кортежей, m — база объединения.

Однако при машинной реализации метода появляются дополнительные операции. Показано, что их число $\frac{r(r-1)}{2} + 2r\log_m r$.

*) Объединение — получение одного упорядоченного по возрастанию (убыванию) некоторого признака массива из большего числа упорядоченных по тому же правилу массивов.

**) Кортеж — упорядоченный конечный набор элементов.

1. АЛГОРИТМ ВНЕШНЕГО ОБЪЕДИНЕНИЯ

Предположим, что

1) массив исходных данных содержит r кортежей;

2) исходные кортежи записаны в порядке возрастания их длин на магнитной ленте № 0 *).

Пусть $v(i)$ — длина (число элементов) i-го кортежа, $i = 1, 2, \ldots, r$.

z — размер зоны магнитной ленты; $n_i = \left[\frac{v(i)}{z}\right]$ — число зон магнитной ленты, занимаемых i-м кортежем;

$\sum_{i=1}^{r} n_i = N$ — число зон, занимаемых исходным массивом.

В предлагаемом алгоритме перед каждым шагом объединения **) необходимо знать длины и адреса объединяемых кортежей. Для этого в МОЗУ отводится r ячеек $(a_1, a_2, \ldots, a_r)$, содержащих список длин и адресов кортежей:

$$(a_i) = (N_a^{(i)}, N_s^{(i)}, n_i), \quad i = 1, 2, \ldots, r,$$

где $N_a^{(i)}$ — номер ленты, на которой находится i-й кортеж; n_i — длина кортежа; $N_s^{(i)}$ — номер зоны, с которой начинается i-й кортеж.

Перед началом сортировки и после выполнения каждого шага сортировки список адресов и длин перестраивается в порядке убывания длин кортежей. После каждого шага последние $(m-1)$ ячейки списка исключаются, а в предшествующую им ячейку заносится длина и адрес кортежа, полученного на данном шаге.

Алгоритм в целях сокращения времени перемоток лент строится так, чтобы объединяемые на данном шаге кортежи были расположены на m разных лентах, а результат — на $(m+1)$-й ленте. Поэтому, если некоторые из m кортежей с наименьшей длиной на данном шаге располагаются на одной ленте, то перед объединением они переписываются на разные ленты. Эти ленты в дальнейшем описании будем называть вспомогательными, а то место, которое использовалось на вспомогательной ленте на данном шаге, на следующем шаге считается свободным. Из сказанного следует, что для объединения на базе m необходимо $(m+2)$ ленты (одну с исходным массивом и $m+1$ — рабочих).

*) Требование расположения массива исходных данных на одной ленте не ограничивает общности излагаемого в работе алгоритма. Если кортежи расположены на нескольких лентах и зафиксирован порядок возрастания длин кортежей, то можно рассматривать последовательность лент как единую ленту.

**) Шаг объединения — объединение m $(m<r)$ кортежей.

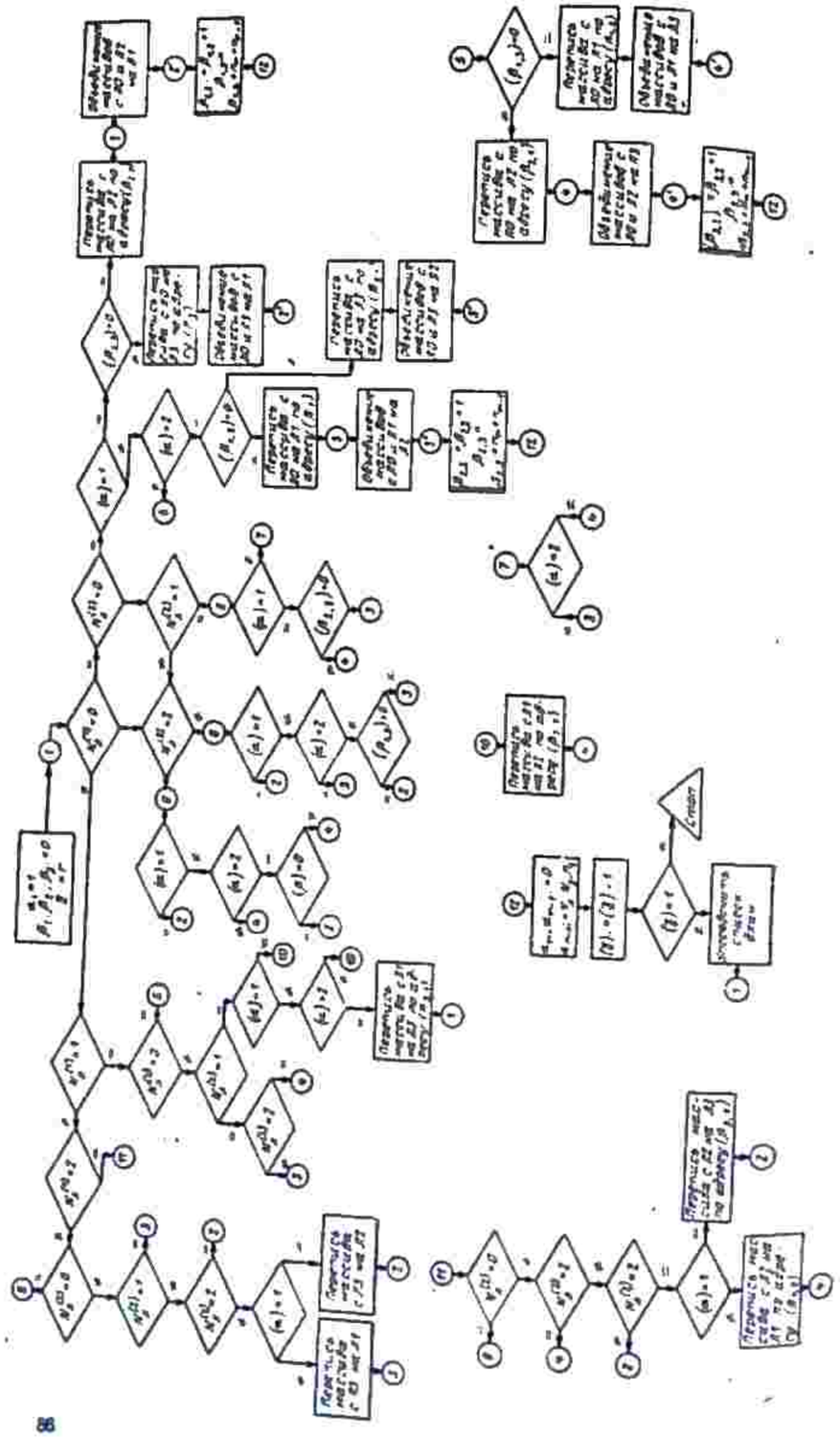


Рис. 1. Блок-схема алгоритма объединения

При построении алгоритма также с целью сокращения времени перемоток лент запись результирующего кортежа, получаемого на данном шаге, осуществляется на ту же ленту, что и на предыдущем шаге, если она не содержит ни один из объединяемых кортежей.

При этом имеют место следующие утверждения.

1. Длина каждого нового кортежа больше (или равна) длин имеющихся кортежей на лентах № 1, 2, 3, ..., $m+1$.

2. Если m кортежей с наименьшей длиной находятся на одной ленте, отличной от ленты № 0, и той, на которую велась запись на предыдущем шаге, то $(m-1)$ рабочие ленты свободны.

3. Если m кортежей с наименьшей длиной находятся на ленте, куда велась запись на предыдущем шаге, то m рабочих лент при этом свободны.

4. Если m кортежей с наименьшей длиной находятся на ленте № 0, то хотя бы $(m-1)$ рабочие ленты свободны.

5. Если m кортежей с наименьшей длиной расположены на разных лентах так, что k ($1 \leqslant k \leqslant m-1$) — на одной, а l ($1 \leqslant l \leqslant m-1$) — на другой, причем $k+l=m$, то $(m-1)$ рабочие ленты свободны.

На основании этих предложений можно сформулировать правила построения алгоритма объединения по двум путям (на базе $m=2$), которые отражены в блок-схеме на рис. 1.

В блок-схеме используются следующие обозначения.

a — номер ленты, на которую велась запись на предыдущем шаге;

$\beta_{1,2}$ — ближайший номер свободной зоны ленты № 1;

$\beta_{1,3}$ — длина массива, записанного на ленте № 1;

$\beta_{2,2}$ — ближайший номер свободной зоны ленты № 2;

$\beta_{2,3}$ — длина массива, записанного на ленте № 2;

$\beta_{3,2}$ — ближайший номер свободной зоны ленты № 3;

$\beta_{3,3}$ — длина массива, записанного на ленте № 3;

γ — число объединяемых кортежей.

II. ОЦЕНКА АЛГОРИТМА ПО КОЛИЧЕСТВУ ПРОГОНОВ ЛЕНТЫ

1. Оценка максимальной длины прогонов лент. Под длиной ленты (части ленты) будем понимать число содержащихся на ней зон размером z.

В алгоритме над информацией, записанной на магнитных лентах, выполняются следующие действия:

1) считывание с ленты (непосредственно при объединении) m кортежей; назовем их полезными считываниями и обозначим их длину Q_1;

2) предварительное считывание $(m-1)$ или $(m-2)$ из m кортежей, расположенных на одной ленте (или на двух так, что l — на одной из них, k — на другой, и $l+k=m$) для записи их на вспомогательные ленты, назовем эти чтения вредными и обозначим их длину Q_2;

3) перемотка ленты на начало записанного на ней кортежа в случае, когда на предыдущем шаге запись велась на эту ленту, а на данном шаге в объединении должны участвовать находящиеся на ней кортежи. Такие перемотки назовем перемотками основных лент и обозначим их длину P_1;

4) перемотка лент № 1 (2, 3, ..., $m+1$) в случае, когда они использовались в качестве вспомогательных, и их необходимо вернуть в исходное положение для очередной записи. Такие перемотки назовем перемотками вспомогательных лент и обозначим их длину P_2.

Введем обозначение $R=Q_1+Q_2+P_1+P_2$. При интерпретации процесса объединения с помощью дерева исходным кортежам соответствуют

конечные точки дерева, а длине n_i i-го кортежа — значение n_i соответствующей точки дерева. Введем обозначение

$$H=\sum_{i=1}^{r} n_i L(i),$$

где r — число конечных точек дерева; $L(i)$ — номер уровня, на котором расположена i-я конечная точка.

Лемма 1. Длина полезных считываний

$$Q_1=H.$$

Доказательство. Q_1 равна сумме длин всех кортежей, включая длины исходных и промежуточных, исключая длину результирующего кортежа суммарной длины N. На дереве каждому кортежу соответствует точка дерева и, наоборот, каждой точке дерева соответствует кортеж, длина которого равна значению точки дерева, следовательно, Q_1 равна сумме значений всех точек дерева без N, т. е.

$$Q_1=\sum_{i=1}^{r} n_i L(i)+N-N=H.$$

Лемма 2. Длина перемоток основных лент $P_1=H-N$.

Доказательство. Если на предыдущем шаге запись велась на некоторую ленту, а на данном шаге в объединении должны участвовать кортежи, находящиеся на этой же ленте, то ее необходимо перемотать на начало первого из рассматриваемых кортежей, который расположен перед всеми записанными на ней кортежами. Эти кортежи на последующих шагах будут подвергнуты полезному считыванию. Так как лента №0 не перематывается, то

$$P_1=Q_1-N=H-N.$$

Лемма 3. Длина Q_2 вредных чтений и длина P_2 перемоток вспомогательных лент не превосходит $\frac{m-1}{m}H$, причем

$$\max_{\sum_{i=1}^{r} n_i=N} Q_2=\max_{\sum_{i=1}^{r} n_i=N} P_2=\frac{m-1}{m}N\lceil\log_m r\rceil.$$

Доказательство. Вредные чтения возникают тогда, когда l из m кортежей с наименьшей длиной находятся на одной ленте ($1<l\leqslant m$). В худшем случае $l=m$; при этом $(m-1)$ кортеж надо считать для записи на разные $(m-1)$ ленты. Сумма длин всех кортежей равна H, следовательно $Q_2\leqslant\frac{m-1}{m}H$. Равенство достигается, если на каждом шаге m наименьших кортежей находятся на одной ленте и длины их равны

При $r=m^k$, где k — целое число, все конечные точки находятся на последнем уровне, и номер последнего уровня равен $\log_m r$, следовательно,

$$H=\sum_{i=1}^{r} n_i L(i)=N\log_m r.$$

В этом случае дерево называется полным (рис. 2).

При $r\neq m^k$ $H=N\lceil\log_m r\rceil$, где $\lceil\log_m r\rceil$ — ближайшее целое число, большее или равное $\log_m r$.

Очевидно, что длина перемоток вспомогательных лент равна длине вредных считываний, т. е

$$P_2\leqslant\frac{m-1}{m}H,\quad \max_{\sum_{i=1}^{r} n_i=N} P_2=\max_{\sum_{i=1}^{r} n_i=N} Q_2=\frac{m-1}{m}N\lceil\log_m r\rceil.$$

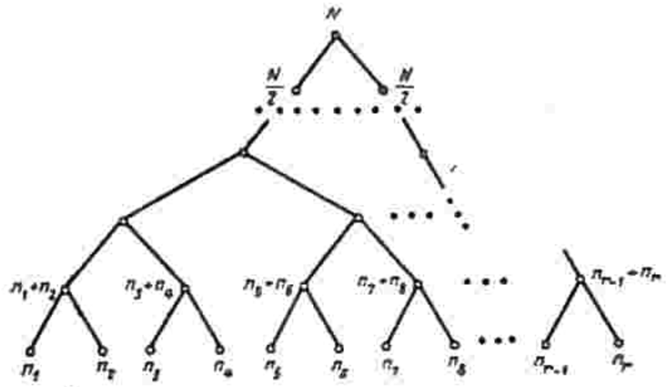


Рис. 2. Полное дерево.

Теорема 1. При сортировке методом минимального дерева на базе m на $(m+2)$ лентах количество L прогонов ленты длиной N не превосходит $\frac{4(2m-1)}{m}\frac{H}{N}-2$, причем

$$\max_{\sum_{i=1}^{r} n_i=N} L=\frac{4(2m-1)}{m}\lceil\log_m r\rceil-2.$$

Доказательство.

$$R=P_1+P_2+Q_1+Q_2\leqslant 2H-N+\frac{2(m-1)}{m}H=\frac{2(2m-1)}{m}H-N.$$

Равенство достигается в случае $P_2=Q_2=\frac{m-1}{m}N\lceil\log_m r\rceil$, т. е. $n_1=n_2=\ldots=n_r$. При этом распределении $H=\max_{\sum_{i=1}^{r} n_i=N} H=N\lceil\log_m r\rceil$, следовательно

$$\max_{\sum_{i=1}^{r} n_i=N} R=\frac{2(2m-1)}{m}N\lceil\log_m r\rceil-N.$$

Поскольку с каждым чтением связана запись информации той же длины, а каждая перемотка осуществляется два раза, то:

1) после перемотки основной ленты с нее производится считывание до тех пор, пока длина записанного на ней куска не станет равна 0, затем на эту ленту вновь будет производиться запись с начальной зоны, т. е. необходимо возвратить ленту на длину считанного с нее куска, равную длине перемотки основной ленты;

2) после перемотки вспомогательной ленты на начало переписанного на нее кортежа этот кортеж считывается при слиянии, т. е. головка ленты снова перемещается на длину кортежа, поэтому, чтобы вести запись на эту ленту как на основную, ее надо перемотать к начальной зоне, длина перемоток лент $L=2R$, и

$$\max_{\sum_{i=1}^{r} n_i=N} L=2 \max_{\sum_{i=1}^{r} n_i=N} R=\frac{4(2m-1)}{m} N\lceil\log_m r\rceil-2N.$$

Этим доказана теорема 1.

Дальнейшие рассмотрения для простоты будут вестись для $m=2$

$$\max_{\sum_{i=1}^{r} n_i=N} L=6N\lceil\log_2 r\rceil-2N.$$

2. Сравнение метода минимального дерева с методом стандартного слияния. Модификация метода стандартного слияния кортежей по 2-м путям на 4-х лентах описана в [7].

Если рассмотреть действия над магнитными лентами аналогично тому, как это сделано для метода минимального дерева, то получим, что количество прогонов ленты длиной N для метода стандартного слияния равно $4\lceil\log_2 r\rceil+4$.

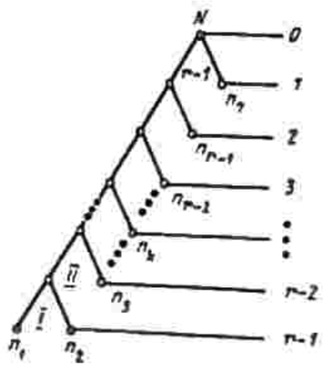


Рис. 3. Ступенчатое дерево.

Величина $\max L$ превосходит $4\lceil\log_2 r\rceil+4$, но могут быть такие распределения (n_i) $(i=1, 2, \ldots, r)$, что $L<4\lceil\log_2 r\rceil+4$, как, например, в случае так называемых ступенчатых деревьев. В этом случае на каждом шаге сортировки, начиная со второго, два кортежа наименьшей длины находятся на разных лентах (один — на ленте № 0, второй — получен на предыдущем шаге). Тогда дерево, названное ступенчатым, имеет r уровней, на каждом из которых две точки, одна — конечная, другая — порождающая семейство предыдущего уровня (см. рис. 3, на котором арабскими цифрами обозначены номера уровней дерева, а римскими — номера шагов сортировки).

Теорема 2.

При распределении $(n_1, \ldots, n_r)$, порождающем ступенчатое дерево, количество прогонов ленты длиной N не превосходит 6.

Доказательство. 1. $P_2=Q_2=n_1$, так как только на 1-м шаге два кортежа наименьшей длины находятся на одной ленте.

2. Обозначим сумму значений конечных точек на j-м шаге сортировки $\varpi(j)$ $(j=1, II, \ldots, r-1)$, тогда на i-м шаге $(i=II, III, \ldots, r-1)$ необходимо делать перемотку основной ленты на длину, равную $\sum_{j=1}^{i-1} \varpi(j)$.

Число шагов равно $(r-1)$, поэтому

$$P_1=\sum_{i=II}^{r-1}\sum_{j=1}^{i-1}\varpi(j).$$

Обозначим класс всех ступенчатых деревьев для данных r и N через

$S_{N,r}$. Найдем в классе $S_{N,r}$ такое распределение (n_i), при котором P_1 принимает максимальное значение:

$$P_1=\sum_{i=II}^{r-1}\sum_{j=1}^{i-1}\varpi(j)=(r-2)\varpi(1)+(r-3)\varpi(II)+\ldots+\varpi(r-2)=$$

$$=(r-1)\sum_{i=1}^{r-2}\varpi(i)-\sum_{i=1}^{r-3}i\varpi(i);$$

$$\sum_{i=1}^{r-2}\varpi(i)=\sum_{i=1}^{r-1}n_i=N-n_r.$$

тогда

$$P_1=(r-1)N-\sum_{i=1}^{r-1}i\varpi(i).$$

P_1 примет максимальное значение в случае

$$\min_{\sum_{i=1}^{r} n_i=N}\sum_{i=1}^{r-1}i\varpi(i)$$

$\sum_{i=1}^{r-1}i\varpi(i)$ примет минимальное значение в классе $S_{N,r}$, если при возрастании i $(i=1, II, \ldots, r-1)$ $\varpi(i)$ возрастает наименее возможным образом $(\varpi(1)=n_1+n_2; \varpi(II)=n_3; \ldots; \varpi(r-1)=n_r)$, т. е. длины кортежей отличаются друг от друга лишь настолько, чтобы сохранился вид ступенчатого дерева. Этот случай имеет место, если выполняются следующие условия:

$$\begin{array}{l|l}
n_1\le n_2\le\ldots\le n_r & n_r=[N/2] \\
n_1+n_2=n_3 & n_{r-1}=[N/2^2] \\
n_1+n_2+n_3=n_4-n_3=n_4/2 & \ldots\ldots\ldots\ldots \\
\ldots\ldots\ldots\ldots\ldots & n_3=[N/2^{r-2}] \\
n_1+n_2+\ldots n_{r-1}=n_r-n_{r-1}=n_r/2 & n_2=[N/2^{r-1}] \\
n_1+n_2+\ldots+n_r=N\to n_r=N/2 & n_1=n_3-n_2=[N/2^{r-1}]
\end{array} \qquad (1)$$

При распределении (1) имеем:

$$\sum_{i=1}^{r-1}i\varpi(i)=n_1+n_2+2n_3+\ldots+(r-1)n_r=N\lfloor 1/2^{r-2}+$$

$$+2(1/2^{r-2}+2\cdot 1/2^{r-3}+3\cdot 1/2^{r-4}+\ldots+(r-1)\cdot 1/2\rfloor=$$

$$=N\lfloor(r-2)+1/2^{r-2}\rfloor, \text{ т. е } \min_{\sum_{i=1}^{r} n_i=N}\sum_{i=1}^{r-1}i\varpi(i)=N\lfloor(r-2)+1/2^{r-2}\rfloor,$$

тогда

$$\max_{S_{N,r}} P_1=N(1-1/2^{r-2})=c_1N,$$

где $0<c_1<1$, $2<r<\frac{N}{2}$,

$$\max_{S_{N,r}} Q_1=N(2-1/2^{r-2})=c_2N,$$

где $0<c_3<2$, $2\le r\le N$

$$\max_{S_N,\ r} L = 2N(c_1+c_2)+4n_3.$$

Эта величина порядка cN, где $c<6$.

Если при решении конкретной задачи наложены ограничения на число лентопротяжных механизмов, то метод минимального дерева выгоднее метода слияния по числу лент для $l>2$ (l — число путей слияния), поскольку он требует $(l+2)$ лентопротяжных механизмов в отличие от $2l$ для метода слияния.

Однако если нет ограничения на число лентопротяжных механизмов, то метод минимального дерева для объединения кортежей может дать абсолютное преимущество в смысле длины перемоток лент, если при объединении по l путям использовать $2l+1$ ленту. При этом алгоритм состоит в следующем.

Исходные кортежи расположены на ленте № 0. В качестве вспомогательных лент будем использовать $(l-1)$ основных лент, с которых в данный момент ведется чтение на данном шаге, а после окончания шага каждая из этих лент возвращается на длину одного массива. Результаты шагов записываются последовательно на ленты № 1, 2, ..., l, 1, 2, ... ($l+1$, ... $l+1$, $l+2$ ...) группы из l лент до тех пор, пока в слиянии не начнут участвовать кортежи с лент этой группы. В этом случае запись ведется на ленты $l+1$, ..., $2l$, $l+1$, $l+2$, ... (1, 2, ..., l, 1, 2, 3, ...). Это возможно, так как из утверждения (1) следует, что если в слиянии участвуют кортежи с лент одной из групп по l лент, то хотя бы одна из лент другой группы к этому времени освободится.

Легко также видеть, что при этом l кортежей с наименьшей длиной всегда расположены на разных лентах (кроме, быть может, кортежей на ленте № 0), поэтому

$$P_3 = Q_3 < \frac{l-1}{l} N.$$

Значения P_1 и Q_1 остались прежними, поэтому

$$L < 4H + \frac{2(l-2)}{l} N.$$

Равенство достигается в случае

$$H = \max_{\sum_{i=1}^{r} n_i = N} H = N\lceil \log_l r \rceil.$$

Следовательно, при всех распределениях $(n_1, n_2, \ldots, n_r)$ метод дает на эффективнее метода слияния $(4N\lceil \log_l r \rceil + 4N)$.

В качестве иллюстрации рассмотрим машиностроительное предприятие, которое характеризуется следующими параметрами:

$N=400$, $r=40$, $N/r=10$.

В табл. 1 приводятся длины кортежей, характерные для решения задачи

Таблица 1

n_1	...	n_{30}	n_{31}	n_{32}	n_{33}	n_{34}	...	n_{37}	n_{38}	n_{39}	n_{40}	..	n_{58}	n_{59}	L	Выигрыш, %
2	...	2	10	15	15	15	...	15	20	20	20	..	20	20	7512	25
2	...	2	15	15	15	15	..	15	15	20	20	...	20	20	7520	25
2	...	2	10	10	10	18	..	18	20	20	20	...	20	20	6580	35

Таблица 2

n_1	...	n_{30}	n_{31}	n_{32}	...	n_{36}	n_{37}	n_{38}	n_{39}	n_{40}	n_{56}	n_{57}	n_{58}	n_{59}	n_{60}	n_{61}	n_{62}	L	Выигрыш, %
1	...	1	16	16	...	15	10	10	10	10	30	30	30	30	30	30	30	7064	30
1	...	1	15	15	...	15	15	15	10	20	20	30	30	30	30	30	30	7112	29,5
1	...	1	15	15	...	15	15	15	15	20	20	20	20	20	30	35	30	7116	29,4

определения потребности предприятия в деталях на программу некоторого периода планирования.

В табл. 2 приводятся длины кортежей, характерные для решения задачи определения спецификированной потребности предприятия в материалах на программу некоторого периода планирования.

При этом длина перемоток лент в случае метода слияния равна 10080, максимальная длина ($4N\log_2 r$) для минимального дерева — 8480, в то время как ступенчатое дерево дает длину перемоток не более 2400.

Из приведенных рассмотрений можно сделать следующий вывод. Математическое обеспечение информационных систем должно содержать в своем составе как алгоритмы сортировки типа слияния, так и алгоритм объединения по методу минимального дерева. Конкретный алгоритм выбирается для решения задач обработки данных в зависимости от информации об исходных данных (значения N, r, распределение $(n_1, \ldots, n_r)$) и от конкретных условий объединения (число свободных лентопротяжных механизмов, объем рабочего поля во внутренней памяти ЭЦВМ и других параметров).

ЛИТЕРАТУРА

1. Матэр Е. А. Некоторые математические вопросы и методы эффективной организации динамических информационных систем для управления предприятием. Информационные материалы ЦЭМИ АН СССР, 1969, сер. 4, вып. 3.
2. Лозинский Л. С. Анализ методов внешней сортировки, использующих слияние. «Кибернетика», 1968, № 1.
3. Лозинский Л. С., Погребинский С. Б. Об одном быстродействующем алгоритме сортировки. «Кибернетика», 1965, № 5.
4. Burge W. H. Sorting trees and order measurement. Information and Control, 1958, v. 1, 3.
5. Хаффман Д. А. Метод построения кодов с минимальной избыточностью. «Кибернетический сборник», 1961, № 3.
6. Шолмов Л. Н. Минимизация числа просмотров исходного массива при внешней сортировке методом слияния. «Кибернетика», 1969, № 5.
7. Корнонова Г. П., Матэр Е. А. Компилирующая система стандартных процедур в нормативной подсистеме АСУП. Информационные материалы ЦЭМИ АН СССР, 1969, сер. 4, вып. 3.

УДК 681.3.62—523.8

ПРИНЦИП ОРГАНИЗАЦИИ ИНФОРМАЦИОННОЙ СИСТЕМЫ (ИС) ОТРАСЛЕВОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ (ОАСУ)

Т. М. АСКЕРОВ, А. П. ФЕЙЗУЛЛАЕВ

Автоматизированные системы управления предприятия (АСУП) обычно являются частями АСУ более высокого уровня, т. е. АСУ высшего уровня управления будет включать в свой состав АСУ низшего уровня управления. С этой точки зрения целесообразно иметь стандартное

структурное построение информационно-поисковых систем (ИПС) для всех уровней управления. При этом алгоритмы ИПС и способы формирования различных массивов будут универсальными. Анализ характера задач АСУП и ОАСУ показывает на возможность выявления типовых процедур обработки экономической информации на всех ступенях управления и создания систем стандартных программ типовых процедур. Тогда на каждом уровне управления конкретные программы обработки будут составляться автоматически из этих стандартных программ. Это дает эффект как с точки зрения применения универсальной системы математического обеспечения, так и с точки зрения экономичного использования объемов памяти. При стандартном структурном построе-

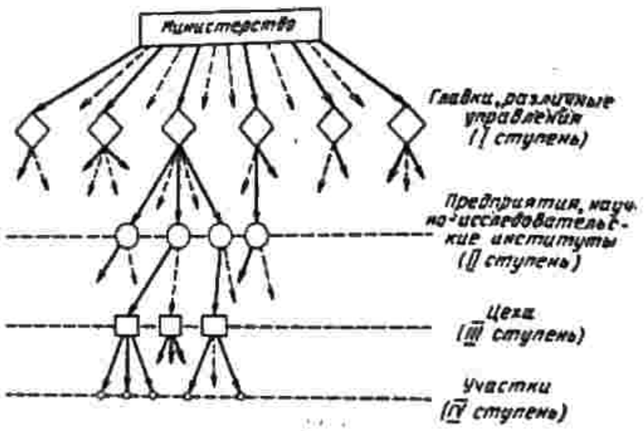


Рис 1 Иерархическая, система управления министерства.

нии ИПС всех уровней управления таких типовых процедур обработки будет минимальное количество

Применяя описанный в [1, 2] метод логических шкал, можно организовать информационную систему (ИС) для различных, в том числе и высших экономических звеньев управления При этом ИС имеет единое построение в глубину, т е одну и ту же структуру для каждой функциональной ступени иерархической системы экономического управления Это дает следующие преимущества

1. Реализация ИС каждой функциональной ступени может производиться с помощью универсальных программ, составленных на основе одних и тех же алгоритмов.

2. Логические шкалы (ЛШ) позволяют весьма компактно организовывать информацию о различных объектах.

3. Макеты носителей первичной информации будут унифицированными, их подготовка и контроль будут вестись единообразно

4. Почти все процессы обработки экономической информации могут быть стандартизованы При этом на основе стандартных (так называемых типовых) процедур образуются программы обработки информации для функциональных ступеней иерархической системы экономического управления

Ниже описывается ИС, применение которой возможно в машиностроительном министерстве

Иерархическую систему управления министерства можно представить состоящей из следующих функциональных ступеней (рис. 1):

1) главки (и соответствующие им управления) — I ступень;

2) предприятия и научно-исследовательские институты — II ступень;

3) цехи предприятий — III ступень,

4) производственные участки — IV ступень.

Обычно организация ИС производится по определенному объектному или предметному принципу В данном случае объектами, информация о которых будет храниться в ИС, являются изделия, материалы и производственные или управленческие органы (главки, предприятия, цехи и т д.). Для построения ИС необходимо сформировать следующие массивы ЛШ. В таблицу сведены все индексы, используемые в дальнейшем изложении

Массив ЛШ «главк — изделия». В этом массиве каждая ЛШ соответствует одному определенному главку министерства, а каждый разряд — одному определенному изделию. Если через i_1 и j_1 обозначим соответственно порядковые номера главков ($i_1=\overline{1, m_1}$) и изделий, выпускаемых предприятиями министерства ($j_1=\overline{1, n_1}$), то количество n_1-разрядных ЛШ будет равно m_1 Таким образом, получится булевская матрица, единичные элементы которой будут отображать распределение изделий, выпускаемых предприятиями министерства между главками министерства

Обозначим данную матрицу через $M_{i_1j_1}$, ее произвольный элемент через $p_{i_1j_1}$, строку этой матрицы через a_{i_1} (фиксированные значения переменных будем обозначать звездочками). Если выделим строку $a^*_{i_1}$ данной матрицы, то ее единичные разряды будут соответствовать изделиям, выпускаемым предприятиями i^*_1-го главка.

Массив ЛШ «изделия—предприятия» формируется для каждой строки a_{i_1} матрицы $M_{i_1j_1}$. Таких массивов будет m_1. Если порядковые номера предприятий, находящихся в подчинении главка $a_{i^*_1}$, обозначим через i_2 (где $i_2=\overline{1, m_{2(i^*_1)}}$), то каждый массив будет представлен булевской матрицей $M_{i_2j'_1}$ размерности $m_{2(i^*_1)}\times n'_1$, где n'_1 есть количество единичных разрядов строки $a_{i^*_1}$, расположенных в прежнем порядке (таким образом устанавливается соответствие между элементами матриц $M_{i_1j_1}$ и $M_{i_2j'_1}$ — $p_{i^*_1j_1}$ и $p_{i^*_1j'_1}$. При этом единичные разряды матрицы $M_{i_2j'_1}$ будут отображать распределения изделий, относящихся к данному главку), между предприятиями, находящимися в подчинении этого же главка (согласно распределению заказов) Обозначим строку данной матрицы через a_{i_2}. Тогда единичные разряды строки $a_{i^*_2}$ будут соответствовать изделиям, изготавливаемым на предприятии i^*_2.

Массив ЛШ «изделия — детали» формируется для каждой строки $a_{i^*_2}$. Очевидно, что таких массивов будет $m_{2(i_2)}$. Для каждого такого массива (матрицы $M_{i_2j''_1}$) разрядность ЛШ будет определяться количеством единичных разрядов в соответствующей $a_{i^*_2}$. Обозначим через i_3 порядковые номера деталей, используемых в производстве предприятия $a_{i^*_2}$ (где $i_3=\overline{1, m_3}$), через n''_1 — количество изделий, выпускаемых данным предприятием, через $j''_1=\overline{1, n''_1}$ — порядковый номер изделия, а через $p_{i_3j''_1}$ — некоторый элемент матрицы $M_{i_3j''_1}$. При этом единичные разряды сформированной матрицы $M_{i_3j''_1}$ будут указывать на применяемость деталей в изделиях, выпускаемых предприятием i^*_2. Размерность данной матрицы будет $m_3\times n''_1$.

Массив ЛШ «детали — материальные ресурсы» представляется булевской матрицей $M_{i_3j_3}$, где $j_3=\overline{1, n_3}$ является порядковым номером

№ п/п	Обозначение объекта	i_1	i_2	i_3	i_4	j'_1	j''_1	i_5	i_6	i_7	i_8	i_9
1	Главки	$\overline{1, m_1}$										
2	Изделия, выпускаемые предприятиями министерства		$\overline{1, n_1}$									
3	Предприятия одного фиксированного главка			$\overline{1, m_2(i^*_1)}$								
4	Детали, используемые на одном фиксированном предприятии $a_{j^*_4}$				$\overline{1, m_3}$							
5	Изделия, выпускаемые предприятиями одного фиксированного главка i^*_1					$\overline{1, n'_1}$						
6	Порядковый номер изделия, выпускаемого предприятием $a_{j^*_4}$						$\overline{1, n''_1}$					
7	Материальный ресурс, используемый на предприятии $a_{j^*_4}$							$\overline{1, n_2}$				
8	Склады предприятия $a_{j^*_4}$								$\overline{1, m_4}$			
9	Цехи и предприятия $a_{j^*_4}$									$\overline{1, m_5}$		
0	Поставщики материалов предприятия $a_{j^*_4}$										$\overline{1, m_6}$	
11	Порядковые номера балансовых счетов											$\overline{1, m_7}$

материального ресурса, используемого в процессе производства данного предприятия. Единичные разряды этой матрицы отображают применяемость материальных ресурсов в изготовлении различных деталей, идущих на производство изделий j''_1.

Массив ЛШ «материальные ресурсы — склады предприятия» формируется для каждого предприятия. Если порядковые номера складов обозначим через $i_8 = \overline{1, m_8}$, то матрица $M_{j_4 i_8}$ будет иметь размерность $n_2 \times m_8$. Таких массивов будет всего $m_{2(i_1)}$. Данная матрица отображает распределение материальных ресурсов, потребляемых данным предприятием, между складами

Массив ЛШ «материальные ресурсы — цехи предприятия» отображает потребляемость материальных ресурсов цехами предприятия в процессе производства. Количество таких массивов будет равно m_2. Обозначим порядковые номера цехов через $i_9 = \overline{1, m_9}$. Тогда матрица $M_{j_4 i_9}$ будет иметь размерность $n_2 \times m_9$.

Массив ЛШ «материальные ресурсы — поставщики» формируется также для каждого предприятия в отдельности. Если через $i_6 = \overline{1, m_6}$ обозначим порядковые номера поставщиков, то матрица $M_{j_4 i_6}$ размерностью $(n_2 \times m_6)$ будет отображать распределение заказов на материальные ресурсы, необходимые для данного предприятия, между прикрепленными к нему поставщиками.

Массив ЛШ «материальные ресурсы — номера счетов бухгалтерского учета» будет отображать прикрепление материальных ресурсов к номерам балансовых счетов. Обозначим порядковые номера балансовых счетов через $i_7 = \overline{1, m_7}$. Тогда матрица $M_{j_4 i_7}$ будет иметь размерность $(n_2 \times m_7)$. Таких массивов также будет $m_{2(i_1)}$.

Остальные, необходимые массивы ЛШ могут формироваться на основе рассмотренных массивов ЛШ. Ниже, в качестве примеров, будет построено несколько таких массивов.

Иерархическая система вышеперечисленных массивов показана на рис. 2. Общее количество таких массивов будет определяться следующей формулой:

$$N = 1 + m_1 + 6 \sum_{i_1=1}^{m_1} (i_2)_{i_1},$$

где $i_2 = \overline{1, m_{2(i_1)}}$.

Рассмотрим объектную часть (ОбЧ) данной ИС, т. е. состав и строение дополнительной информации о каждом из упомянутых выше объектов. Организация объектной части любой ИС является весьма важной [3]. При этом ее структурное построение должно быть согласованным с поисковой структурой ИС. Отметим следующие моменты в строении ЛШ, важные для организации объектной информации. Одноименные строки или столбцы матриц, просматриваемые сверху вниз и слева направо, располагаются в последовательно формируемых матрицах в первоначальном порядке, с учетом исключенных строк или столбцов. При этом они соответствуют единичным элементам ЛШ.

Для каждой матрицы строится объектная часть, которая состоит из строк, представляющих собой последовательность (список) единиц экономической информации (наименование объекта, шифр, характеристики, количественные показатели и др. или же множество подобных единиц). Связь матрицы ЛШ и объектной части осуществляется с помощью адресов связи (АС). Некоторые из вышеперечисленных матриц, кроме ОбЧ, имеют также поля данных (ПД). Данными могут быть как наименование объектов, шифры, характеристики, а также количественные данные или же комбинации перечисленных единиц. В последнем случае производится дополнительная внутренняя индексация данных по позициям. ПД представляет собой последовательный список единиц информации. ПД матрицы состоит из ПД строк (ЛШ). Каждая ЛШ связывается со своими ПД посредством АС. ПД матрицы, состоящей из совокупности ПД строк, имеет обратную связь к своей структурной части (булевской матрицы) посредством АС. Порядок следования полей данных строго соответствует порядку следования единичных разрядов в соответствующей ЛШ. Каждое ПД заканчивается пустой ячейкой (с признаком П, см. [1]). ПД матрицы снабжается фиксатором, который со-

стоит из двух частей. В первой части указывается количество незанятых ячеек в ПД, а во второй части — адрес первой свободной ячейки.

$M_{i_1 j_1}$ имеет две ОбЧ. Первая (ОбЧ-1) состоит из списка глав (строки), а вторая (ОбЧ-2) — из списка изделий (столбцы), выпускаемых предприятиями министерства. При этом для фиксации этих списков могут быть использованы боковые шкалы [2]. Алгоритмы и принцип формирования данных списков и соответствующих ЛШ описаны в [1, 2].

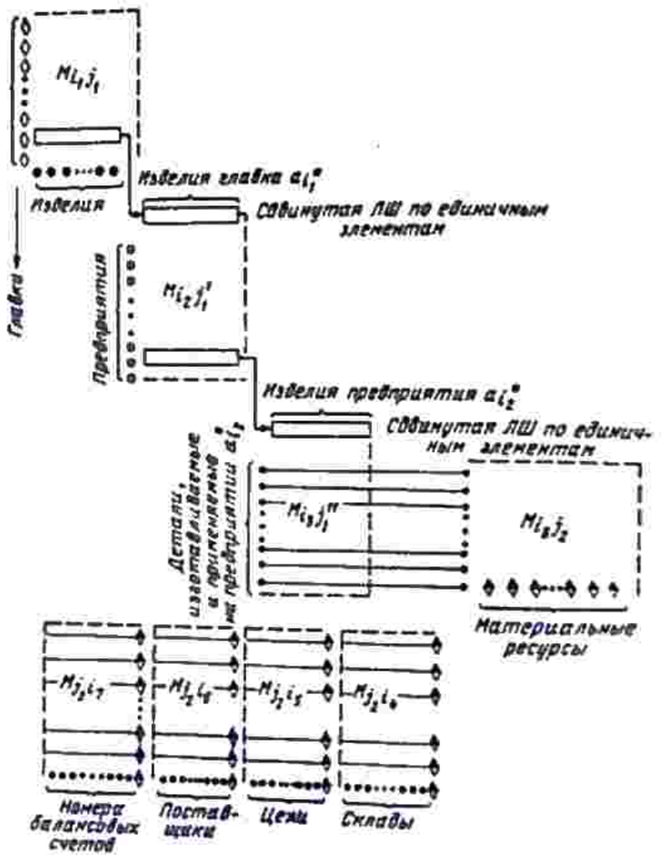


Рис. 2. Иерархическая система массивов информации, организованная с применением логических шкал.

Схема ПД и структурной части данного массива приведена на рис 3. Как видно при этом, массивы ЛШ строк располагаются отдельно. Зачастую их целесообразно хранить вместе с массивом A (если разрядность ЛШ строки не превышает 1—2 ячеек памяти). Данными могут быть различные параметры, относящиеся к изделиям. В некоторых случаях ПД могут отсутствовать. Тогда $M_{i_1 j_1}$ будет отображать только распределение изделий по главкам (массив $Б$). В этом случае массив A получится в два раза короче.

Массив $M_{i_1 j_1 \ldots}$ по структуре аналогичен вышеописанному массиву (ПД и структурная часть). Здесь ОбЧ-1 и ОбЧ-2 соответственно состоят из

перечня предприятий, относящихся к выделенному главку, и списка изделий, выпускаемых только предприятиями данного главка.

Массив $M_{i_1 j_1 \ldots}$ и связанные с ним массивы имеют аналогичную структуру. Разница заключается лишь в том, что ОбЧ-1 и ОбЧ-2 для этого массива соответственно состоят из последовательных списков деталей, расходуемых на данном предприятии, и выпускаемых изделий. При этом ПД в большинстве случаев состоит из целых неотрицательных чисел, выражающих количество определенных деталей, расходуемых на единицу определенного изделия.

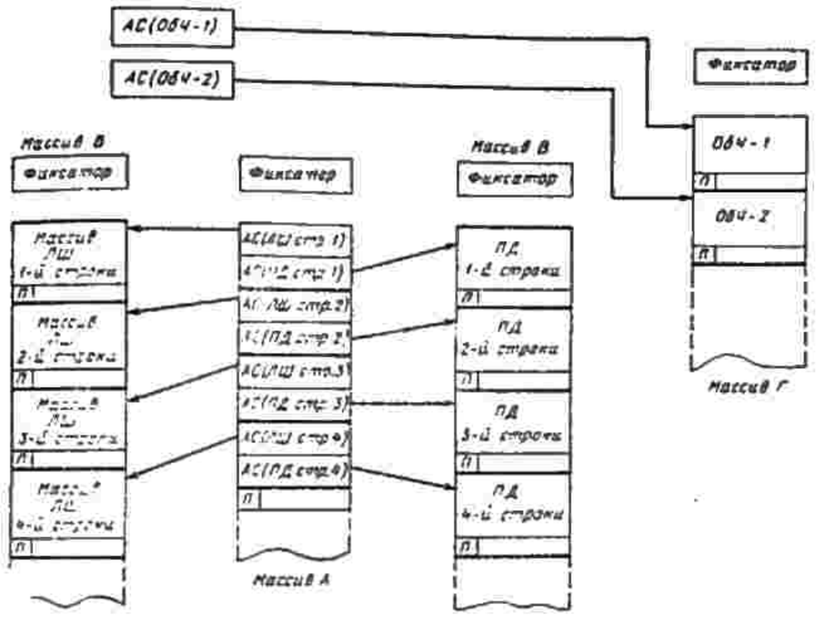


Рис. 3. Схема поля данных (ПД) и структурной части массива.

Обозначим данную величину через $K_{i_1 j_1 \ldots}$. Кроме этого, в ПД могут быть зафиксированы и другие данные, связанные с выделенными деталью и изделием (например, трудовые затраты, геометрические размеры и др.). При этом каждый элемент $M_{i_1 j_1 \ldots}$ будет соответствовать определенному вектору-столбцу, каждая координата которого будет иметь свой индекс (например, сверху вниз 1-я координата — $K_{i_1 j_1 \ldots}$, 2-я координата — геометрические размеры, 3-я координата — трудовые затраты и т. д.).

Матрица $M_{i_2 j_2}$ и связанные с ней массивы организуются аналогично предыдущим. Однако здесь в качестве ОбЧ-1 выступает ОбЧ-1 предыдущего массива (список деталей). Это рационально, так как данные массивы строятся для одного и того же предприятия. ОбЧ-2 является последовательным списком материальных ресурсов, используемых в производственном процессе только данного предприятия.

Особый интерес представляет в этом случае ПД, в котором хранится нормативная информация. При этом каждый элемент $M_{i_2 j_2}$ соответствует трехмерному вектору-столбцу, координаты которого соответствуют норме, черному весу и чистому весу выделенного материала, расходуемого на из-

готовление единицы выделенной детали. Обозначим эти величины соответственно через $H_{i^*d^*_1}$ и $H^{пр}_{i^*d^*_2}$ и $H^{час}_{i^*d^*_3}$ (1-я, 2-я и 3-я координаты). Такая индексация при необходимости введения других сведений может быть продолжена

Следующие группы массивов необходимы для решения оперативных, учетно-статистических и прочих задач. Здесь в качестве ОбЧ-1 выступает ОбЧ-2 предыдущего массива (последовательный список материальных ресурсов)

В качестве ОбЧ-2 для $M_{j_1d_1}$ выступает последовательный список складов предприятий и каждая единица будет соответствовать выделенному материальному ресурсу, который связан с выделенным складом. Каждая единица в ПД будет представляться вектором-столбцом, размерность которого определяется количеством индексируемых величин (например, приходные ордера, количество поступившего на склад материала, количество израсходованного материала, номер цеха, получившего данный материал, остаток и др.).

ОбЧ-2 для $M_{j_2d_1}$ является последовательный список цехов предприятия. В ПД каждому элементу матрицы будет соответствовать вектор-столбец чисел (например, лимитная ведомость цеха, количество израсходованного материала, номер склада, откуда получен материал, дата получения и др.).

Последние две группы массивов организуются для определенных календарных периодов

Для $M_{j_1d_2}$ в качестве ОбЧ-2 выступает последовательный список поставщиков материальных ресурсов, прикрепленных к данному потребителю. Каждой единице данной матрицы в ПД будет соответствовать вектор-столбец определенной размерности (например, заказ, дата поступления материала от поставщика, количество поступившего материала и т. д.).

В качестве ОбЧ-2 для $M_{j_2d_2}$ выступает список номеров счетов бухгалтерского учета; ПД здесь зачастую отсутствует.

Как видно, при необходимости цепь таких групп массивов может быть продолжена

Следует отметить, что результаты обработки экономической информации предлагается организовывать таким же образом, в виде соответствующих массивов и булевских матриц.

Все массивы ЛШ хранятся в сжатом виде. Сжатие производится согласно разработанным алгоритмам сжатия. Такой способ хранения массивов ЛШ, с одной стороны, позволяет эффективно использовать память вычислительной машины, а с другой стороны, улучшает (по временным параметрам) алгоритмы поиска.

Обработка экономической информации производится в следующей последовательности

1. Устанавливается факт наличия или отсутствия элементов информации в НС, отвечающих заданным условиям поиска.

2. При наличии таких элементов информации в НС определяются (расчетным путем) адреса соответствующих массивов и элементов. При этом в большинстве случаев предварительно производится соответствующая логическая обработка шкал, в результате чего избегают выполнения «холостых» операций извлечения элементов информации из НС, т. е. таких элементов, которые по условиям обработки (по характеру решаемых задач) для выполнения расчетов не нужны. Ниже будет рассмотрен подобный пример.

3. Извлекаются из НС нужные элементы информации.

4. Выполняется вычислительный процесс.

5. Производится запись полученных результатов в НС с соответствующей организацией поисковой структуры.

Рассмотрим несколько примеров применения описанной методики для обработки экономической информации.

Пример 1. Требуется определить сводные нормы расхода материальных ресурсов на изготовление отдельных единиц изделий.

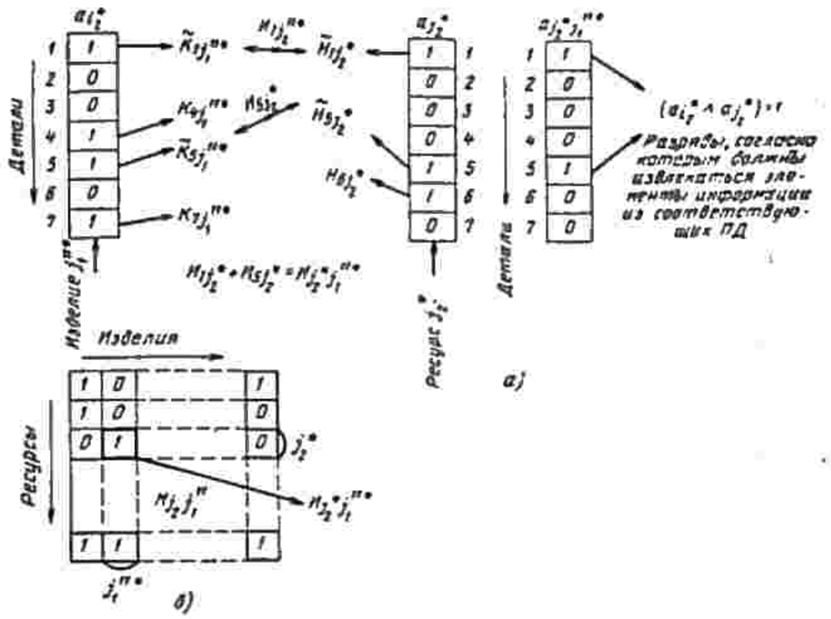


Рис. 4.

Обозначим через $H_{j^*d^*_1 \ldots}$ потребность в материальном ресурсе j^*_2 на изготовление единицы изделия j''^*_1. При решении данной задачи используются массивы $M_{j_1d^*_1}$ и $M_{j_2d_1}$ (см. рис. 2). В этом случае используется первая координата вектора-столбца, которая соответствует определенному единичному элементу матрицы $M_{j_2d_1}$. Причем, что $m_s = 7$, т. е. изделия, выпускаемые предприятием, $a_{i^*_1}$ состоят максимум из 7 деталей (рис. 4, *а*).

Сначала из матрицы $M_{j_1d^*_1}$ по заданному j''^*_1 выделяется столбец ЛШ ($a_{i^*_1}$), который показывает входимость деталей в данное изделие. Затем из $M_{j_2d_1}$ по заданному материальному ресурсу j^*_2 извлекается столбец ($a_{j^*_2}$) ЛШ, отображающий применяемость данного ресурса в изготовляемых на предприятии деталях. После этого производится поразрядное логическое умножение указанных ЛШ. Получается производная ЛШ (обозначим ее через $a_{j^*_2 j''^*_1 \ldots}$), единичные разряды которой соответствуют тем деталям, на изготовление которых требуется материал j^*_2 и они входят в изделие j''^*_1 (этот процесс мы называем предварительной логической обработкой ЛШ). Следующий шаг обработки заключается в определении адресов ПД (полей данных), содержащих искомую информацию. Эти адреса определяются на основе производной ЛШ $a_{j^*_2 j''^*_1 \ldots}$ с использованием ЛШ $a_{i^*_1}$ и $a_{j^*_2}$. Производится поразрядный параллельный сдвиг всех трех ЛШ (начиная с первого верхнего разряда — см. рис. 4, *а*) с одновременным подсчетом единичных

разрядов в ЛШ $a_{i^*_1}$ и $a_{j^*_1}$ и сравнением с соответствующим разрядом ЛШ $a_{f^*_1}$. Например, в ЛШ $a_{i^*_1}$ до 5-го разряда имеется 2 единичных разряда $a_{i^*_1 j^{(2)}_1 \ldots}$, и значение 5-го разряда совпадает со значением 5-го разряда (обязательно единичного) ЛШ $a_{i^*_1 j^{(5)}_1 \ldots}$.

Затем, используя фиксаторы, где указаны начальные адреса соответствующих ПД, прибавляя к ним подсчитанное количество единиц, определяются адреса подлежащих извлечению данных, с обеих ПД. Произведение этих чисел дает потребность в материале j^*_2, идущем на изготовление отдельных деталей, а их суммы дает потребность в этом ресурсе для изготовления единицы изделия $И_{i^*_1 \ldots}$.

Значками (⌒) и (—×→) обозначены соответственно извлекаемые элементы и операция арифметического умножения. $И_{1j^*_2}$ и $И_{5j^*_2}$ обозначают потребности в ресурсе j^*_2 на изготовление деталей *1* и *5*.

Производя вышеописанного типа обработку всех столбцов $M_{i_1 j^{(1)}_1}$ со столбцом $j^{(1)*}_1$, получим сводные нормы расхода всех материальных ресурсов на изготовление данного изделия А если произвести вышеописанный процесс последовательно для каждого столбца матрицы $M_{i_1 j^{(1)}_1}$, то получим производную булевскую матрицу $M_{i_1 j^{(1)}_1}$, единичные элементы которой будут отображать применяемость материальных ресурсов в изготовлении всех изделий предприятия. ПД при этом должны состоять из сводных норм расхода (см. рис. 4,*б*) материалов.

Пример 2. Требуется определить потребности предприятия в различных материальных ресурсах на выполнение квартальной производственной программы.

Для решения данной задачи плановая информация (программа производства) организуется следующим образом. На каждый календарный срок формируется ЛШ. Последовательные разряды данной ЛШ (слева направо) соответствуют изделиям (как в $M_{i_1 j^{(1)}_1}$ и в полученной в предыдущем примере $M_{i_1 j^{(1)}_1}$) и вся ЛШ отражает выпуск изделий в соответствующем календарном периоде (единичные разряды соответствуют изделиям, подлежащим выпуску). ПД будет содержать показатели планового задания. Обозначим календарный период через q (q=I кв., II кв., III кв., IV кв.), программу производства изделия $j^{(1)}_1$ в данном периоде — через $П_{qj^{(1)}_1}$, а массив ЛШ — через $M_{qj^{(1)}_1}$.

Из матрицы $M_{i_1 j^{(1)}_1}$ берется ЛШ ресурса j^*_2 и производится поразрядное логическое умножение с ЛШ программы производства q-го квартала. В результате получается производная ЛШ, единичные разряды которой обозначают те изделия, которые подлежат изготовлению в данном квартале и требуют использования ресурса j^*_2 (рис. 5). Затем из ПД, соответствующих матрицам $M_{i_1 j^{(1)}_1}$ и $M_{qj^{(1)}_1}$, извлекаются числа (предварительно, как и в примере 1, расчетным путем определяются адреса), парное произведение которых дает потребность ресурса j^*_2 на изготовление отдельных изделий, а их сумма выражает общую потребность предприятия в данном ресурсе на q-й период. Если описанный процесс произвести для всех значений j_2 (без суммирования полученных произведений и с суммированием), то получим соответственно матрицы $M_{i_1 j^{(1)}_1 j_2}$ и $M_{i_1 q}$. ПД этих матриц должны состоять соответственно из поздельных и сводных потребностей предприятия в материальных ресурсах в q-м плановом периоде. Полученные результаты записываются в системе памяти машины таким же образом с организацией соответствующей поисковой структуры (рис. 5).

Пример 3. Требуется определить потребности главка в различных материальных ресурсах на q-й плановый период.

При решении данной задачи могут обрабатываться как $M_{i_1 j^{(1)}_1 \ldots}$, так и $M_{i_1 q}$ всех предприятий, относящихся к данному главку. Мы будем рассматривать вариант, где обрабатываются матрицы $M_{i_1 q}$ и связанные с ними ПД. Обработка ведется по следующим этапам. ОбЧ-2 $M_{i_1 j_2}$ всех предприятий главка обрабатывается совместно. Берется из ОбЧ-2 первого предприятия шифр ресурса и во всех ОбЧ-2 оставшихся предприятий ищется этот же шифр.

Затем, согласно найденному шифру во всех ОбЧ-2 извлекаются соответствующие ЛШ матриц $M_{i_1 q}$ (при этом ЛШ одного календарного периода будет состоять из одного двоичного разряда) и по единичным значениям этих разрядов извлекающиеся $И_{j^*_2 j^{(1)}_1 \ldots q^*}$ суммируются. Данная процедура

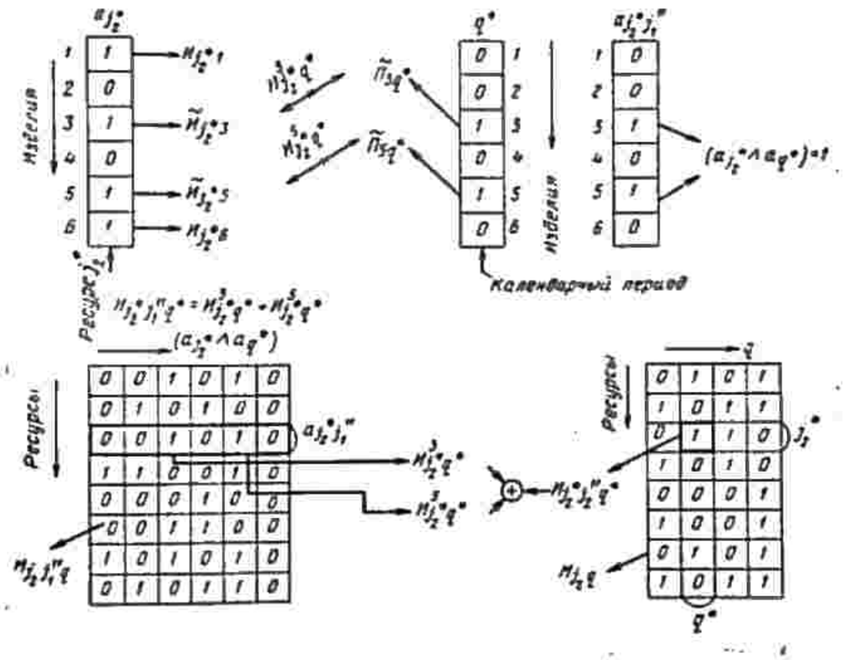


Рис. 5.

производится по всем шифрам материальных ресурсов, используемых на предприятиях этого главка.

Вышеприведенные примеры показывают, что на всех стадиях обработки информации применяются единообразные алгоритмы поиска. При этом для получения одного и того же типа показателей последовательных экономических звеньев обработка информации ведется снизу вверх по иерархической системе.

ВЫВОДЫ

1. Применение принципа логических шкал в организации ИПС дает возможность компактно хранить различные массивы экономической информации и эффективно осуществлять поиск данных по признакам.

2. Метод ЛШ дает возможность на всех уровнях управления производить обработку экономической информации стандартными способами и с относительно небольшой затратой времени. Это достигается в результате применения двухстадийного поиска: а) на основе заданных условий поиска элементов информации предварительной логической обработкой соответствующих ЛШ устанавливается факт наличия или отсутствия искомых элементов в системе, б) расчетным путем (по стандартным программам) определяются адреса подлежащих обработке элементов информации.

3. Применение вышеописанного принципа организации НПС всех уровней управления дает возможность создать централизованное математическое обеспечение всей системы.

ЛИТЕРАТУРА

1. Аскеров Т. М., Асланов А. М. Об одном способе организации информационно-поисковой системы. В сб. «Цифровая вычислительная техника и программирование», под ред. А. И. Китова, вып. 5. Изд-во «Советское радио», 1969.
2. Аскеров Т. М., Асланова О. Об одном способе организации нормативной информации в автоматизированной системе управления материально-технического снабжения (АСУ МТС). Ученые записки Азербайджанского института народного хозяйства им. Д. Буниатзаде, III, 1969.
3. Китов А. И. Программирование информационно-логических задач. Изд-во «Советское радио», 1967.

УДК 681.3.053

МЕТОД ПРОСТЫХ ЧИСЕЛ ДЛЯ КОДИРОВАНИЯ ПОИСКОВЫХ ОБРАЗОВ ОБЪЕКТОВ В ИНФОРМАЦИОННО-ПОИСКОВЫХ СИСТЕМАХ

Ю. И. ВУЛЬ, В. Ф. ИНЯКИН

В настоящее время все большее распространение приобретает разработка информационно-поисковых систем для сбора, хранения и обработки больших объемов информации. Процесс обработки в них заключается в категоризации объектов по той или иной задаваемой комбинации признаков и поиску необходимых объектов.

При разработке программ информационно-поисковых систем широко используется совокупность методов ассоциативного программирования. Сущность этих методов заключается в организации в памяти ЭВМ поисковых массивов объектов по списковой структуре с установлением ассоциативных связей между отдельными объектами системы. Существуют различные способы организации списков, поисковых массивов объектов и алгоритмов поиска объектов, отвечающих заданной комбинации признаков [1].

Весьма распространенным приемом является разбиение всей информации об i-м объекте на две части: поисковый образ объекта ($ПОО_i$) и аннотированное описание объекта ($АОО_i$). $ПОО_i$ формируется набором кодов признаков (дескрипторов):

$$КД_1^{(i)},\ КД_2^{(i)},\ \ldots,\ КД_{m_i}^{(i)}.$$

Определенным образом организованный список дескрипторов с указанием их кодов (словарь, тезаурус, классификатор и т. п.), с помощью которого можно составить краткое описание образа объекта ($ПОО_i$), разрабатывается заранее для данной тематики.

Аннотированное описание составляется в произвольной форме в виде текста. Если поисковые образы и аннотированные описания объектов разделены между собой и представляют два различных массива, то выбирается какой-либо признак, например номер N_i объекта, который указывается одновременно в $ПОО_i$ и $АОО_i$.

Для поиска объектов, обладающих заданным комплектом признаков, составляется с помощью дескрипторного словаря поисковый образ запроса (ПОЗ) — список кодов дескрипторов, соответствующих этим признакам. Определение объектов, соответствующих данному запросу, производится путем проверки на полную входимость всех дескрипторов ПОЗ в список дескрипторов ПОО. По номеру N_i найденного объекта отыскивается соответствующее аннотированное описание, которое и выдается на печатающее устройство машины.

Ниже предлагается метод кодирования поисковых образов объектов и запросов, который, по мнению авторов, вследствие простоты алгоритма поиска целесообразно использовать при программировании информационно-поисковых задач, использующих ограниченное количество признаков (дескрипторов) для категоризации объектов (до 100—150).

ОПИСАНИЕ МЕТОДА

Каждому дескриптору словаря, состоящего из r дескрипторов

$$Д_1,\ Д_2,\ Д_3,\ \ldots,\ Д_r,$$

однозначно присваивается числовой код, выбираемый из ряда простых чисел:

$$(a) = a_1,\ a_2,\ a_3,\ \ldots,\ a_r.$$

Полагая, что

$$КД_i \in (a),$$

принимаем

$$КД_1 = a_1,\ КД_2 = a_2,\ КД_3 = a_3,\ \ldots,\ КД_r = a_r\ \ldots \qquad (1)$$

В этом случае ПОО i-го объекта будет описываться списком из m_i дескрипторов

$$Д_1^{(i)},\ Д_2^{(i)},\ Д_3^{(i)},\ \ldots,\ Д_{m_i}^{(i)},$$

а в машинном изображении — совокупностью кодов

$$КД_1^{(i)},\ КД_2^{(i)},\ КД_3^{(i)},\ \ldots,\ КД_{m_i}^{(i)},$$

т. е.

$$a_1^{(i)},\ a_2^{(i)},\ a_3^{(i)},\ \ldots,\ a_{m_i}^{(i)}.$$

Тогда изображение P_i поискового образа объекта можно записать в виде произведения всех входящих в $ПОО_i$ кодов дескрипторов, т. е

$$P_i = a_1^{(i)} a_2^{(i)} a_3^{(i)} \ldots a_{m_i}^{(i)}, \qquad (2)$$

или

$$P_i = \prod_{j=1}^{m_i} a_j^{(i)}. \qquad (3)$$

Действительно, не может быть двух различных комбинаций (наборов) простых чисел, дающих одно и то же произведение и, наоборот, всякое число может быть разложено на простые сомножители только единственным образом.

При этом следует отметить, что для поиска ПОО, соответствующих заданному ПОЗ, нет необходимости восстановления списка дескрипто-

роа, т. е. разложения P на простые сомножители. Поисковый образ каждого запроса изображается также в виде произведения всех кодов, входящих в него дескрипторов, т. е.

$$Q_\nu = \prod_{\lambda=1}^{n_\nu} a_\lambda^{(\nu)}, \tag{4}$$

где ν — номер запроса; n_ν — число дескрипторов в запросе; $a_\lambda^{(\nu)}$ — код λ-го дескриптора в запросе (простое число); Q_ν — машинное изображение ν-го запроса.

Поиск нужных объектов по запросу заключается в проверке делимости нацело P_t на Q_ν. При программировании эта проверка сводится к определению величины остатка от деления, т. е.

$$R = P_t - Q_\nu \operatorname{entier}(P_t/Q_\nu), \tag{5}$$

и выполнению условия

$$R = 0. \tag{6}$$

Если условие (6) выполняется, то объект N_t соответствует запросу Q_ν, в противном случае не соответствует.

Очевидно также, что при кодировании дескрипторов простыми числами нет необходимости заранее вводить в память машины и постоянно хранить в ней «словарь» простых чисел, так как он может быть получен программным путем.

Можно предложить следующий весьма продуктивный способ получения на ЭЦВМ простых чисел a_j, начиная с $j=4$ ($a_1=2$, $a_2=3$, $a_3=5$).

Для чисел, формируемых по соотношению

$$K_{s+1} = K_s + 3 + (-1)^s, \quad s = 3, 4, 5, \ldots, \quad (K_2 = 5),$$

производится проверка делимости нацело каждого из них на уже найденные предыдущие простые числа в диапазоне

$$5 \leqslant a_j \leqslant \sqrt{K_s + 1}$$

Если число нацело не разделилось, оно является простым, в противном случае оно отбраковывается. При этом способе для получения, например, сотого ($a_{100}=541$) простого числа необходимо произвести лишь (после получения $a_{99}=523$) проверку деления нацело (после отбраковки чисел 527, 529, 533, 549 путем всего 19 делений) числа 541 на 7 простых чисел (5, 7, 11, 13, 17, 19, 23).

Аналогично (при необходимости) можно восстановить и список дескрипторов ПОО или ПОЗ по известному их произведению, используя метод канонического разложения натурального числа.

Существенным недостатком метода является ограниченность его применения. Это связано с тем, что разрядность современных ЭВМ позволяет работать с целыми числами в относительно узком диапазоне ($\pm 10^{12}$ — для БЭСМ-6, $\pm 10^{10.8}$ — для ЭВМ серий «Минск» в режиме с фиксированной запятой).

Определим зависимость между разрядностью τ ЭВМ, числом дескрипторов r в словаре и максимально возможным числом дескрипторов m в поисковом образе.

Пусть имеется словарь кодов дескрипторов

$$a_1, a_2, a_3, \ldots, a_r,$$

где

$$a_1=2, \ a_2=3, \ a_3=5, \ a_4=7, \ldots, a_r = A.$$

Рассмотрим тот неблагоприятный случай, когда поисковый образ описывается m дескрипторами

$$a_{r-m+1}, \ a_{r-m+2}, \ldots, \ a_r.$$

Тогда

$$P = \prod_{j=1}^{m} a_{r-m+1}.$$

Предположим, что $a_{r-m+1}, a_{r-m+2}, \ldots, a_r$ близки по значению к a_r, тогда $P' = a_r^m$, причем $P' > P$. Для записи P' как целого двоичного числа в одну ячейку памяти машины необходимо выполнение условия

$$P' \leqslant 2^\tau - 1 \text{ или } P' < 2^\tau. \tag{7}$$

Приняв $P' \approx P$, получим

$$a_r^m < 2^\tau. \tag{8}$$

Для $r \leqslant 150$ можно утверждать *), что

$$A = a_r \approx 1{,}2\, r \ln(r+1) \tag{9}$$

Эта эмпирическая формула в указанном диапазоне дает погрешность, не превышающую 10% от простого числа. Тогда условие (8) будет иметь вид

$$[2{,}75\, r \lg(r+1)]^m < 2^\tau, \tag{10}$$

откуда

$$m < \frac{0{,}3\tau}{0{,}44 + \lg r + \lg\lg r}. \tag{11}$$

Таким образом, по формуле (11) можно проверить, применим ли данный метод при известных значениях общего числа r применяемых признаков (дескрипторов) и максимально возможного числа m дескрипторов в поисковом образе.

*) Эта эмпирическая формула получена авторами.

ЛИТЕРАТУРА

1. Китов А. И. Программирование информационно-логических задач. Изд-во «Советское радио», 1967.

УДК 681.3.06

ОБ ОДНОМ СПОСОБЕ ОРГАНИЗАЦИИ ПОИСКОВОГО МАССИВА В БИБЛИОГРАФИЧЕСКИХ ИПС

Д. Д. АРНАУДОВ

Вопрос организации поискового массива при разработке информационной системы возникает тогда, когда поисковый массив становится настолько большим, что невозможно или, по крайней мере, неэкономично просматривать каждый элемент информации для проверки его соответствия запросу. В этом случае поисковому массиву придают специальную структуру, обеспечивающую возможность осуществления поиска информации за приемлемое время.

В настоящей статье рассматривается вопрос построения поискового массива и дескрипторного словаря, дающих возможность создать эффективную информационно-поисковую систему. Предполагаются большой объем массива документов и словаря дескрипторов и ограничен-

ные объемы внешней и оперативной памяти (как например, у ЭВМ «Минск-22»). В качестве основных принципов построения информационно-поисковой системы приняты:

— иерархическое (многоуровневое) построение поискового массива и процесса поиска,

— ассоциативно-адресный способ расположения информации об объектах в виде списков;

— разделение (сегментация) однородных списков на части с целью сокращения их средней длины, а отсюда — и времени поиска.

Пополнение системы новыми документами происходит в хронологическом порядке

В соответствии с указанными принципами для организации эффективного поиска при наличии большого объема информационного массива и дескрипторного словаря будем использовать четыре типа массивов:

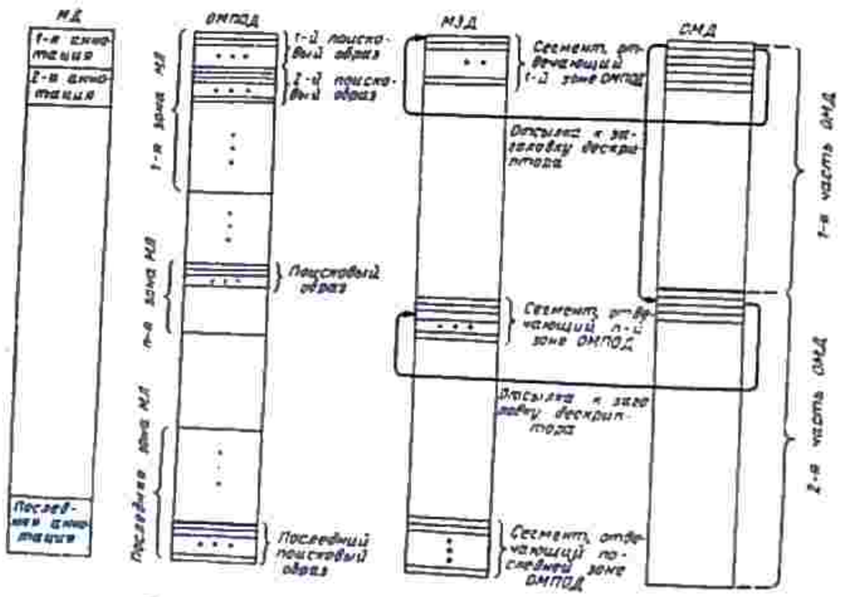
1. МД — массив библиографических данных о документах, называемый кратко массивом документов.

2. ОМПОД — основной массив поисковых образов документов, построенный в виде ассоциативных узловых списков.

3. МЗД — массив заголовков дескрипторов.

4. ОМД — основной массив дескрипторов.

Общая структурная схема системы показана на рисунке. Поясним сущность каждого из массивов.



Структурная схема информационно-поисковой системы.

Массив документов (МД) в поиске не участвует. К нему происходит обращение только после того, как поисковые образы необходимых документов найдены. Массив библиографических данных строится на МЛ следующим образом. Магнитная лента разбивается на зоны и внутри каждой зоны последовательно размещаются аннотации документов.

Массив ОМПОД строится как отображение массива документов. Это значит, что индексы (адреса) первых ячеек узлов в ОМПОД должны находиться в соответствии с индексами (адресами) аннотаций документов в массиве документов. Это соответствие может быть реализовано различными способами. Например, оно может осуществляться с помощью адресов связи, включенных в состав поисковых образов документов. Второй возможный способ заключается в том, что под аннотацию каждого документа отводится в k раз больше ячеек, чем под его поисковый образ (k=const). Причем, число k выбирается таким образом, чтобы объем памяти, отведенный под аннотацию документов, был достаточным для этой цели

Основной массив поисковых образов документов (ОМПОД) строится следующим образом. Поисковым образом каждого документа является некоторый набор дескрипторов. Эти дескрипторы, сгруппированные вместе, образуют так называемый узел, характеризующий данный документ. Номер документа (т. е. индекс в массиве документов) в явном виде не указывается. Он совпадает с номером (индексом) того члена данного узла, в котором находится его первый дескриптор. Каждый узел документов имеет вид последовательности четверок кодов, которую можно представить следующим образом:

$$\begin{array}{llll} 1 & a_{i_1} & b_{i_1} & c_{i_1} \\ 0 & a_{i_2} & b_{i_2} & c_{i_2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & a_{i_{k-1}} & b_{i_{k-1}} & c_{i_{k-1}} \\ 2 & a_{i_k} & b_{i_k} & c_{i_k} \end{array}$$

Рассмотрим отдельные составные части каждого узла. Цифра в первой колонке представляет собой некоторый признак, принимающий одно из трех значений: 1, 0, 2. Значение 1 указывает начальную строку узла некоторого документа, 2 — его последнюю строку, а значение 0 присутствует в каждой из промежуточных строк. Коды a_i и b_i, вместе взятые, образуют условный код некоторого дескриптора, индексирующего данный документ; a_i является номером зоны магнитной ленты; b_i — номером ячейки в этой зоне; c_i — адрес той ячейки в данной зоне (адрес связи), в которой в составе узла, ближайшего к данному, снова записан тот же дескриптор. Вся память, отведенная под ОМПОД, делится на зоны, в которых размещаются узлы, причем в каждой зоне должно быть размещено целое число узлов. Списки ОМПОД формируются в пределах одной зоны. Заметим, что один и тот же дескриптор может встречаться в списках разных зон.

Массив заголовков дескрипторов (МЗД) строится позонно с использованием сегментации. Элементами этого массива являются фиксаторы списков поисковых образов документов. Каждый фиксатор состоит из шести кодов и имеет вид

$$a_i\ b_i\ N_{ij}\ d_j\ e_j\ c'_j,$$

где a_i и b_i — условные коды некоторого (i-го) дескриптора; N_{ij} — число документов, включенных в j-й список этого дескриптора, d_j, e_j — номера лентопротяжного механизма и зоны, указывающие адрес списка; c'_j — адрес первого документа списка

В одной зоне массива заголовков могут оказаться фиксаторы различных списков для одного и того же дескриптора. Это обстоятельство определяет необходимость сегментации массива заголовков в соответствии с зонами ОМПОД

Таким образом, каждая строка МЗД является представителем некоторого дескриптора и связана с первой строкой, представляющей тот же дескриптор в зоне ОМПОД, отвечающей тому сегменту, который содержит упомянутую строку МЗД. Каждой зоне магнитной ленты, содержащей ОМПОД, соответствует сегмент в МЗД.

Массив заголовков дескрипторов используется при поиске. Основной массив дескрипторов (ОМД) является указателем начал ценных списков документов. ОМД в свою очередь имеет списковую структуру. В список объединены элементы, содержащие адреса отсылок к заголовкам одного и того же дескриптора. Сами дескрипторы в ОМД в явном виде не указываются. Они представлены в виде индексов (адресов) начал списков основного масива дескрипторов.

Остановимся на описании ОМД несколько подробнее. ОМД состоит из двух частей (см. рисунок). В первой части каждая ее строка соответствует определенному дескриптору и имеет связь:

а) со строкой МЗД, которая является первой из соответствующих тому же дескриптору;

б) со строкой, входящей во вторую часть ОМД, если упомянутому дескриптору в МЗД соответствуют строки, расположенные в разных сегментах. Во второй части ОМД каждая из строк соответствует одному из дескрипторов, который представлен либо строкой, содержащейся в первой части ОМД, либо строкой второй части ОМД, расположенной выше данной строки. Каждая строка второй части ОМД имеет связь а) со строкой некоторого сегмента в МЗД, отвечающей тому же дескриптору, и б) с нижерасположенной строкой второй части ОМД, если упомянутый дескриптор представлен строкой в некотором сегменте МЗД, с которой не связаны ни строка первой части ОМД, ни вышерасположенная строка второй части ОМД. Каждый элемент основного массива дескрипторов содержит адрес следующего элемента списка ОМД (e_1, c_1) и адрес заголовка дескриптора (e_2, c_2). Последний элемент каждого списка рассматриваемого массива отмечен признаком конца списка (КС).

Организация системы поиска. В соответствии с принятой организацией информационных массивов, процесс поиска документов состоит из четырех этапов. Допустим, что при поиске используется наиболее простой критерий смыслового соответствия — критерий на полное вхождение дескрипторов запроса в поисковый образ документа.

На первом этапе поиска происходит обращение по заданным дескрипторам запроса к ОМД. Результатом этого является определение общих сегментов в МЗД, содержащих заголовки для дескрипторов каждого запроса. Если для какого-либо запроса общего сегмента не окажется, запрос аннулируется. Запросы на первом этапе обрабатываются последовательно.

На втором этапе поиска происходит обращение к общим для дескрипторов запроса сегментам МЗД. Выбранные заголовки дескрипторов сортируются по признаку отношения к одной магнитной ленте. Затем в каждой группе заголовки сортируются по запросам. Запрос аннулируется, если в группе заголовков, характеризующих его, не окажется заголовка хотя бы одного дескриптора, принадлежащего данному запросу.

Назначение третьего этапа — определение общих зон в массиве ОМПОД для всех дескрипторов каждого запроса. Запрос аннулируется, если общих зон не окажется.

На четвертом этапе для каждого запроса среди выбранных зон ОМПОД определяется самый короткий для зоны список. Дальнейший поиск осуществляется просмотром таких списков.

Рассмотренный алгоритм поиска допускает одновременную обработку нескольких десятков запросов (число одновременно обрабатываемых запросов ограничено памятью машины и ее быстродействием). В таблице приведены результаты машинного эксперимента для случаев трех, пяти и десяти одновременно обрабатываемых запросов.

С увеличением количества одновременно обрабатываемых запросов абсолютное время поиска растет, а относительное — падает. Это дает основание считать, что при работе поисковой системы большого объема невыгодно обрабатывать одиночные запросы.

Расход времени ЦВМ «Минск-22» на информационный поиск

Количество запросов	3	5	10
Минуты	53	69	92

Важной особенностью данной организации дескрипторного словаря и поискового массива является то, что время поиска нужных документов практически не зависит от объема информационного массива.

УДК 681.3.065.4

ОРГАНИЗАЦИЯ ХРАНЕНИЯ И ПОИСКА ИНФОРМАЦИИ О ХИМИЧЕСКИХ СОЕДИНЕНИЯХ

С. К. КЕРИМОВ, А. А. МЕХТИЕВ

Использование наименований химических соединений в качестве дескрипторов в информационно-поисковой системе (ИПС) вызывает практически неразрешимые трудности, так как общее их количество исчисляется миллионами. Кроме того, в современной химической литературе не принята единая номенклатура, что также вызывает трудности в индексировании химических наименований.

Указанные затруднения могут быть устранены путем специального кодирования химических соединений. Одним из эффективных способов кодирования является использование так называемых линейных кодов. При этом каждая структурная формула химического соединения представляется в виде линейного кода, составляемого по строго определенным правилам. Линейный код представляет собой некоторую последовательность символов, комбинация которых не допускает двусмысленностей. При этом одному коду соответствует одна структурная формула.

В настоящее время в СССР и за рубежом разработаны различные системы кодирования химических соединений, которые связаны с отдельными, более или менее сложными, поисковыми задачами. Эти системы в большинстве случаев рассчитаны на информационный поиск в массивах, содержащих только перечень химических соединений.

Разрабатываемая в Научно-исследовательском институте технико-экономических исследований (НИИТЭХИМ) министерства химической промышленности СССР совместно с Азербайджанским институтом нефти и химии ИПС по химии и химической технологии рассчитана на поиск информации не только в списках химических соединений, но и в массивах информационных документов, в которых описываются химические соединения в более широком аспекте (описание свойств, методик получения, технологических процессов, способов переработки и т. д.). Поэтому данная ИПС предъявляет определенные требования к системе кодирования химических соединений, которые сводятся к следующему.

1. Код должен быть функцией дескриптора, т. е. содержать в себе необходимые иерархические и ассоциативные связи.

2. Код должен являться составной частью записанного с помощью дескрипторов поискового образа документа (ПОД) и запроса (ПОЗ).

3 Код должен обеспечить осуществление поиска как в массивах текстовых информационных документов, так и в списках структурных формул

Исходя из этих требований была разработана система кодирования органических и неорганических соединений и определены правила кодирования [1] *. Рассмотрим кратко эту систему

Для составления кода органического соединения сначала его структура разбивается на фрагменты, затем по правилам написания фрагментов составляется код:

структура⟶фрагмент⟶код

Фрагменты кода представляют собой линейную запись, состоящую из набора цифр, знаков и букв.

При составлении кода необходимо соблюдать:

— правила разбиения структуры соединения на фрагменты;

— порядок перечисления фрагментов в коде,

— порядок и правила написания самого фрагмента.

Эти три положения и отражаются в правилах кодирования. В общем код будет представлен последовательностью фрагментов.

I фрагмент, II фрагмент, III фрагмент и т. д.

Количество фрагментов в коде определяется структурой органического соединения.

Первый фрагмент кода любого соединения — это углеводородная часть брутто формулы соединения, записанная четырехзначным числом. Далее фрагменты перечисляются в таком порядке:

ациклические⟶циклические⟶гетероциклические⟶

связки между фрагментами

Функциональные группы указываются в соответствующем фрагменте.

Конец фрагмента фиксируется знаком ( ), а конец линейного кода (т. е. конец последнего фрагмента линейного кода) — признаком (:.).

Примеры записи химических соединений в виде линейных кодов:

1) Ацетон — CH_3COCH_3

Линейный код — 03 06 : A : 3 (2 (1) 1) - OF :.

Число атомов C; Число атомов H

2) Хлороформ — $CHCl_3$

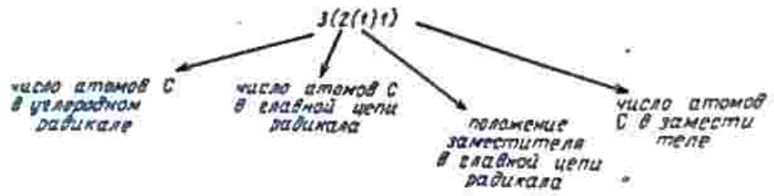
0101 : A : 1 - Hal cl, 3 :.

3) Толуол — $C_6H_5CH_3$

0708 : У : Бl alk 1 :.

Здесь У — углеводород, А — ациклический, Бl — бензольное кольцо, имеющее одно замещение, О — кислород, Hal — галогены, F — двойная связь.

Поясним строение связок в первом примере



*) В разработке системы кодирования органических и неорганических соединений принимали участие сотрудники НИИТЭХИМа И. М. Игнатьева, В. М. Брусникина, Л. В. Яковлева, Г. В. Сидорова и др.

Неорганические соединения кодируются путем последовательного описания элементов структурной формулы атомов соединений с указанием их числа и типов связей между ними, признаков классов соединений и т. д. Разработана система кодирования для следующих классов соединений неорганической химии: окислов, перекиси, надперекиси, оснований, гидридов, кислот, солей, комплексных соединений, интерметаллических соединений. Чтобы различить линейные коды органических соединений от неорганических, используется специальный признак.

В поисковый образ информационного документа или запроса может входить несколько линейных кодов. При этом каждый линейный код рассматривается как поддокумент (или подзапрос) и в памяти машины представляется как самостоятельная единица хранения информации.

Каждый поддокумент заканчивается девятизначным числом, включающим в себя, номер поддокумента — документа (7 цифр), грамматический признак (1 цифра) и признак органического или неорганического соединения (1 цифра)

В основу грамматики информационно-поискового языка (ИПЯ) разрабатываемой НПС положен принцип синтаксической классификации понятий (ключевых слов), входящих в состав ПОД или ПОЗ [2]. Естественный текст при переводе на ИПЯ представляется в виде ряда предложений (линейных записей), которые образуют ПОД.

В состав грамматического признака, в общем случае, входит номер категории и номер линейной записи, к которым относится данное понятие. Категории образуются на основе анализа и предметно-тематической классификации понятий ИПЯ. В данном случае все химические соединения относятся к одной категории — «вещества», и поэтому в качестве грамматического признака в линейных кодах указывается только номер линейной записи

Ниже показан пример записи документа № 0251719, имеющего три линейных кода:

I поддокумент 0918 : У : 6 (3) 2 (1) 1 (0) Fa :.010251719

II поддокумент 1112 : У : 2 |Ц62f.. Бla1| :.020251719

III поддокумент 1916 : Г3 (0) 1 alk 1 :.030251719

признак органики; грамматический признак; номер документа

Так как общее число символов (оно достигает 180), принятых для кодирования химических формул, превышает число различных знаков в коде М2 и в коде перфокарты (ПК), каждый символ линейного кода кодируется при помощи двух знаков в коде М2 или в коде ПК. Однако запись в памяти ЭВМ символов линейного кода двумя знаками кода М2 (или кода ПК) неэкономична и поэтому в памяти ЭВМ каждый символ линейного кода представляется трехзначным восьмеричным числом, называемым машинным кодом символа (примеры кодов даны в таблице). При этом способе (в случае использования ЭВМ «Минск-22») экономятся 3 бита на каждый записываемый символ.

Символ	I	I (индекс)	I	a	A	Na	P
Код М2	01	I +	I—	A<	AA	NA	P—
Машинный код	001	002	003	004	005	006	007

Линейные коды перфорируются и вводятся в ЭВМ в коде М2 или в коде ПК. При помощи специальной программы «Замена линейных кодов» символы линейных кодов заменяются на машинные коды.

Организация хранения документов с линейными кодами осуществляется прямым способом. Линейные коды (поддокументы) располагаются на магнитной ленте (МЛ) последовательно (без промежутков). Запись и поиск линейных кодов на МЛ происходит по зонам и поэтому расположение одного документа в двух зонах (а также на двух лентах) не допускается.

После записи каждого нового массива документов с линейными кодами определяется адрес свободной области (АСО) на МЛ для записи последующих массивов.

Запись на МЛ документов с линейными кодами осуществляется при помощи специальной программы «Запись линейных кодов».

Критерий смыслового соответствия (КСС) был сформулирован исходя из задач, стоящих перед машинным поиском информации и специфики информационных документов, заключающейся в наличии в ПОД информации о химических соединениях, представляемой в двух видах: линейными кодами и ключевыми словами (дескрипторами).

В общем случае, если в ПОД и ПОЗ содержатся и дескрипторная часть, и линейные коды, то документ считается отвечающим на запрос, если имеет место совпадение дескрипторных частей и линейных кодов, соответствующих химическим соединениям, указанным в ПОЗ и ПОД.

КСС для дескрипторных частей ПОЗ и ПОД и алгоритм его реализации подробно изложены в работе [3]. Мы рассмотрим этот вопрос по отношению к линейным кодам.

КСС линейных кодов ПОЗ и ПОД относится к типу критериев «на вхождение». К вложению линейных кодов ПОЗ в ПОД предъявляются следующие требования.

I. Полное совпадение фрагментов ПОЗ и ПОД или совпадение одного из фрагментов, представленных в ПОЗ с помощью логической операции *или*.

II. Отсутствие одного или нескольких фрагментов ПОЗ в ПОД (тех фрагментов ПОЗ, которые имеют отрицание *не*).

III. Вхождение фрагментов ПОЗ в ПОД в строго заданном порядке (фрагменты ПОЗ должны быть связаны при этом логической операцией *и*).

В зависимости от характера запроса фрагменты ПОЗ и ПОД считаются совпавшими, если:

1) фрагмент ПОЗ полностью входит во фрагмент ПОД, т. е. все символы и порядок их следования во фрагментах совпадают. Например.

ПОЗ	ПОД
DIII·FOO:,	3Fa:DIII·FOO:,

2) фрагмент ПОЗ частично входит во фрагмент ПОД. Это условие вводится в ПОЗ с помощью знака «~»*). Здесь возможны три случая:

ПОЗ	ПОД
а) ГЗ (0) ~ :	ГЗ (0) alk1 :
б) ~alk2Fa:	Blalk 2Fa:
в) ГЗ (0) ~·Hal cl:	ГЗ (0) lalk1·Hal cl:

Особенности и различия этих трех случаев рассматриваются ниже при описании алгоритма поиска.

*) Знак «~» имеет следующую интерпретацию: «допускается комбинация любых символов, в том числе и пусто».

Возможны различные сочетания указанных правил оценки соответствия ПОД и ПОЗ в едином сложном КСС, что обеспечивает гибкость разработанного метода поиска химических соединений. Результаты проведенных экспериментальных поисков показали достаточно высокую эффективность принятого КСС.

Алгоритм поиска химических соединений рассчитан на совместную обработку нескольких запросов за один прогон МЛ. Число линейных кодов в запросе и фрагментов в линейном коде не ограничено. При поиске в случае необходимости учитывается также совпадение грамматических признаков, приписанных линейным кодам ПОЗ и ПОД. Поиск по совместно обрабатываемым запросам осуществляется параллельно по всем зонам на МЛ. В пределах зоны по отдельным запросам поиск проводится последовательно в порядке их следования.

Поиск проводится в пять этапов.

I этап. Осуществляется проверка на совпадение первых фрагментов (углеводородных частей) ПОЗ и ПОД. При совпадении углеводородных частей ПОЗ и ПОД управление передается второму этапу поиска, при несовпадении осуществляется переход к рассмотрению следующего документа в данной зоне.

Отсутствие первого фрагмента в ПОЗ *) принимается как совпадение первых фрагментов ПОЗ и ПОД. Случай наличия первого фрагмента в ПОЗ и отсутствия его в ПОД принимается как несовпадение ПОЗ и ПОД. При проверке на совпадение первых фрагментов учитывается также возможность неполного совпадения первых фрагментов ПОЗ и ПОД, которая определяется знаком «~».

II этап. Проверка на совпадение последующих фрагментов линейного кода ПОЗ и ПОД. При их совпадении управление передается третьему этапу поиска, при несовпадении осуществляется переход к рассмотрению следующего документа из данной зоны и передача управления первому этапу поиска.

III этап выполняется при обнаружении признака «~» во фрагментах ПОЗ. Как было отмечено выше, при этом возможны три случая.

а) Признак «~» встречается в начале фрагмента. В этом случае для совпадения фрагментов требуется наличие во фрагменте ПОД последовательности символов ПОЗ, стоящих после признака «~». В случае совпадения фрагментов ПОЗ и ПОД осуществляется переход к рассмотрению следующего фрагмента ПОЗ и передача управления второму этапу поиска. При несовпадении — переход к рассмотрению следующего фрагмента ПОД и передача управления второму этапу.

б) Признак «~» встречается в середине фрагмента. При этом для совпадения рассматриваемых фрагментов ПОД и ПОЗ требуется наличие во фрагменте ПОД последовательности символов ПОЗ, расположенных перед признаком «~» и после признака «~». В случае совпадения осуществляется переход к рассмотрению следующего фрагмента ПОЗ и передача управления второму этапу поиска, иначе — переход к рассмотрению следующего фрагмента ПОД и передача управления второму этапу.

в) Признак «~» встречается в конце фрагмента ПОЗ. В этом случае проверка на совпадение символов во фрагментах ПОЗ и ПОД производится до появления признака «~» в рассматриваемом фрагменте ПОЗ. При совпадении фрагментов ПОЗ и ПОД осуществляется переход к рассмотрению следующего фрагмента ПОЗ и передача управления второму этапу, иначе — переход к рассмотрению следующего фрагмента ПОД и передача управления второму этапу поиска.

*) В этом случае на месте первого фрагмента указывается знак «~ ~».

IV этап выполняется при появлении признака конца фрагмента ( ) Проверяется наличие признаков, указывающих на наличие логических операций *или, не, и*.

а) При обнаружении признака *или* если произошло совпадение предыдущих сравниваемых фрагментов ПОЗ и ПОД, то пропускается фрагмент, стоящий за признаком *или* и проверяется наличие в ПОЗ других операций типа *и, не*, в противном случае в качестве очередного сравниваемого фрагмента ПОЗ берется фрагмент, стоящий за признаком *или* и управление передается второму этапу.

б) При обнаружении признака *не* формируется признак, указывающий на специальный режим проведения второго этапа поиска и управление передается второму этапу: при положительном ответе второго этапа рассматриваемый ПОД считается не удовлетворяющим данному ПОЗ, в противном случае осуществляется переход к рассмотрению следующих фрагментов ПОЗ

в) При обнаружении признака *и* в качестве очередного проверяемого фрагмента ПОЗ принимается фрагмент, стоящий за признаком *и*, управление передается второму этапу поиска. При этом проверка ПОД на втором этапе начинается после предыдущего совпавшего фрагмента ПОД.

V этап обеспечивает отбор документов с учетом грамматики, он выполняется после завершения работы I—IV этапов для всех совместно обрабатываемых запросов. Сначала устанавливается необходимость проведения поиска с учетом грамматики и затем осуществляется проверка на совпадение грамматических признаков, приписанных линейным кодам ПОЗ и ПОД. Далее выполняется процедура объединения ответов, полученных при поиске по линейным кодам (подзапросам), т. е. выявление документов, имеющих одинаковые номера.

Полученные номера документов выводятся на печать в упорядоченном виде и (или) в случае необходимости запоминаются для сопоставления с ответами поиска по дескрипторам, и выдачи общих ответов.

Эта необходимость устанавливается заранее (до проведения поиска) в зависимости от требуемого режима поиска; данный режим поиска задается с помощью ключа на пульте управления ЭЦВМ.

В общем случае в разрабатываемой ИПС по химии и химической технологии предусматриваются три режима поиска

I. Режим поиска документов по ключевым словам (дескрипторам). В этом случае в ПОЗ содержатся только ключевые слова (свойства веществ, процессы, методы и средства получения, характеристики оборудования, методы анализа и т. д.) с приписанными им грамматическими признаками. Поиск проводится в массиве документов с ключевыми словами. При этом в зависимости от характера запроса и требований к выдаче поиск можно произвести а) без учета грамматических признаков, б) с учетом грамматических признаков.

Результатами поиска в режиме I являются документы (или их номера), в которых говорится о веществах, процессах, оборудованиях, свойствах и т. д., описанных ключевыми словами (дескрипторами) ПОЗ.

II. Режим поиска документов по линейным кодам. При этом в ПОЗ содержатся только линейные коды химических соединений с приписанными им грамматическими признаками. Поиск проводится в массиве документов с линейными кодами. Здесь, как и в режиме I, предусматривается возможность проведения поиска без учета и с учетом грамматики.

Результатами поиска в режиме II являются документы (или их номера), в которых описываются свойства, методы и средства получения и т. д. веществ, представленных линейными кодами ПОЗ.

III. Режим совместного поиска документов по ключевым словам и по линейным кодам. В этом случае в ПОЗ содержатся и ключевые слова, и линейные коды. Запрос разбивается на две части: дескрипторную и кодовую, и поиск идет в две стадии по ключевым словам (дескрипторам) и по линейным кодам. По ним проводятся автономные поиски и запоминаются полученные списки ответов. После чего из этих двух списков выделяются общие ответы на одинаковые запросы.

Работа по реализации поиска информации о химических соединениях на ЭВМ «Минск-22» была начата в 1969 г. Введено в ИПС около 14 000 химических соединений. Среднее время ввода, затрачиваемое на замену символов линейного кода на машине и запись на МЛ 150 химических соединений (это число обусловлено емкостью ОЗУ), составляет 2 минуты.

Объем программ «Замена линейных кодов» и «Запись линейных кодов» — 2000_8 команд.

Программа поиска рассчитана на одновременную обработку 16 запросов. Среднее время поиска на один запрос в массиве 14 000 соединений (40 зон МЛ) составляет 4 минуты.

Объем программ поиска — 2000_8 команд.

По проведенным экспериментальным поискам по 100 запросам получены следующие показатели системы: полнота 75%, точность 80%.

Приведенные данные показывают на достаточно высокую эффективность разработанной специализированной ИПС по химии и химической технологии. Принципы построения данной ИПС, сочетающей дескрипторный принцип со специализированным символьным языком линейных кодов, могут быть использованы при разработке других фактографических ИПС.

ЛИТЕРАТУРА

1. «Разработка методики прямого кодирования химических соединений». В сб. «Вопросы разработки механизированной информационно-поисковой системы для центрального справочно-информационного фонда по химии и химической промышленности», вып. 10. НИИТЭХИМ, 1968.
2. Иванова Н. И., Маргаритов В. Б. Введение элементов грамматики в дескрипторный информационно-поисковый язык по химии и химической промышленности. НТИ, 1968, сер. 2, № 10, стр. 20—27.
3. «Алгоритмы поиска». В сб. «Вопросы разработки механизированной информационно-поисковой системы для ЦСНФ по химии и химической промышленности», вып. 17, 1969.
4. Керимов С. К., Иванова Н. И. Машинная реализация ИПС «Аргон-1». НТИ, 1970, сер. 2, № 3, стр. 25—27.

УДК 681.3.06

ОБ ОДНОЙ ПРОГРАММЕ АНАЛИЗА ГРАФОВ

М. Г. ГААЗЕ-РАПОПОРТ, С. Н. ЛУКИАНОВА

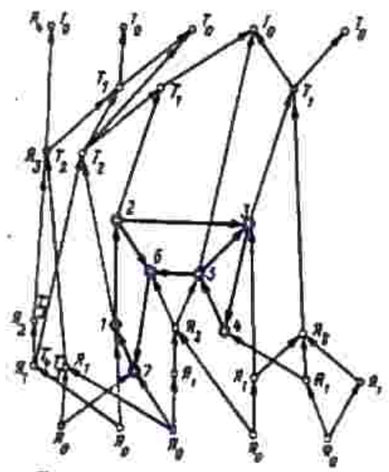
В разнообразных практических задачах часто приходится иметь дело с анализом ориентированных графов. Такие графы строятся, например, для решения задач сетевого планирования и управления, к их анализу сводится ряд задач организации многопроцессорной обработки больших потоков информации, а также многие задачи исследования документооборота на промышленных предприятиях, где документы изображают в виде вершин, а ребра указывают связи между документами, показывая, какие документы используются при составлении рассматриваемого.

В перечисленных задачах приходится иметь дело с большими графами, содержащими, обычно, несколько тысяч ребер. Вычерчивание таких графов и их визуальный анализ — весьма сложные процессы и требуют большого времени. С целью упрощения анализа графов их часто

представляют в так называемой ярусно-параллельной форме (ЯПФ), когда каждая вершина данного яруса может являться концом только тех ребер, которые начинаются в предыдущих ярусах, причем хотя бы одно из этих ребер начинается вершиной из непосредственно предыдущего яруса.

Одной из существенных задач анализа ориентированных графов является также отыскание в них логических ошибок, проявляющихся в виде замкнутых циклов. Отыскание циклов в больших и сложных графах также является весьма трудоемким процессом, требующим большого перебора.

В связи с трудностями ручного анализа графов возникла задача представления их в ЯПФ с выделением «остатка», содержащего циклы, с помощью ЭЦВМ.



Пример выполнения алгоритма.

Граф задается в виде таблицы пар чисел (a_i, b_i), соответствующих ребрам; первое число пары a_i является номером начальной, а второе b_i — номером конечной вершины ребра.

Для реализации сформулированной выше задачи на ЭВМ, т. е. для представления заданного таким образом графа в ЯПФ с выделением «остатка», содержащего циклы, был составлен алгоритм, идея которого заключается в следующем.

Первым этапом является поиск входных вершин (начал), осуществляемый путем просмотра всех начал ребер (элементов первого столбца) и проверки их присутствия в столбце окончаний. При просмотре отмечаются те начальные вершины, которых не оказалось в столбце окончаний, т. е. для которых не существует предшествующих. Отмеченные вершины образуют список входных вершин.

После отыскания входных вершин осуществляется поиск вершин первого яруса. Для этого просматриваются и отмечаются те вершины из второго столбца, которым предшествуют только отмеченные перед этим (начала). Отмеченные при этом концы ребер образуют список вершин первого яруса.

Для отыскания следующего яруса процедура повторяется, и так до тех пор, пока либо не окажутся отмеченными все вершины, либо очередной ярус не окажется пустым. Если в результате все вершины оказались отмеченными — задача решена, т. е. выделены все ярусы, и «остатка», содержащего циклы, у графа нет.

Если же после окончания поиска ярусов остались неотмеченные вершины, процедура повторяется в обратном порядке, т. е. отмечаются и ищутся «тупики» — те вершины, которые не являются началом ни для одного ребра. После отыскания «тупиков» (выходов графа) отыскиваются «тупики» первого ранга, т. е. те, за которыми следуют только отмеченные вершины. Процедура повторяется до тех пор, пока число «тупиков» очередного ранга не окажется равным нулю.

Вершины, оставшиеся неотмеченными, образуют «остаток» графа, т. е. часть, содержащую по крайней мере один цикл. Легко видеть, что любой маршрут в «остатке» всегда приводит к циклу, который отыски-

вается путем последовательного прохода ребер, начиная с произвольной вершины «остатка» до первого повторения какой-либо из вершин.

Схема счета [1], соответствующая описанному алгоритму, имеет следующий вид:

$$\prod_{(i)} A_{1,i} K_{1,i} \prod_{(\lambda)} \prod_{(k)} (A_{2k\lambda} K_{2k\lambda})\, D_1 \prod_{(j)} (B_{1,j} H_j)\, D_2 \prod_{(s)} \prod_{(l)} (B_{2ls} H_{ls}) \prod_{(s)} C_{1,s} D_3.$$

Здесь $A_{1,i}$ — блок (оператор), осуществляющий проверку элемента a_i на вхождение в $\mathfrak{B}$ и выделение элемента $a_i^0 (a_i^0 \in \mathfrak{A},\ a_i^0 \notin \mathfrak{B})$,
где $\mathfrak{A} = \{a_n\}$ — множество всех начал ребер графа,
$\mathfrak{B} = \{b_n\}$ — множество всех концов ребер;
$K_{1,i}$ — оператор, отмечающий выделенные элементы $(a'_i \in \mathfrak{A})$,
$A_{2k\lambda}$ — блок, осуществляющий для каждого b_λ $(b_\lambda \in \mathfrak{B})$ выделение элементов b_{n_k} $(b_{n_k} \in \mathfrak{B},\ b_{n_k} = b_\lambda)$. Для выделенных элементов $\{b'_{n_k}\}$ отыскивается множество начал $\{a'_{n_k}\}$. Если $\{a_{n_k}\} \subset \{a'_i\}$, элемент b_λ выделяется. Блок просматривает все значения $\lambda = \overline{(1, N)}$ последовательно;
$K_{2k\lambda}$ — блок, отмечающий символом k все выделенные элементы b_λ и все элементы $a_i \in \mathfrak{A}$, такие, что $a_i = b_\lambda$. Блоки $A_{2k\lambda}$ и $K_{2k\lambda}$ работают до тех пор, пока либо во множестве $\mathfrak{A}$ не останется неотмеченных элементов, либо пока после цикла работы блоков $A_{2k\lambda}$ и $K_{2k\lambda}$ число отмеченных элементов во множестве $\mathfrak{A}$ не изменится;
D_1 — блок, печатающий пары, соответствующие ребрам, оканчивающимся в i-м ярусе;
$B_{1,j}$ — блок отыскания тупиков, осуществляющий поиск и выделение из $\{b_j\}$ всех $b_s (b_s \in \mathfrak{B};\ b_s \notin \mathfrak{A})$;
H_j — блок, отмечающий связи, предшествующие выделенным тупикам;
D_2 — блок, печатающий тупики;
B_{2ls} — блок, выделяющий тупики l-го ранга, начиная с $l=1$ из связей, оставшихся неотмеченными;
H_{ls} — блок, поярусно отмечающий связи, относящиеся к выделенным тупикам;
$C_{1,s}$ — блок, выделяющий цикл из оставшихся связей;
D_3 — блок, печатающий «остаток» графа, выделенный цикл и относящиеся к нему связи.

Блоки B_2 и H_2 работают до тех пор, пока не перестанут выделяться новые ранги тупиков.

Работа описанного алгоритма поясняется на графическом примере, изображенном на рисунке, на котором приняты следующие обозначения: R_0 — входные вершины графа, R_i — вершины, принадлежащие i-му ярусу; T_0 — «тупики», T_i — «тупики» i-го ранга.

В результате выполнения алгоритма выделяется «остаток» графа, содержащий два цикла. При начале поиска цикла из вершин 1, 2, 6, 7 — может быть выделен любой из двух циклов, из вершин 3, 4 и 5 — только один цикл (треугольный).

Описанный алгоритм был реализован на ЭВМ «Урал-14» и использовался для решения задач анализа документооборота.

Составленная по описанному алгоритму программа позволяет анализировать графы, содержащие до 2000 ребер. Программа печатает в формате нормальных страниц списки входов, поярусные списки ребер, оканчивающихся в данном ярусе, тупики, ребра, образующие «остаток» графа, один выделенный цикл и списки ребер, оканчивающихся в вершинах цикла и исходящих из его вершин.

Программа содержит 1757 одноадресных команд, из которых 1057 потребовалось для организации печати. Наибольший анализируемый

граф содержал 1473 ребра, при этом для решения задачи потребовалось около 60 минут.

Программу можно получить у авторов.

ЛИТЕРАТУРА

1. Ляпунов А. А. О логических схемах программ. В сб. «Проблемы кибернетики», вып. I. Физматгиз, 1958.

УДК 681.3.06

О СОСТАВЕ ОПЕРАЦИЙ И СТАТИСТИКЕ ИХ ИСПОЛЬЗОВАНИЯ В ПРОГРАММАХ УПРАВЛЯЮЩИХ ЦВМ

В. В. ЛИПАЕВ, К. К. КОЛИН

1. ПОСТАНОВКА ЗАДАЧИ

В настоящее время известна статистика использования различных типов команд в программах универсальных ЦВМ при решении инженерных и научных задач [1, 2]. Однако сведения о частоте использования различных типов операций в программах управляющих ЦВМ и рекомендации по ориентированию их системы команд практически полностью отсутствуют. Анализ такого рода статистических данных для реальных систем позволяет более обоснованно подойти к проектированию структуры управляющих ЦВМ и к оптимизации систем команд ЦВМ этого класса, что особенно важно при жестких ограничениях на возможные характеристики ЦВМ [3].

Ниже приводятся данные о составе операций в программах, составленных для четырех ЦВМ, использующихся в сложных автоматизированных системах управления и работающих в реальном масштабе времени. Три из этих ЦВМ являются одноадресными с фиксированной запятой и имеют раздельные накопители для хранения программ и числовой информации. Накопители программ и чисел в этих машинах односторонние и допускают непосредственное обращение к любой ячейке. Первая машина (ЦВМ-1) имеет накопитель программы объемом порядка 10 тыс. команд. Эта программа почти полностью использовалась при обработке информации и лишь в небольшой степени — для функций управления объектами. Вторая машина (ЦВМ-2) имеет память программы объемом около 100 тыс. команд, которая практически полностью используется. Программа служит для обработки информации и управления реальными объектами, но с преимуществом (по объему программ) задач управления.

Объем анализируемой части программы третьей машины (ЦВМ-3) составляет порядка 20 тыс. команд. Эта программа предназначена, в основном, для решения задач обработки информации, поступающей от внешних абонентов. Система команд ЦВМ-3 содержит ряд специальных команд для обработки малоразрядной упакованной информации. Эта машина в наибольшей степени из всех других ЦВМ приспособлена для решения информационно-логических задач. Четвертая машина (ЦВМ-4) представляет собой трехадресную универсальную ЦВМ М-220, которая несколько конструктивно видоизменена с целью обеспечения взаимодействия с внешними абонентами и работы в реальном масштабе времени. Программа этой машины имеет объем 20 тыс. команд и предназначается для обработки большого объема информации.

Информацию, обрабатываемую в указанных выше программах, можно разделить на два типа: квантованные результаты измерения непрерывных величин с точностью 0,1—0,01% (10—14 разрядов) и малоразрядные информационные признаки (1—6 разрядов).

Состав операций в анализируемых программах производился статически, т. е. путем непосредственного подсчета количества различных типов команд в программе без учета частоты их использования при функционировании системы, а анализ различных типов команд во второй программе производился также динамически в процессе ее работы. Оказалось, что во второй программе характер распределения наиболее употребительных операций, полученный при статическом анализе, близок к характеру их распределения, полученному при динамическом анализе. При этом в случае динамического анализа происходит лишь некоторое повышение частоты использования условных переходов, операций, связанных с использованием индексных регистров и, в особенности, операций, изменяющих значение этих регистров. Кроме того, удельный вес длинных операций умножения и деления несколько снижается, так как при программировании обычно стремятся выносить эти операции из цикла.

2. АНАЛИЗ СТАТИСТИЧЕСКИХ ДАННЫХ О СОСТАВЕ ОПЕРАЦИЙ В ПРОГРАММАХ УПРАВЛЯЮЩИХ ЦВМ

Для проведения анализа операций в программах все виды команд были разделены на шесть функциональных групп следующим образом (табл. 1).

— арифметические операции,
— логические операции,
— операции пересылки данных между памятью ЦВМ и арифметическим устройством,
— переходы;
— операции с индексными регистрами;
— прочие операции (обмен ЦВМ с внешними устройствами, контрольные и вспомогательные операции).

Внутри каждой группы различные модификации однотипных команд объединялись и учитывались суммарно. Результаты исследований трех первых программ, составленных для одноадресных ЦВМ, приведены в табл. 1, а данные для программы, составленной для трехадресной машины, представлены в табл. 2. При динамическом анализе второй программы было исследовано несколько миллионов операций в процессе функционирования этой программы в системе управления. Подсчет операций производился специальной подпрограммой, включаемой через систему прерывания. При этом не учитывались операции, выполняемые в программе-диспетчере операционной системы ЦВМ.

Арифметические операции во всех исследованных программах составляют небольшую долю (8—12%). Наиболее употребимыми в этой группе являются операции сложения и вычитания, которые составляют 6—11% от общего числа команд в программах. В то же время длительные по времени выполнения операции умножения и деления в программах встречаются довольно редко. Их доля составляет всего 0,93% в четвертой программе. Для второй и третьей программ эта доля равна соответственно 2 и 2,5%. Правда, в третьей программе процент этих операций оказывается равным 6,2%. При этом частота использования наиболее длительной по времени выполнения операции деления не превышает 0,5%. Эти данные резко отличаются от данных о частоте использования операции деления в вычислительных задачах [1, 2].

В системе команд одноадресных ЦВМ, на которых выполнялись первая и третья программы, имелись специальные команды для выполнения арифметических операций над частями слов и выполнения действий над малоразрядной упакованной информацией. Этим в определенной степени объясняется некоторое повышение процента арифме-

Таблица 1

Процентный состав операций в программах одноадресных управляющих ЦВМ

Номер группы	Наименование группы	Тип операции	Процент использования в программе			
			ЦВМ-1	ЦВМ-2	ЦВМ-3	ЦВМ-2 (в дополнение)
I	Арифметические операции	Сложение	4,6	2,74	4,69	2,75
		Вычитание	4,9	3,43	4,26	3,47
		Умножение	1,6	1,98	4,54	1,44
		Деление	0,32	0,5	0,69	0,23
		Всего	11,4	8,65	14,17	7,88
II	Логические операции	Сравнение	1,9	2,55	1,02	4,52
		Выделение	6,0	7,12	3,32	8,52
		Формирование	2,1	2,82	2,30	1,52
		Сдвиг	3,5	2,25	2,70	1,77
		Всего	13,5	14,74	9,34	16,33
III	Операции с сумматором АУ	Посылка в сумматор из ОЗУ	24,3	21,2	21,6	19,1
		Запись сумматора в ОЗУ	17,7	16,6	17,0	12,3
		Всего	42,0	37,8	38,6	31,4
IV	Переходы	Условные переходы	10,3	8,0	7,0	14,45
		Безусловные переходы	12,9	10,2	8,7	6,93
		Всего	23,2	18,2	15,7	20,38
V	Операции с индексными регистрами	Обмен с индексными регистрами	7,6	9,50	6,62	9,8
		Изменение индексных регистров	1,4	2,0	1,98	3,8
		Всего	9,0	11,5	8,60	13,6
VI	Прочие операции	Прием и выдача информации	0,7	2,84	1,0	2,67
		Контрольные операции	—	3,2	9,1	2,4
		Остальные операции	0,2	3	3,46	5,4
		Всего	0,9	10,1	13,56	10,47

тических операций в первой и третьей программах и соответствующее снижение логических операций по сравнению со второй программой. Заметим, что доля арифметических операций над частями слов в первой программе составляет всего 1,2%

Логические операции во всех программах, кроме третьей, встречаются чаще, чем арифметические операции. Доля этих операций в общем объеме программ одноадресных ЦВМ составляет 10—15%, а в программе для трехадресной ЦВМ удельный вес этих операций достигает 39%. Такой значительный вес логических операций обусловлен в последней программе, в основном, частым использованием операций выделения и формирования кодов (логического умножения и логического сложения) и объясняется тем, что во всех ЦВМ применялась плотная упаковка малоразрядной информации в пределах одной ячейки оперативной памя-

Таблица 2

Процентный состав операций в программах трехадресных ЦВМ

Номер группы	Тип операции	Процент использования в программе	
		ЦВМ-1 (М-220)	М-20 (данные [1])
I	Сложение и вычитание	11,0	21,0
	Умножение	0,84	21,0
	Деление	0,09	2,0
	Всего	11,93	44,0
II	Логические операции	39,3	17,6
III	Пересылки	7,1	4,8
IV	Условные и безусловные переходы	24,2	25,6
V	Изменение регистра адреса	15,4	2,8
VI	Прочие операции	2,07	5,2

ти, особенно в зонах длительного хранения данных, а также в сообщениях, предназначенных для обмена ЦВМ с внешними абонентами. В программах для ЦВМ-1 и ЦВМ-3 операции выделения и формирования имеют значительно меньший вес, чем в программе для ЦВМ-2, так как в этих машинах была предусмотрена возможность выполнения операций выборки и засылки информации части слова (около 10%)

Операции пересылки данных между арифметическим устройством (АУ) и памятью ЦВМ наиболее часто встречаются во всех программах одноадресных ЦВМ. Процент использования этих операций очень высок и составляет 38—42%, с преобладанием операций посылки чисел в сумматор арифметического устройства. При этом в первой программе почти четверть операций этой группы (10%) являются операциями над частью слова, что также является причиной снижения общего количества логических операций

Высокий удельный вес операций этой группы объясняется тем, что в одноадресных ЦВМ при выполнении большинства арифметических и логических операций приходится использовать отдельные команды для запоминания и чтения промежуточных результатов. В программе же трехадресной ЦВМ операции пересылки составляют всего 7%.

Переходы. Порядка 15—25% всего объема исследованных программ составляют команды передачи управления. В среднем после каждых четырех команд другого типа в программе следует условный или безусловный переход. При этом подавляющая часть этих команд служит для реализации логических разветвлений вычислительного процесса. Определенное количество (~2—4%) безусловных переходов использовалось для реализации «заплаток» в программу, внесенных в процессе ее отладки.

Таким образом, в среднем не менее чем на каждые 10 команд программы приходится одно разветвление, связанное с анализом обрабатываемой информации и принятием решения. При динамическом исследовании второй программы установлено, что удельный вес переходов в программе возрастает до 20% за счет более частого использования условных переходов при реализации циклов программы.

Операции с индексными регистрами в исследованных программах составляют 8—15%, т. е. имеют примерно такой же вес, что и арифметические операции. Эти операции широко используются при организации

шиклов программы, а также при декодировании информации с использованием регистров индексной системы. Подавляющую часть операций этой группы (6—10%) составляют операции заполнения индексных регистров и запоминания их содержимого в оперативной памяти. Команды же, связанные с арифметическими операциями над содержимым индексных регистров, сравнительно редки (1,5—2,0% от всего объема программы).

Прочие операции занимают в программах ЦВМ-1 и ЦВМ-1 порядка 1—2%, а в программе ЦВМ-2 и ЦВМ-4 — порядка 10%, так как последние ЦВМ имеют более развитую систему контроля и обмена информацией с внешними абонентами. Следует ожидать, что это соотношение будет справедливо и при динамическом анализе работы программы, так как в процессе функционирования осуществляется интенсивный обмен данными с внешними абонентами, пультами операторов и другими специализированными устройствами.

3. СОПОСТАВЛЕНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ С РЕЗУЛЬТАТАМИ ИССЛЕДОВАНИЯ ПРОГРАММ УНИВЕРСАЛЬНЫХ ЦВМ

В работе [1] приводятся достаточно подробные статистические данные о составе операций в четырех различных программах, разработанных для универсальной ЦВМ М-20. Две из этих программ предназначались для решения задач газодинамики, третья программа применялась для решения системы эллиптических уравнений, а четвертая программа использовалась для трансляции с языка АЛГОЛ. Усредненные статистические данные по составу основных операций для этих программ приведены в табл. 2 и позволяют сопоставить эти данные с аналогичными данными для программы ЦВМ-4. Однако при этом необходимо учитывать, что приведенные в работе [1] данные получены путем моделирования выполнения программы на ЦВМ, и поэтому отражают динамику использования различных операций, в то время как данные для ЦВМ-4, приведенные в статье, являются статическими. Это различие, на наш взгляд, не является принципиальным, так как приведенные в табл. 1 данные для программы ЦВМ-2 показывают, что для управляющих ЦВМ процесс исполнения программы хотя и несколько изменяет процентные соотношения между различными типами операций, однако соотношения между наиболее важными группами операций, в основном, сохраняются. Авторы считают, что аналогичная закономерность окажется справедливой и для трехадресных ЦВМ.

Важным фактором, который необходимо учитывать при сопоставлении данных табл. 1 и 2, является различная адресность ЦВМ. Вследствие того, что ЦВМ М-20 и М-220 являются трехадресными, число операций пересылки кодов между ОЗУ и сумматором арифметического устройства в программах этих ЦВМ значительно меньше числа этих операций в программах одноадресных ЦВМ.

Сопоставление приведенных в табл. 2 данных о составе операций в программе ЦВМ-4 с аналогичными данными для ЦВМ М-20 показывает, что процент использования арифметических операций в программе управляющей ЦВМ-4 (~12%) значительно ниже, чем в программах универсальной машины (~44%). При этом операции умножения и деления используются в программе ЦВМ-4 примерно в 20 раз реже, чем в программе М-20 при решении задач вычислительного характера. В то же время число логических операций, используемых в управляющей ЦВМ-4, превосходит более, чем в 2 раза число этих операций в вычислительных задачах. В программах управляющих ЦВМ также интенсивно используются индексные регистры, в частности, при организации циклов

и организации переадресаций. В программах вычислительного характера процент использования этих операций оказывается в несколько раз меньшим.

4. НЕКОТОРЫЕ ВЫВОДЫ И РЕКОМЕНДАЦИИ

Особенности систем команд управляющих ЦВМ. Приведенные выше статистические данные о распределении различных типов операций в программах управляющих ЦВМ, позволяют сделать некоторые выводы и рекомендации о составе систем команд управляющих ЦВМ. Заметим, что в каждом конкретном случае общие рекомендации следует применять с учетом специфики управляющих систем и сложности их технической реализации.

Приведенное процентное распределение операций в программе (см. табл. 1) достаточно характерно для сложных систем, обрабатывающих информацию и управляющих реальными объектами. В этом убеждает близость между частостью использования отдельных типов команд в ЦВМ-1, ЦВМ-2 и ЦВМ-3, хотя эти машины предназначались для решения различных задач, имели разные системы команд и различную разрядность программной и числовой информации.

Поскольку частота использования отдельных типов операций в управляющих и вычислительных программах существенно различна (см. табл. 2), то следует считать целесообразным создание в управляющих ЦВМ систем команд, специально ориентированных на решение информационно-логических задач. Так, например, сравнительно большая доля операций упаковки и распаковки данных свидетельствует о необходимости иметь специальные команды, позволяющие непосредственно оперировать с малоразрядной упакованной информацией. Оценки, проведенные авторами, убеждают в том, что в случае наличия в ЦВМ специальных команд для упаковки и распаковки информации объем программ мог бы быть уменьшен в среднем на 20%.

Большой удельный вес логических операций в программах ЦВМ, выполняемых над малоразрядной (1—6 разрядов) информацией различного состава, позволяет надеяться, что расширение перечня команд, работающих с частью слова, в ЦВМ-1 свыше имевшихся пяти типов применительно к рассматриваемой системе могло бы повысить эффективное быстродействие этой машины еще на 10—15%. При этом основная доля отмеченного выигрыша должна получиться за счет операций упаковки и распаковки информации. В то же время вряд ли имеет смысл вводить арифметические операции с частью слова, вследствие их малого удельного веса (только 1,2% арифметических операций выполнялись в ЦВМ-1 с частью слова).

Использование констант. В программах управляющих ЦВМ очень часто встречаются операции, в которых используются константы. Примерами таких операций являются логические операции (сравнение, выделение, формирование и сдвиг), арифметические операции со счетчиками или с использованием масштабных коэффициентов, а также операции посылки кодов на некоторые регистры машины. Большинство констант являются многократно употребляемыми. В программе ЦВМ-1 таких констант насчитывается порядка 700, а в программе ЦВМ-2 — около 2000. Несмотря на то, что общее количество констант в программах сравнительно невелико и составляет обычно 5—8% от ее объема, количество же операций в программе, обращающихся к константам, значительно больше и достигает 20% (19,7% для ЦВМ-1). При использовании ЦВМ с возможностью совмещенных обращений к различным блокам запоминающего устройства и при размещении программ и констант в одном блоке быстродействие машины за счет обращений к константам снижается на 15—20% по сравнению с быстродействием, достигаемым при

размещении программ и констант в разных блоках. Отсюда следует, что для повышения быстродействия управляющих ЦВМ необходимо применять в них такие аппаратные решения, которые позволяли бы сократить дополнительные затраты времени на обращение к константам.

ЛИТЕРАТУРА

1. Штаркман В. С., Кузнецова Т. П. Результаты статистического исследования программ на машине М-20. В сб. «Вопросы технической эксплуатации вычислительных машин». М., 1967, вып. 1, стр. 54—75 (ВЦ АН СССР).
2. Благовещенский Ю. В., Попов Б. А., Сергиенко И. В., Тесслер Г. С. Исследование вычислительного процесса на ЭВМ типа «Промінь» и пути повышения эффективности быстродействия этого класса машин. «Кибернетика», 1970, № 1.
3. Малиновский Б. Н., Янович И. А., Египко В. М., Карташев В. И., Слободянюк Т. Ф., Забара С. С., Лучук А. М. Основы проектирования управляющих машин промышленного назначения. Под ред. Малиновского Б. Н. Изд-во «Машиностроение», 1969.

УДК 681.306.22

СИСТЕМА АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ И ВЫПУСКА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ НА ПРОГРАММУ ДЛЯ УПРАВЛЯЮЩИХ ЦВМ (ЯУЗА-1)

Л. А. СЕРЕБРОВСКИЙ, Л. Г. СИБИРЯКОВ, Н. Е. ЛИПЕЦ, Л. А. ПАНОВА

ЯУЗА-1 представляет собой систему автоматизации программирования для управляющих ЦВМ. Система реализована на универсальной вычислительной машине М-220. С ее помощью обеспечивается:

— трансляция программ с универсального автокода, разработанного специально для определенного класса управляющих ЦВМ;

— автоматическая стыковка отдельных подпрограмм в единую программу большого объема (порядка сотен тысяч команд);

— работа с переменными, упакованными по нескольку в одной ячейке и сгруппированными в иерархические структуры;

— автоматический выпуск полной документации на программу и перфокарт ввода (прошивки) программы в управляющую ЦВМ;

— корректировка программы и документации (в частности, по результатам отладки);

— простая настройка на логику конкретной управляющей ЦВМ из достаточно широкого класса.

Общая функциональная схема системы ЯУЗА-1 приведена на рис. 1. Эта система является составной частью, находящейся в процессе создания, более мощной системы автоматизации программирования ЯУЗА, которая предназначается для автоматизации всего процесса разработки и отладки алгоритмов и программ для управляющих ЦВМ.

1. ОПИСАНИЕ ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ И СОСТАВЛЕНИЕ БИБЛИОТЕКИ ОПИСАНИЯ ДАННЫХ

Обычно программа для управляющих ЦВМ использует большое число глобальных переменных (до нескольких тысяч). При этом имеется сильная взаимосвязь отдельных подпрограмм по этим переменным, т. е. как правило, одна и та же переменная используется большим числом подпрограмм. Это обстоятельство определило необходимость выделения специального этапа в разработке программы — этапа описания и распределения памяти глобальных переменных и составления БИБЛИОТЕКИ ОПИСАНИЙ. Это позволило исключить необходимость для каждой транслируемой подпрограммы описывать уже включенные в БИБЛИОТЕКУ глобальные переменные.

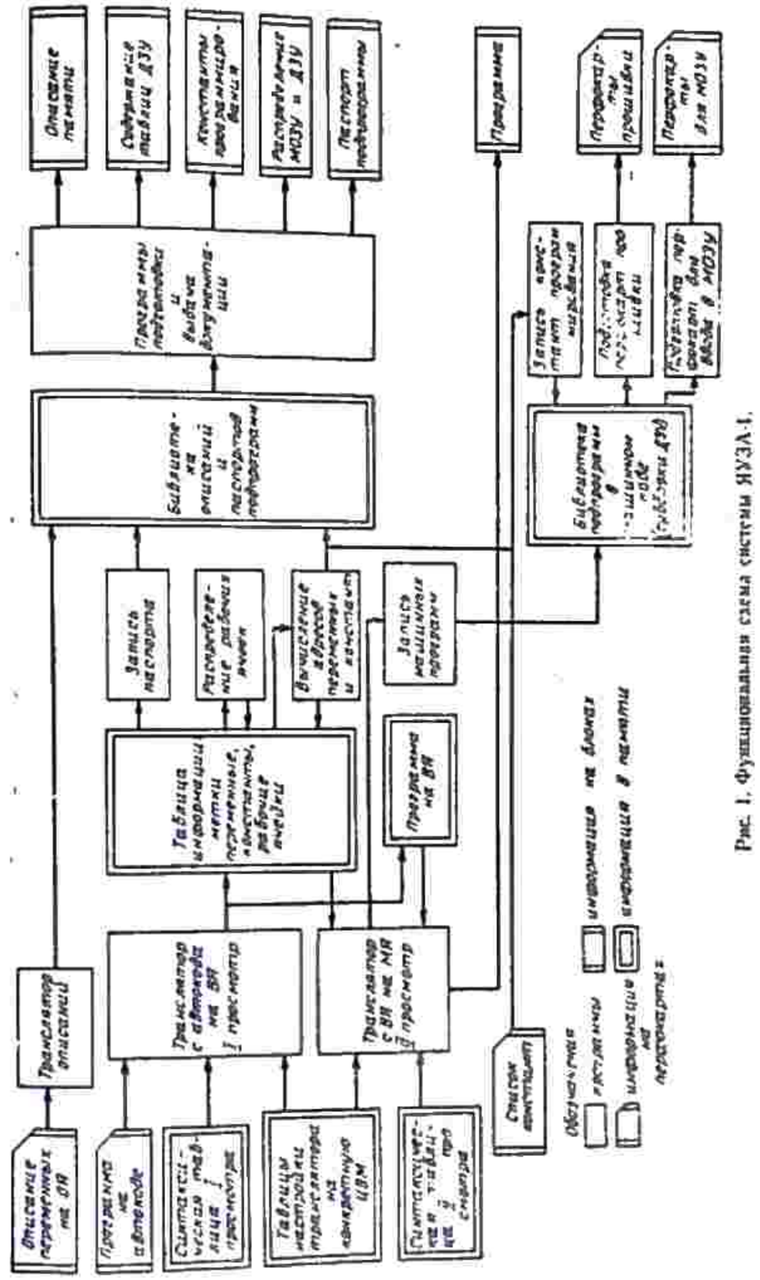


Рис. 1. Функциональная схема системы ЯУЗА-1

Средства описания глобальных переменных близки к средствам описания данных языка АЛГЭМ [1] Основное отличие состоит в том, что в языке ЯУЗА кроме характеристик переменных, входящих в составные переменные или составные массивы, задается их расположение в памяти конкретной ЦВМ.

Описание глобальных переменных составляется по зонам. Под зоной понимается участок внутренней или внешней памяти ЦВМ, в котором может быть размещена составная переменная или массив составных переменных [1] Составная переменная, состоящая из простых переменных, констант и массивов, размещается в одном или нескольких машинных словах (ячейках). В целях экономии памяти, предусматривается возможность размещения нескольких величин в одной ячейке, т. е.

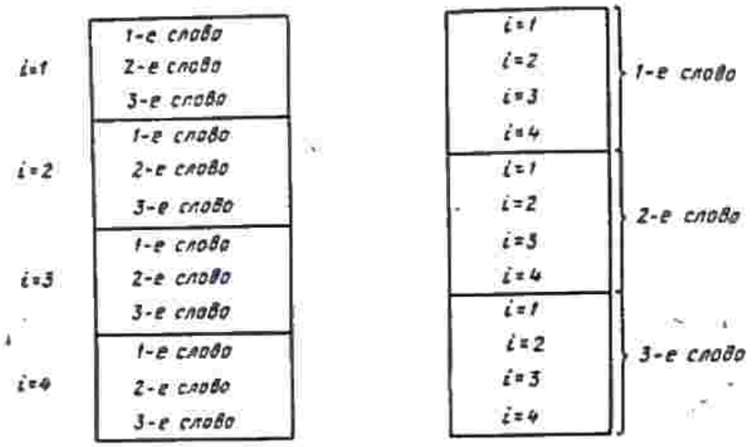


Рис. 2. Объектовая зона [1 4, сл. 1 : 3] зона А (*а*), словная зона [сл. 1 3, 1 4] зона В (*б*).

упаковка данных. Размещение массива составных переменных может быть организовано двумя способами: пословно или пообъектно. При объектовом способе слова, образующие составную переменную, размещены в памяти последовательно, друг за другом При словном способе последовательно размещаются первые слова всех элементов массива составных переменных, затем все вторые слова и т. д. (рис. 2).

Описание зоны состоит из заголовка зоны и формуляра. В заголовке приводится идентификатор и граничные пары зоны, включающие в себя граничные пары массива составных переменных и граничную пару с описателем **СЛ**, определяющую количество и порядок слов зоны. Для зоны, в которой размещается простая составная переменной, указывается только одна граничная пара с описателем **СЛ** Если граничная пара слов записана слева в списке граничных пар зоны, то этим задается пословная структура зоны, если справа, то — пообъектная (рис 2).

Зоны могут быть заданы постоянного или переменного размера в зависимости от того, являются ли границы в граничных парах числами или идентификаторами переменных В заголовке зоны записывается адрес зоны в виде восьмеричного числа (если зона фиксирована) или в виде идентификатора (если зона плавающая), а также наименование устройства (**ДЗУ, МБ**), в котором размещается зона (описатель **МОЗУ**

опускается). После заголовка следует формуляр зоны, в котором последовательно описываются переменные, константы и массивы составной переменной Для каждой переменной, кроме ее идентификатора и указателя типа, задаются разряды слова, в которых размещается эта переменная, масштаб, цена младшего разряда (для масштабированных переменных) и пределы изменения переменной. Для массива, входящего в составную переменную зоны, указываются его граничные пары

Отдельные описания переменных и массивов в формуляре зоны отделяются друг от друга разделителями **И** и **ИЛИ**. Если описания отделяются разделителем **И**, то задаваемые ими переменные должны размещаться в одном слове, но в разных разрядах. В случае разделителя **ИЛИ** переменные должны размещаться в одном и том же участке памяти. Ни-

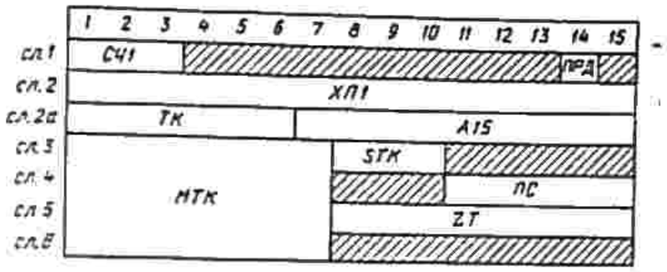


Рис. 3. Расположение переменных в памяти ЦВМ.

же приведен пример описания зоны Описание соответствует расположению переменных в памяти ЦВМ, приведенному на рис. 3.

```
[1 . 15,1 : 4, СЛ 1 . 6] ЗОНА ТСЧ АДР 15432;
СЛ1.        ЦЕЛ СЧ1 Р 1 · 3 И
            БУЛ ПРД Р 14 · 14;
СЛ2.        ВЕЩ ХП1 Р 1 . 15 Ц 100 ИЛИ
            ЦЕЛ ТК Р 1 . 6 И
        АДР А15 Р 7 . 15;
СЛ3 — 6 : [1 6] ЦЕЛ МТК Р 1 . 7 И
            ЦЕЛ STK Р 8 : 10;
          [1 : 5] БУЛ ПС Г 11 : 15,
            ВЕЩ ZTP 8 · 15 М 360.0,
ФОРМ;
```

Приведенное выше описание задает размещение двумерной зоны ТСЧ в памяти ЦВМ. Эта зона имеет пообъектовую структуру. Формуляр зоны занимает 6 слов (с 1 по 6) В первом слове в разряды с 1 по 3 помещена целая переменная СЧ1 и в 14-м разряде — булева переменная ПРД. Второе слово отведено под вещественную масштабированную переменную ХП1 с ценою младшего разряда 100 В этой же ячейке расположены целая переменная ТК и адресная переменная А15. В словах 3—6 в разрядах 1 · 7 расположены элементы массива МТК. В этих же словах помещены другие переменные: целая переменная STK (разряды 8 . 10 слова 3), булевый массив ПС, элементы которого занимают последовательные разряды слова 4, и вещественная величина Т в разрядах 8 : 15 слова 6 с масштабом представления 360 0

Описания зон набиваются на перфокартах, вводятся в ЦВМ, проходят полный синтаксический контроль и записываются в БИБЛИОТЕКУ ОПИСАНИЯ ПАМЯТИ, организованную на магнитной ленте БИБЛИОТЕКА имеет каталог, в котором перечисляются идентификаторы всех зон, уже записанных в библиотеку.

2. СОСТАВЛЕНИЕ ПРОГРАММЫ НА АВТОКОДЕ И ЕЕ ТРАНСЛЯЦИЯ

Подпрограмма записывается на покомандном автокоде ЯУЗА, в котором каждая команда состоит из списка меток прихода, метки и списка подкоманд. Метка имеет обычный смысл. В списке меток прихода могут быть перечислены с описателем **ОТ** метки тех команд, от которых передается управление на данную команду. В случае сложных программ указание меток прихода существенно помогает программисту ориентироваться в программе.

Каждая подкоманда состоит из операции, операнда и модификатора. Совокупность подкоманд определяет общую операцию, выполняемую командой.

Названия операций составляются из знаков, подчеркнутых букв и цифр. Набор знаков, букв и цифр произволен и выбирается при настройке автокода на конкретную управляющую ЦВМ.

В автокоде ЯУЗА операнды задаются с помощью констант, меток, названий, глобальных переменных и выражений. В качестве названий глобальных переменных и меток используются составные имена, состоящие из идентификатора переменной, точки и идентификатора зоны или подпрограммы, которой принадлежит переменная или метка. Например, X.L или M1. ТОМ. В случае модификации операнда модификатор записывается в квадратных скобках после операнда с указанием имени регистра модификатора и модифицирующей величины.

Подпрограмма снабжается заголовком, в котором указывается ее идентификатор и начальный адрес. Если подпрограмма разбита на части (основная часть и вставки), то каждая вставка начинается с указания ее начального адреса. В текст программы можно вводить любые примечания, заключенные в символы «*» и «;» или «*» и «*Z». Пример программы, записанной на автокоде ЯУЗА для одно-, двух- и трехадресной условных ЦВМ, приведен на рис. 4.

В приведенных на рис. 4 программах используются глобальные величины:

переменная X.L из зоны L,

массивы N.R и B.R из зоны R.

Зона локальных (рабочих) переменных и зона констант названы соответственно P и K.

В названиях операций используется следующая мнемоника:

→ и ← — пересылка информации;

⇒ передача управления;

ВД — выделение;

0, 1 — значения сигналов ω;

СМ, И1, РА, РР — обозначения устройств ЦВМ.

Трансляция состоит из двух просмотров. При первом просмотре осуществляется полный синтаксический контроль входного текста, составляется список (таблица информации — ТИ) всех переменных, меток и констант (глобальных и локальных). Вместо идентификаторов в текст программы включается ссылка на ТИ и программа переводится на внутренний язык транслятора.

Программа ВЫЧИСЛЕНИЯ АДРЕСОВ осуществляет просмотр ТИ и, обращаясь к БИБЛИОТЕКЕ ОПИСАНИЙ ПАМЯТИ, заполняет ТИ значениями базовых адресов, номеров разрядов, занимаемых переменной, и адресов глобальных меток.

Локальные переменные всех подпрограмм в целях экономии памяти размещаются в общем массиве памяти — зоне рабочих ячеек. Выбор адресов производится с учетом указаний об эквивалентности, содержащихся в программе, которые позволяют нескольким переменным выделять одни и те же ячейки памяти.

Одноадресная	Двухадресная	Трехадресная
ПРОГР ПРИМЕР; НАЧАЛО 3200, ← X.L; ВД X.L; → X.P; СМ ← 0.K; → РЕЗ. P; И1 ⇐ 77776; ОТ М2 + 1: М1: ← X.P; —• N.R • [И1], ПС — 0 ⇒ М2; → X.P; ← B.R [И1], + РЕЗ. P; → РЕЗ. P; ⇒ М1; ВСТ 3600, М2 : И1 ⇐ 1 [И1], ПС — 0 ⇒ М1; ← РЕЗ. P; + X.P; → РЕЗ. P, ОСТ; КОНЕЦ	ПРОГР ПРИМЕР, НАЧАЛО 1500, X.L ВД X.L → PP; PP → X.P, 0 → PP; PP → РЕЗ. P; 1 K → PP; PP → И.2; ОТ М2, М1:X.P—N.R[И.2]→PP; 1 ⇒ М2, PP → X.P; РЕЗ P + B.R[И.2]→РЕЗ P; ⇒ М1; ВСТ 2600, М2 . И.2 — 1 ⩾ 0 ⇒ М1, РЕЗ P + X.P. → PP; PP → РЕЗ. P, ОСТ, КОНЕЦ	ПРОГР ПРИМЕР; НАЧАЛО 717; X.L ВД X.L → X P; РА ← 0; 0 → РЕЗ. P; ОТ М2; М1. X. P ВЧМ N R [РА]→Д.Р; 1 ⇒ М2, Д.Р → X.P; РЕЗ. Р СЛМ В. R [РА]→РЕЗ. P; ⇒ М1, ВСТ 1200; М2 : РА — 1 < 0 ⇒ М1; РА ← 1; РЕЗ. Р СЛМ X.P →РЕЗ. P; ОСТ; КОНЕЦ

Рис. 4. Программа, записанная на автокоде ЯУЗА для одно-, двух- и трехадресной ЦВМ.

По окончании трансляции программы в БИБЛИОТЕКУ ПАСПОРТОВ ПОДПРОГРАММ, организованную аналогично библиотеке описаний памяти, заносятся адреса всех меток, которые являются входами в программу и адреса рабочих ячеек. Информация, собранная в паспорте программы, позволяет осуществить стыковку отдельных подпрограмм по передачам управления и по рабочим ячейкам.

Полученные в результате трансляции машинные команды, заносятся в БИБЛИОТЕКУ ИНФОРМАЦИИ ПАМЯТИ. Для ЦВМ, имеющих память типа ДЗУ в БИБЛИОТЕКЕ размещаются сведения о командах и константах, которые располагаются в порядке возрастания адресов и группируются по конструктивному признаку: субблоки, модули, платы и т. п.

3. ВЫПУСК И КОРРЕКТИРОВКА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ

Система автоматизации программирования ЯУЗА производит техническую документацию, которая включает в себя:

— описание памяти глобальных переменных;

— значения величин, входящих в зоны ДЗУ;

— значения констант;

— глобальное распределение памяти МОЗУ и ДЗУ;

— паспорт программы;

— программу.

Вся документация выпускается в соответствии с ЕСКД на бланках формата 11 и 12.

Описание памяти глобальных переменных. Содержит описание зоны, выпускается автоматически после записи описания зоны в БИБЛИОТЕКУ ОПИСАНИЙ или по желанию пользователя путем вызова из БИБЛИОТЕКИ описания требуемой зоны.

Значения величин, входящих в зоны ДЗУ. Документ содержит численные значения каждой переменной (элемента массива), полученные с учетом масштаба величин и способа их упаковки. Документ выпускается автоматически при записи зон ДЗУ в БИБЛИОТЕКУ ИНФОРМАЦИИ ПАМЯТИ.

Значения констант. Документ содержит восьмеричные значения всех констант, заданных в программе. Выпускается автоматически при изменениях констант и по желанию пользователя.

Глобальное распределение памяти. Составляется по устройствам памяти и по режимам работы программы, отличающимся распределением памяти. Содержит наименование участка памяти (зоны, подпрограммы), начальный и конечный адреса.

Паспорт программы. Составляется по подпрограммам и содержит информацию о расположении в памяти, адресах входа и выхода, используемых константах, рабочих ячейках и глобальных переменных. Документ выпускается автоматически после окончания трансляции подпрограммы.

Программа. Составляется по подпрограммам и содержит команды в восьмеричном коде, команды на автокоде и примечания к группам команд. Документ соответствует рукописному тексту, передаваемому пользователем для трансляции программы.

Система может изготовлять перфокарты программ и перфокарты для прошивки программ в ДЗУ управляющей ЦВМ. Перфокарты могут выдаваться после трансляции каждой подпрограммы и предназначаются для ввода программы в МОЗУ управляющей ЦВМ для отладки подпрограммы. Перфокарты для прошивки выдаются блоками, соответствующими конструктивным элементам ДЗУ (субблокам, модулям). После набора на КЗУ данных об участке памяти, соответствующей элементу ДЗУ, система по информации БИБЛИОТЕКИ выдает колоду перфокарт для прошивки этого элемента. По этим перфокартам можно осуществить ручную или автоматическую прошивку ДЗУ и проконтролировать правильность прошивки на специальном контрольном стенде.

В системе ЯУЗА-1 осуществляется также корректировка программ и всей документации. Все изменения фиксируются в БИБЛИОТЕКАХ и выпускаются соответствующие этим изменениям извещения.

4. НАСТРОЙКА ТРАНСЛЯТОРА НА КОНКРЕТНУЮ ЦВМ

Система ЯУЗА-1 построена таким образом, что вся информация о характеристиках управляющей ЦВМ, системе команд и изобразительных средствах автокода сосредоточена в таблицах. Чтобы использовать систему для программирования для конкретной управляющей ЦВМ, необходимо составить таблицы, соответствующие этой ЦВМ.

В систему ЯУЗА-1 входит программа, позволяющая формировать эти таблицы по информации, записанной на специальном языке.

Программной базой системы ЯУЗА является синтаксически-управляемый транслятор [2], с помощью которого производится формирование понятий при анализе входного текста автокод-программы и описаний зон памяти. Семантическая часть транслятора реализуется отдельными подпрограммами и сменными таблицами, о которых упоминалось выше.

Система ЯУЗА-1 реализована на М-220 с комплектностью: 2 МОЗУ, МБ и 3 МЛ. Скорость трансляции с печатью всех документов составляет

25—40 команд в минуту в зависимости от объема библиотек. Система практически используется для семи типов управляющих ЦВМ.

ЛИТЕРАТУРА

1 Шилдер Ф. Ф. Алгоритмический язык описания экономико-математических задач — АЛГЭМ. В сб. «Цифровая вычислительная техника и программирование», вып. 1, 1966.

2 Жоголев Е. А. Алгоритм выделения понятий с помощью синтаксической таблицы. «Журнал вычислительной математики и математической физики», 1965, т. 5, № 4.

УДК 681.3.51

К ВОПРОСУ ОЦЕНКИ ФУНКЦИОНИРОВАНИЯ ЗВЕНА «ОПЕРАТОР—ПУЛЬТ УПРАВЛЕНИЯ» С ПОМОЩЬЮ МОДЕЛИРОВАНИЯ НА ЦВМ

В. П. ИСАЕВ, М. Ф. ПОПОВ, Р. А. ПОПОВ

Управление в сложных системах с большим объемом обрабатываемой информации и малым допустимым временем ее обработки $T_{опер}$ целесообразно осуществлять программно с использованием ЦВМ. Однако при автоматизации управления процессами человек является неотъемлемым элементом системы, из которой он принципиально неустраним, так как в реальных условиях имеет место эволюция его целей, требующих для своего осуществления целенаправленных управляющих воздействий, являющихся прерогативой человеческой деятельности. Аналогичная ситуация имеет место в системах, решающих задачи оперативного управления, при котором необходимо гибко и своевременно реагировать на случайные изменения, характер и время возникновения которых трудно предсказать.

Ниже рассматриваются такие СЧМ (системы «человек — машина»), в которых оператор вмешивается в процесс управления либо, если алгоритм работы не предусматривает детерминированную реакцию на некоторые ситуации, либо при отклонениях процесса от нормального. Оператор, участвующий в работе такой системы, прежде всего должен быстро и точно ориентироваться в средствах управления и выполнять заданные функции в определенные оперативные сроки. Поэтому в условиях резко ограниченного времени ошибкой оператора считается как собственно ошибка в производстве управляющего воздействия, так и превышение времени действия. На количество и характер ошибок влияют как параметры технических средств (способ предъявления информации, размещение и структура органов управления, темп изменения и др.), так и характеристики самого оператора (квалификация, тренированность, эмоциональная устойчивость и т. д.).

Все ошибки, происходящие в СЧМ, как сенсорные (неправильное восприятие информации), так и моторные (неправильная реализация принятого решения) нами считаются случайными событиями, приводящими к снижению эффективности и задержке в применении технических средств. При оценке эффективности СЧМ необходимо учитывать прежде всего психофизиологические особенности и функциональные возможности оператора, исходя из средних данных оператора соответствующей квалификации. При этом для выявления общих закономерностей известные различия между отдельными индивидуальными операторами должны устраняться статистически.

Из анализа этапов деятельности оператора [1] известно, что, приняв и переработав информацию, он выполняет те или иные действия, направленные на изменение управляемого объекта. С помощью действий

информация передается от человека к системе посредством оконечных технических средств, основным и наиболее распространенным из которых следует считать пульт управления (ПУ). Имеющиеся в литературе данные [2, 3] позволяют определить лишь самые общие требования к органам управления и не дают оснований ни для выбора такой конструкции, которая обеспечила бы необходимую точность и надежность действий оператора, ни для объективного определения необходимых затрат времени на производство управляющих воздействий.

При разработке системы «оператор — пульт управления» (О — ПУ) необходимо производить количественный анализ и получить оценки продолжительности выполнения алгоритма с учетом размещения, количества и типа органов управления и контроля. До сих пор такой анализ фактически не производится из-за отсутствия методики и трудностей, связанных с вероятностным характером участвующих в процессе вычислений. Необходимость разработки такой методики очевидна, так как существует много разновидностей ПУ, и для учета технических, и психофизиологических особенностей работы с ними целесообразно иметь типовую, базовую модель функционирования звена «О — ПУ».

Исследование такой модели возможно только с использованием современных ЦВМ как в качестве датчика любой необходимой информации, так и в целях обработки результатов, как это предлагалось в работе [1]. Однако в качестве объекта исследования в определенных условиях лучше иметь не реального оператора, а его математическую модель, что позволяет достаточно просто получить оценку влияния различных параметров и характеристик рассматриваемого звена «О — ПУ». Принимая, что исполнительным действиям предшествовала оценка обстановки и принятие решения, которые выходят за рамки рассматриваемого вопроса, применим для исследования статистическую модель, использующую принцип дробления деятельности на элементарные операции [5], а также статистические зависимости и известные постоянные при выполнении элементарных операций. В основу модели «О — ПУ» закладывается процесс информационного поиска, поскольку при выполнении заданного алгоритма оператор отыскивает глазами отдельные строго фиксированные органы управления и контроля, а руками — органы управления. Эти операции (как показывают теория и опыт) занимают наибольшее время и могут являться источником ошибок и задержек.

Управление системой с помощью ПУ состоит в осуществлении связи через оператора между двумя его основными частями, информирующей (контрольной) и соответственно управляющей (операционной). Процесс работы оператора-исполнителя управляющих воздействий, имея в виду связь между входными и выходными величинами, детерминирован и может быть представлен в виде логической цепи, имеющей причинно-следственный характер. Логическая цепь, включающая восприятие входной информации, содержащей требование на исполнительное действие, и заканчивающаяся выдачей в систему соответствующего регулирующего воздействия, представляет собой цикл управления. Для нормального функционирования необходимо, чтобы

$$T_{\text{п упр}} = T_{\text{опер}}.$$

Анализируемая логическая цепь представлена на рис. 1. Функционирование звена «О — ПУ» состоит в выборе участка операционного поля из всего множества участков, имеющихся на ПУ, выборе «критических» элементов из имеющихся на данном участке, наборе кода, контроле правильности его набора и передачи кода, предписывающего определенное поведение управляемого объекта.

Поскольку управляющее действие возникает в ответ на те или иные сигналы, появляющиеся на индикаторах и контрольных приборах, большое значение имеет вопрос о зависимости его от соотношения сенсорного и моторного полей ПУ, и характера соответствия стимулов и реакций. Известно, что влияние фактора соответствия проявляется незначительно в величинах латентного периода и более значительно — в величинах моторного компонента на стадии выполнения движения.

Поэтому целесообразно более подробно остановиться на анализе принципов построения ПУ, с тем чтобы, с одной стороны, определить необходимые при исследовании ограничения, а с другой — выявить наиболее общие закономерности в построении ПУ, для того чтобы полученные результаты были достаточно широко применимы при различных конструктивных решениях. Из имеющегося в литературе [6—8] анализа размещения органов управления можно сделать выводы о необходимости строгого разделения сенсорной и моторной зон ПУ. При этом для исключения запоминания сложной последовательности обращения к органам управления необходимо упорядоченное размещение их на операционном поле, которое в общем виде может состоять из нескольких разноудаленных поверхностей (1—10), каждая из которых в свою очередь может иметь несколько субоперационных полей (СОП) — функционально самостоятельных участков, целостных в смысловом (целевом) отношении. Это оправдывается тем, что перемещение глаза и самого оператора при поиске органов управления на большой поверхности операционного поля, естественно, занимает много времени, поскольку точное опознание возможно в сравнительно небольшой зоне центрального зрения. Размер и расположение субоперационных полей должны выбираться такими, чтобы внутри них мог осуществляться оптимальный информационный поиск, без существенных затруднений для оператора с учетом его пропускной способности. При этом органы контроля должны соответствовать органам управления, т. е. размещаться «оперативно-упорядоченно», мерой последнего может служить интегральная оценка времени выполнения алгоритма (при отсутствии частных количественных мер).

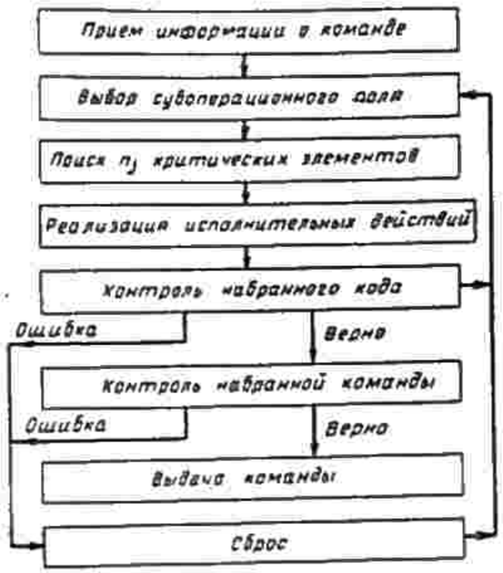


Рис. 1. Блок-схема деятельности оператора в СЧМ.

Предполагаем целесообразным использование однородных органов управления, по крайней мере в пределах одного СОП, т. е., например, только кнопок (клавиш), чтобы исключить различные способы манипуляции с ними и не перестраивать двигательные стереотипы, что увеличивает время действия и может вносить ошибки. Из рассмотрения на ПУ исключаются органы управления критическими функциями, т. е. наиболее ответственные органы, для задействования которых оператор должен сделать по крайней мере два движения (конструктивно они снабжаются специальными замками, фиксаторами или предохранительными крышками).

Все разнообразие органов управления при отдаче команд, расчет жений, сообщений по целевому назначению можно разделить на за принципиальные части: смысловую и адресную, каждая из которы может по-разному конструктивно реализоваться. Представляется цель сообразным принять, что адресная часть, как выдающая наиболее ра кообразные сообщения, должна иметь универсальный характер постро ния, например в виде матричного наборного поля с упорядоченным р положением цифрового набора. Последнее является типичным СО с собственной клавишей сброса. Смысловая часть является обычно м нее разнообразной и может быть представлена в виде наборного поля с фиксированным набором смыслов при произвольном конструктивно расположении (обычно также в пределах одного СОП). В случае н обходимости универсальной смысловой передачи (например, произво ных текстовых сообщений) необходимо использовать универсальну клавиатуру первого типа. В каждом конкретном случае может имет место несколько смысловых и адресных полей и их сочетание, но дл исследования на модели это не имеет значения, так как указанное ра деление позволяет нам полностью оценить особенности информационна го поиска на каждом из них в отдельности.

Индикаторная часть ПУ здесь рассматривается лишь в плане ко троля за производимыми действиями, исходя из данного выше опред ления функций оператора, и представляет собой контрольную стро при сличении с которой и тождественности набора передаваемой инфо мации оператор производит результирующее действие по отсылке путе нажатия кнопки на исполнительном СОП.

С учетом принятых допущений была построена статистическая мо дель, представленная на рис. 2, основные блоки которой производят сл дующие действия. Моделируется время, затрачиваемое оператором н выбор СОП с учетом основных факторов: установочного движения моторного действия. Временные характеристики первого характеризуют ся латентным периодом, направленностью и взяты из работы [11], а ком поненты моторного действия моделировались по данным [12].

В зависимости от типа и конкретных характеристик M, N, СОП определяются затраты времени на набор соответствующего кода. Та левая ветвь блок-схемы алгоритма отображает набор цифрового код при равновероятном появлении каждой из 10 цифр в числе. В соответ ствии с принятым допущением об упорядоченном расположении инфор мации на данном СОП принимаем, что информационный поиск имеет дихотомический характер (с последовательным делением поля на две части). Среднее число фиксаций, определенное в процессе поиска, дает время, затрачиваемое на реализацию этого процесса. Время фиксации T_Φ взято постоянным [3], ибо, как показано в работе [10], среднее время фиксации не зависит от количества элементов информационного поля, индивидуальных свойств отдельных элементов и числа значимых эле ментов поля. При этом различия в продолжительности фиксаций на критических и фоновых сигналах статистически не значимы.

Затраты времени на простую сенсомоторную реакцию моделирова лись на основании статистических данных, приведенных в работе [13]. Данные аппроксимировались нормальным законом, проверка произво дилась по критерию Пирсона. Время на проверку набранного кода мо делировалось с учетом установочного движения и временем фиксации каждого элемента набранного кода. Учет влияния ошибок, допускаемых оператором в процессе набора и проверки кода моделировался с по мощью датчика случайных чисел. Вероятность ошибок в наборе одного элемента изменялась параметрически в пределах от 0 до 0,035.

Правая ветвь блок-схемы моделирует набор смысловой части команды. Среднее количество фиксаций определялось по методике ин формационного поиска, разработанной для случайно выбираемых эле-

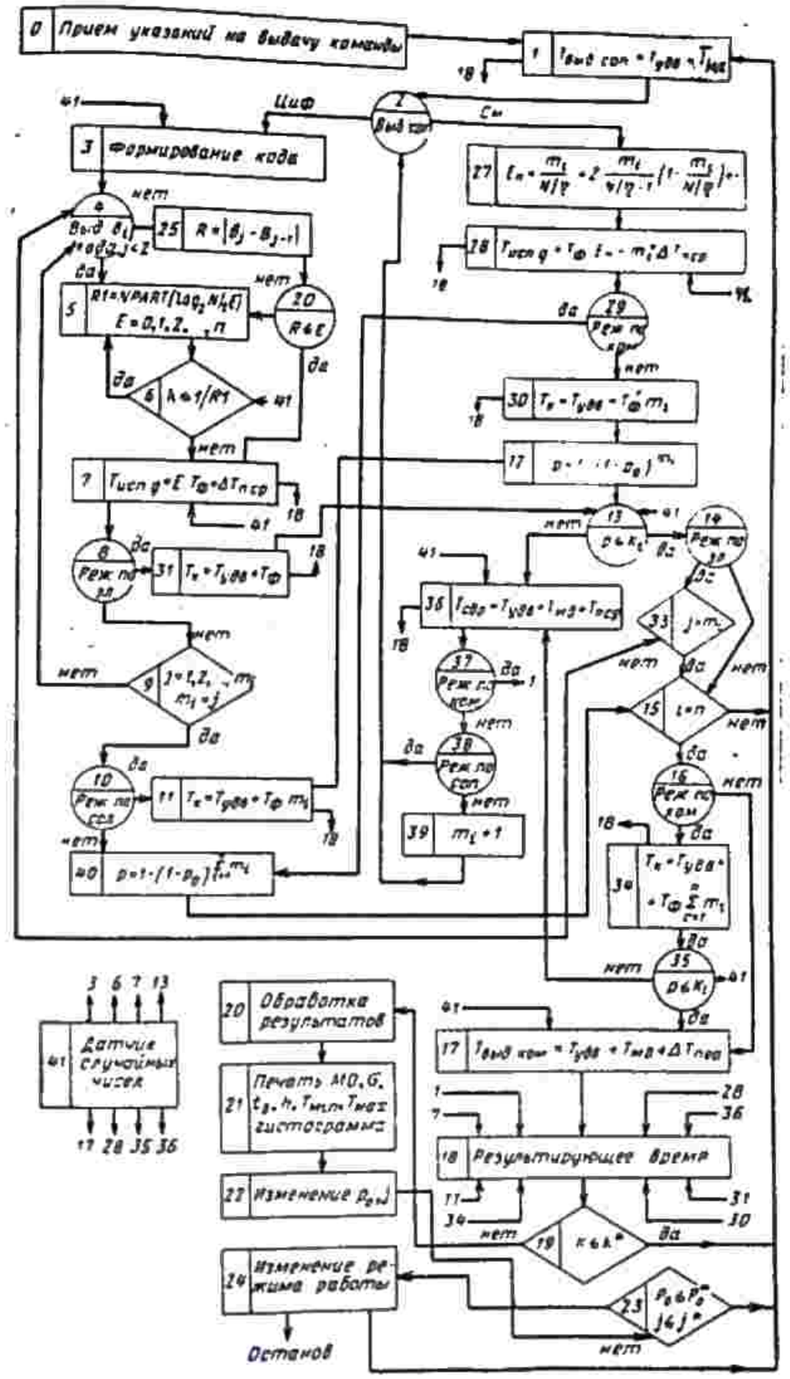


Рис. 2. Блок-схема модели «оператор—пульт управления».

ментов [9]. При этом моделируются три режима функционирования звена, отличающиеся способом организации проверки сброса набранного кода:

— режим с проверкой каждого набранного элемента и его сбросом при наличии ошибки;

— режим с проверкой кода, набранного в целом на СОП, и сбросом кода данного СОП при наличии ошибки;

— режим с проверкой кода всей команды и последующим ее сбросом при наличии ошибки.

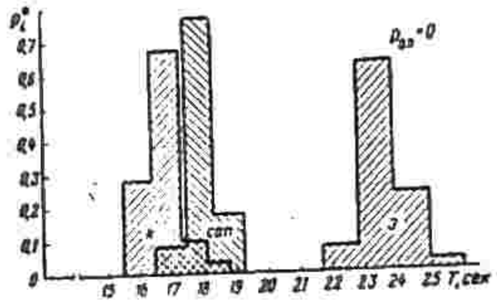


Рис. 3. Гистограммы для трех режимов функционирования звена «О—ПУ».

Результирующее время, затрачиваемое на все операции, накапливалось в сумматоре и определяло случайное значение времени процедуры выдачи команды. В результате моделирования получены гистограммы времени набора, контроля, передачи команды и зависимости среднего времени формирования и выдачи команды от величины вероятности ошибки, допускаемой оператором при наборе одного элемента кода.

На рис. 3 приведены гистограммы для всех трех режимов функционирования звена «О — ПУ» при условии идеальной работы оператора,

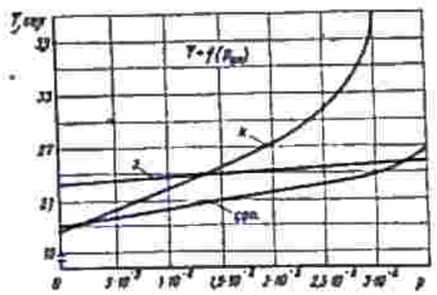


Рис. 4. График оценки качества функционирования звена «О—ПУ».

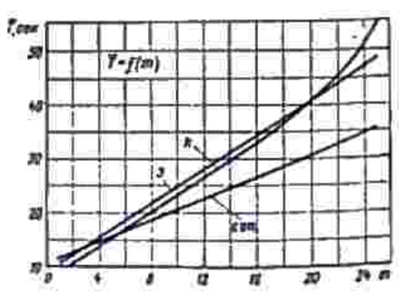


Рис. 5. Зависимости среднего времени формирования и выдачи команды от количества элементов на СОП

т. е. $p=0$. Сопоставляя это время с допустимым $T_{опер}$, можно оценить качество функционирования звена и возможность его использования в конкретной системе управления.

В реальных условиях оператор будет допускать ошибки, вероятность появления которых зависит от подготовки оператора, его возраста, типа нервной системы, условий обитаемости, наличия экстремальных нагрузок и ряда других факторов, трудно поддающихся учету. Поэтому для оценки качества функционирования звена «О—ПУ» по интегральному временному критерию в каждом из трех режимов смогут быть использованы графики, приведенные на рис. 4.

С целью иллюстрации возможностей данной модели приводятся зависимости среднего времени формирования и выдачи команды от количества элементов на СОП с универсальной клавиатурой при постоянной вероятности ошибки, допускаемой оператором при наборе одного элемента кода, равной 0,01 (рис. 5). Зависимости построены при условии, что разброс времен набора информации на смысловых СОП оказывает несущественное влияние на характер изменения общего времени. Внося поправку на время набора информации на смысловых полях, с помощью этих зависимостей можно производить оценку затрат времени в функции от количества элементов на двух СОП с универсальной клавиатурой.

В общем случае чистая зависимость времени формирования и выдачи команды от количества элементов клавиатуры универсального типа может быть легко получена исключением части блоков правой ветви алгоритма, изображенного на рис. 2.

Кроме того, за счет изменения начальных условий программы модель позволяет анализировать зависимость среднего времени формирования и выдачи команды от количества СОП, их взаимного расположения, от количества элементов любого СОП, т. е. практически может быть использована как инструмент при проектировании пультов управления и оценке существующих конструкций.

Предлагаемая методика с использованием статистической модели функционирования звена «О — ПУ» может быть использована для оценки качества функционирования (путем интегральной оценки времени), сравнительной оценки и разработки требований к ПУ в СЧМ, а также в качестве составной (типовой) части при построении общей модели пункта управления системы.

ЛИТЕРАТУРА

1. Зинченко В. П. и др. Анализ деятельности человека — оператора. В сб. «Инженерная психология». Изд. МГУ, 1964.
2. Ломов Б. Ф. Человек и техника (очерки инженерной психологии). Изд. 2-е. Изд-во «Советское радио», 1966.
3. «Инженерно психологические требования к системам управления». Под ред. Зинченко В. П. ВНИИТЭ, 1967.
4. Барабошкин Ю. М. О применении ЭВМ при исследовании информационной психологии. Вып. 2, 1968.
5. Ляпунов А. А., Шестопал Г. А. Об алгоритмическом описании процессов управления. «Математическое просвещение», 1957, № 2.
6. Гаврилов Л. В., Николаев В. И., Темпов В. И. Результаты исследования некоторых режимов работы оператора. Системы «Человек и автомат». Изд-во «Наука», 1965.
7. Николаев В. И. Контроль работы судовых энергетических установок. Изд-во «Судостроение», 1965.
8. Фокин Ю. Г. Принципы упорядочения размещения органов управления и контроля. «Стандарты и качество», 1967, № 3.
9. Березкин Б. С., Зинченко В. П. Исследование информационного поиска. «Проблемы инженерной психологии». Изд-во «Наука», 1967.
10. Гапенко Д. Ю., Зинченко В. П. и др. О правомерности использования среднего времени фиксации в математической модели информационного поиска. «Проблемы инженерной психологии». Вып. 2, 1968.
11. Гиппенрейтер Ю. Б. Опыт экспериментального исследования работы зрительной системы наблюдателя. В сб. «Инженерная психология». Изд. МГУ, 1964.
12. Вудсон У., Коновер Д. Справочник по инженерной психологии для инженеров и художников-конструкторов. Изд-во «Мир», 1968.
13. Яковец Б. И. О соотношении быстроты простой и сложной двигательных реакций. «Проблемы инженерной психологии». Вып. 3, 1968.

УДК 681.3.06:519.152

НЕКОТОРЫЕ МАТЕМАТИЧЕСКИЕ МОДЕЛИ СИСТЕМ С РАЗДЕЛЕНИЕМ ВРЕМЕНИ

С. Ф. ЯШКОВ

ВВЕДЕНИЕ

Основным назначением систем с разделением времени (СРВ) является оперативное предоставление ресурсов вычислительной машины большому количеству одновременно работающих абонентов. Организация эффективного взаимодействия пользователей с системой осуществляется с помощью алгоритма планирования (АП), который распределяет время центрального процессора (ЦП) между несколькими задачами. Поэтому при проектировании СРВ первостепенное значение приобретает проблема оценки качества функционирования системы с различными АП в постановке [1].

Алгоритм планирования СРВ отводит каждой задаче некоторый интервал (квант) рабочего времени ЦП. Если задача не решилась полностью в отведенный ей квант, то она прерывается и перемещается в конец очереди заявок*), ожидающих дальнейшей обработки, причем в процессе смены задач некоторое время ЦП (время переключения — r) непроизводительно расходуется на обмен между ступенями памяти и работу служебных программ. При одной общей очереди заявок, приоритетность которых предполагается одинаковой, подобный алгоритм осуществляет стратегию циклического планирования. Дальнейшими его модификациями являются многоприоритетные (многоочередные) АП, позволяющие сократить потери времени ЦП на обмен. При такой стратегии планирования новые заявки поступают в очередь высшего приоритета. Если при предоставлении задаче очередного кванта обслуживание не завершено, то она перемещается в конец очереди более низкого приоритета, в дальнейшем вызываясь для исполнения только при отсутствии заявок в очередях с бо́льшими приоритетами. Задачи из последней очереди обрабатываются до конца без прерываний. Существует еще ряд модификаций АП, однако в основу их заложены принципы, рассмотренные выше.

Для математического описания процесса распределения временных ресурсов СРВ можно представить ее в виде системы массового обслуживания (СМО). Из большого числа работ по математическим моделям (ММ) СРВ, представление о которых может дать обзор [4], в первую очередь следует отметить [5—9]. Большинство моделей СРВ строятся в предположении нулевого времени переключения ЦП ($r=0$). Только в работах [5—7] исследуются одноочередные СРВ при $r\neq 0$, однако при анализе используется весьма трудоемкая техника преобразований. Предлагаемые в настоящей статье математические модели отражают более общий случай в отношении учета времени переключения не только для одноочередных СРВ, но и для СРВ с произвольным количеством очередей, причем при построении ММ используется достаточно удобный математический аппарат.

Анализ рассматриваемых ниже моделей проводится при следующих предположениях.

1) потоки заявок от нескольких абонентов аппроксимируются одним пуассоновским потоком с интенсивностью λ ($\lambda=\text{const}$);

2) время обслуживания поступающих заявок, определяемое длительностью обработки в ЦП при однопрограммном режиме, распределено экспоненциально с функцией распределения (ф. р.)

$$B(t)=P(S<t)=1-e^{-\mu t};$$

3) используется дисциплина квантованного обслуживания в порядке поступления заявок при наличии обратной связи (т. е. перевода не полностью обслуженной заявки обратно в очередь);

4) потери времени на переключение заявок постоянны ($r=\text{const}$);

5) длина очереди заявок неограниченна;

6) рассматривается стационарный режим при отсутствии перегрузки.

*) Термины «заявка» и «задача» в данной статье имеют одинаковый смысл.

СРВ С ЦИКЛИЧЕСКИМ АЛГОРИТМОМ ПЛАНИРОВАНИЯ (МОДЕЛЬ 1)

Схема функционирования модели 1 изображена на рис. 1. Основной целью анализа является нахождение аналитических выражений, позволяющих определять среднее время ожидания в очереди задач, классифицированных по длительности их выполнения. Пусть максимальное

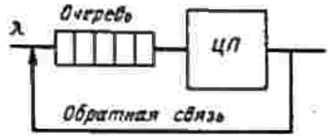


Рис. 1. СРВ с циклическим алгоритмом планирования.

время кванта составляет величину Q и время решения некоторой задачи S, тогда величина k, получаемая из неравенства

$$(k-1)Q\leqslant S<kQ,\quad k=1,\ 2,\ \ldots, \tag{1}$$

будет определять необходимое число квантов для обработки данной задачи и являться признаком классификации заявок входного потока. Определим время цикла τ_i как время, проходящее между моментами окончания $(i-1)$-го кванта и начала i-го кванта обслуживания данной заявки. Тогда время ожидания заявки в очереди составит величину

$$W_k=\sum_{i=1}^{k}\tau_i. \tag{2}$$

Целесообразно для учета r определить расширенный квант Q_p как

$$Q_p=Q+r. \tag{3}$$

На основании предположений 2—4 ф. р. длительности k квантов обслуживания ($k=1,\ 2,\ \ldots$) имеет вид

$$F_k(t)=P(V<t)=\begin{cases}0 & \text{при } (j-1)Q_p<t<r+(j-1)Q_p,\\ 1-e^{-\mu(t-jr)} & \text{при } r+(j-1)Q_p<t<jQ_p,\\ 1 & \text{при } t>k\cdot Q_p,\ j=1,\ 2,\ \ldots,\ k.\end{cases} \tag{4}$$

Введем обозначение

$$s=e^{-\mu Q}. \tag{5}$$

В дальнейших рассуждениях будут использоваться выражения для $M_k(V)$ и $M_k(V^2)$ — первого и второго моментов ф. р. $F_k(t)$. Опуская

простые, но несколько громоздкие промежуточные выкладки, можно записать:

$$M_k(1) = (1-\varepsilon^k)\left(\frac{1}{\mu}+\frac{r}{1-\varepsilon}\right), \tag{6}$$

$$M_k(V^2) = \frac{2}{\mu^2}(1-\varepsilon^k - k\mu Q\varepsilon^k + \theta_k), \tag{7}$$

где

$$\theta_k = r\frac{Q\mu^2[\varepsilon-\varepsilon^k(\varepsilon+k-k\varepsilon)]+\mu(1-\varepsilon)[1-\varepsilon^k(1+k-k\varepsilon)]+}{(1-\varepsilon^2)} \rightarrow$$
$$\rightarrow \frac{+\frac{\mu^2 r}{2}[(1+\varepsilon)(1-\varepsilon^k)-2k\varepsilon^k(1-\varepsilon)]}{} \tag{8}$$

Возвращаясь к анализу модели, рассмотрим случай поступления в некоторый момент в систему заявки, еще не получавшей ни одного кванта обслуживания (отмеченной заявки). Можно считать, что перед ней в системе находится среднее количество $\bar{n}_1$ заявок, а ЦП с вероятностью δ занят обработкой первой из этих заявок. Средняя длительность обслуживания (средняя длина расширенного кванта $\bar{Q}_p$) определяется как 1-й момент ф. р $F_k(t)$ при $k=1$ и находится из формулы (6)

$$\bar{Q}_p = (1-\varepsilon)/\mu + r. \tag{9}$$

Величина $\bar{Q}_p$ состоит из среднего времени $\bar{Q}=(1-\varepsilon)/\mu$, затрачиваемого ЦП на обработку заявки в течение кванта, и времени переключения. Так как случайные моменты поступления в СРВ новой заявки и начала обслуживания очередной заявки независимы, то ф р времени обслуживания заявки, обрабатываемой в ЦП в момент входа отмеченной заявки в систему, выражает фактически распределение остатка кванта обслуживания. Используя для нахождения этой ф. р. результаты [3] и определяя ее первый момент, получим выражение для средней длительности остатка кванта обслуживания:

$$\bar{Q}_1 = \frac{1}{1-\varepsilon}\left[\frac{1}{\mu}(1-\varepsilon) - Q\varepsilon\right].$$

Легко видеть, что время первого цикла составляет

$$\tau_1 = \bar{n}_1\bar{Q}_p - \lambda(\bar{Q}_1 - \bar{Q}_1) = \bar{n}_1\bar{Q}_p - \delta\beta, \tag{10}$$

где

$$\beta = \frac{\varepsilon}{1-\varepsilon}\left[Q - \frac{1}{\mu}(1-\varepsilon)\right]. \tag{11}$$

По распределению длины очереди модель 1 эквивалентна СМО без обратной связи с тем же входным потоком заявок и ф. р. времени обслуживания заявок $F_k(t)$ при $k=\infty$. Тогда, определяя 1-й и 2-й моменты ф р $F_k(t)$ с помощью формул (6) и (7) при $k=\infty$ и используя формулу Поллачека — Хинчина [2], получим выражение для среднего количества заявок в СРВ:

$$\bar{n}_1 = \frac{\lambda}{\mu}\left[1+\frac{\mu r}{1-\varepsilon}+\frac{\lambda(1+\theta)}{\mu-\lambda-\frac{\mu\lambda r}{(1-\varepsilon)}}\right], \tag{12}$$

где

$$\theta = r\frac{\mu^2 Q\varepsilon + \mu(1-\varepsilon)+\frac{\mu^2 r}{2}(1+\varepsilon)}{(1-\varepsilon)^2}$$

Вероятность занятости ЦП определяется как $\lambda M_k(V)$, $k=\infty$. Таким образом,

$$\delta = \lambda/\mu + \lambda r/(1-\varepsilon) \tag{13}$$

и представляет фактическую загрузку СРВ, совпадающую с $\rho(\rho=\lambda/\mu)$ при $r=0$.

Теперь определены все компоненты выражения (10). Если S отмеченной заявки превосходит величину Q, то в момент второго поступления заявки в очередь перед ней будет находиться $\bar{n}_2$ заявок. Так как после срабатывания обратной связи момент i-го ($i=2, 3, \ldots$) поступления заявки в очередь определяет момент начала обработки очередной заявки, то в этом случае $\bar{Q}_i=\bar{Q}$ и $\beta=0$. Величину $\bar{n}_2$ можно определить как сумму среднего количества новых заявок, поступающих в СРВ за время (τ_1+Q_p), и среднего количества частично обслуженных заявок, вновь возвращающихся в очередь. Используя теорему Литтла [2] и основное свойство экспоненциального распределения, находим выражение для τ_2:

$$\tau_2 = \bar{Q}_p[\lambda(\tau_1+Q_p)+\bar{n}_1\varepsilon].$$

После простых преобразований, принимая во внимание (10), получим

$$\tau_2 = \tau_1\alpha + Q_p(\alpha-\varepsilon) + \delta\beta\varepsilon, \tag{14}$$

где

$$\alpha = \lambda/\mu + \varepsilon(1-\lambda/\mu) + \lambda r. \tag{15}$$

Рассуждая аналогично и применяя метод математической индукции, можно показать, что

$$\tau_i = \tau_{i-1}\alpha + Q_p(\alpha-\varepsilon), \qquad (i=3, 4, \ldots).$$

Выражая последовательно τ_i через τ_{i-1}, τ_{i-2}, ..., τ_{i-j} ($i-j\geqslant 2$) и используя (10) и (14), находим общее выражение для времени цикла:

$$\tau_i = \begin{cases} \bar{n}_1\bar{Q}_p - \delta\beta, & i=1, \\ \bar{n}_1\bar{Q}_p\alpha^{i-1} - \delta\beta\alpha^{i-1} + Q_p(\alpha-\varepsilon)(1-\alpha^{i-1})/(1-\alpha)+\beta\delta\varepsilon\alpha^{i-2}, & i=2, 3, \ldots \end{cases} \tag{16}$$

Подставляя (16) в (2), получаем окончательное выражение для математического ожидания времени пребывания в очереди заявок k-го класса:

$$W_k = \frac{1-\alpha^k}{1-\alpha}(\bar{n}_1\bar{Q}_p - \delta\beta) + \frac{kQ_p(\alpha-\varepsilon)}{1-\alpha} - \frac{Q_p(\alpha-\varepsilon)(1-\alpha^k)}{(1-\alpha)^2} + \beta\delta\varepsilon\frac{1-\alpha^{k-1}}{1-\alpha}, \tag{17}$$

где Q_p, ε, $\bar{Q}_p$, β, $\bar{n}_1$, δ и α определяются формулами (3), (5), (9), (11), (12), (13) и (15) соответственно.

Теперь можно найти время отклика *) СРВ на любую задачу

$$U = W_k + S \tag{18}$$

Заметим, что с помощью выражения (12) можно определить минимальную граничную длительность кванта $Q_{гр}$, ниже которой длина очереди становится бесконечной:

$$Q_{гр} = \frac{1}{\mu}\ln\frac{\mu-\lambda}{\mu-\lambda-\mu\lambda r}. \tag{19}$$

С помощью этого же выражения можно определить верхнюю границу интенсивности входного потока

$$\lambda_{гр} = \frac{\mu}{1+\mu\cdot r/(1-\varepsilon)}, \tag{20}$$

которая совпадает с μ при $r=0$.

*) Под временем отклика СРВ подразумевается среднее время, проходящее от момента окончания запроса на решение задачи до начала выдачи ее результатов.

СРВ С МНОГОПРИОРИТЕТНЫМИ АЛГОРИТМАМИ ПЛАНИРОВАНИЯ (МОДЕЛЬ 2)

Рассмотрим СРВ с произвольным количеством очередей N ($N \geqslant 2$), схема функционирования которой изображена на рис. 2. Анализ модели 2 проводится для случая отличного от нуля времени переключения ЦП на основе методики исследования подобных моделей при $r=0$ в работах [8—9]. При построении ММ будут использоваться некоторые результаты, полученные в предыдущем разделе.

Поступающие заявки при $S<NQ$ классифицируются согласно неравенству (1), а оставшаяся часть заявок относится к классу $k=N$, так как для последней очереди обратная связь отсутствует. Приоритет очередей уменьшается с увеличением номера очереди. В соответствии с дисциплиной обслуживания в модели 2 время ожидания в очередях некоторой заявки k-го класса (отмеченной заявки) составляет

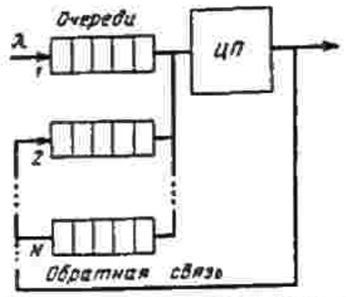


Рис. 2. СРВ с многоприоритетным алгоритмом планирования.

$$w_k = T_{1k} + T_{2k},$$

где T_{1k} — сумма времени, необходимого для завершения кванта обслуживания ранее выбранной заявки, и времени, требуемого для обработки всех заявок, находящихся в первых k очередях в момент прибытия отмеченной заявки; T_{2k} — время обработки всех новых заявок, поступающих в систему за время w_k.

В форме математических ожиданий это уравнение имеет вид:

$$W_k = M(T_{1k}) + M(T_{2k}). \tag{21}$$

Определим основные компоненты выражения (21).

Предполагая $k<N$, для определения $M(T_{1k})$ преобразуем модель 2 в эквивалентную ей трехочередную СМО с относительными приоритетами. В преобразованной СМО без обратной связи в первую очередь (высшего приоритета) поступают заявки с интенсивностью $\lambda^*_1=\lambda$, требующие времени обработки не более kQ, следовательно, ф. р. времени обслуживания этих заявок $F_k(t)$ определяется выражением (4). Во вторую очередь с интенсивностью λ^*_2 поступают заявки, требуемое время обслуживания которых находится в пределах $kQ<S\leqslant(N-1)Q$. Из специфики функционирования модели ясно, что $F_1(t)$ [формула (4)] представляет ф. р. времени обслуживания заявок потока λ^*_2. И, наконец, в последнюю очередь низшего приоритета поступает поток λ^*_3 заявок класса N, ф. р. времени обслуживания которых можно записать в виде:

$$G(t) = P(L<t) = \begin{cases} 0 & \text{при } t<r, \\ 1-e^{-\mu(t-r)} & \text{при } r<t\leqslant\infty. \end{cases}$$

Второй момент ф. р. $G(t)$ составляет

$$M(L^2) = (2/\mu^2) + (2r/\mu) + r^2. \tag{22}$$

Используя результаты анализа СМО с относительными приоритетами

[2], определим $M(T_{1k})$ как среднее время ожидания в очереди высшего приоритета в преобразованной СМО

$$M(T_{1k}) = W_{0k}/(1-\rho^*_1),$$

где W_{0k} — время, требуемое для завершения обслуживания заявки; ρ^*_1 — коэффициент загрузки заявками высшего приоритета.

В рассматриваемом случае

$$\rho^*_1 = \lambda^*_1 M_k(V),$$

$$W_{0k} = \frac{1}{2}\left[\lambda^*_1 M_k(V^2) + \lambda^*_2 M_1(V^2) + \lambda^*_3 M(L^2)\right], \quad k=1,2,\ldots,N-1,$$

где $M_k(V)$, $M_k(V^2)$ ($k=1, 2,\ldots,N-1$) и $M(L^2)$ определяются формулами (6), (7) и (22) соответственно. Очевидно, что

$$\lambda^*_2 = \lambda\sum_{i=k+1}^{N-1} a^{i-1} = \lambda\left(\frac{a^k - a^{N-1}}{1-a}\right) = \lambda\gamma_{kN}, \tag{23}$$

$$\lambda^*_3 = \lambda a^{N-1}.$$

Подставляя полученные выражения в уравнение для $M(T_{1k})$, получим

$$M(T_{1k}) = \frac{\lambda\left[M_k(V^2) + \gamma_{kN}M_1(V^2) + a^{N-1}M(L^2)\right]}{2\left[1-(1-a^k)\left(\rho+\frac{\lambda r}{1-a}\right)\right]}, \tag{24}$$

где $\rho=\lambda/\mu$.

Для нахождения $M(T_{2k})$ обратимся к начальной модели 2. Время от момента поступления отмеченной заявки класса k до момента начала ее последнего кванта обслуживания составляет величину $W_k + (k-1)Q_p$. Новые заявки, прибывающие за это время, получают $(k-1)$ квантов, следовательно, величина $M_{k-1}(V)$ [формула (6)] представляет среднюю длительность обслуживания этих заявок. Тогда, используя теорему Литтла, можно записать

$$M(T_{2k}) = \lambda M_{k-1}(V)\left[W_k + (k-1)Q_p\right]. \tag{25}$$

Подставляя (24) и (25) в (21) и решая полученное уравнение относительно W_k, находим

$$W_k = \frac{\lambda\left[M_k(V^2) + \gamma_{kN}M_1(V^2) + a^{N-1}M(L^2)\right]}{2\left[1-(1-a^k)\left(\rho+\frac{\lambda r}{1-a}\right)\right]\left[1-(1-a^{k-1})\left(\rho+\frac{\lambda r}{1-a}\right)\right]} +$$

$$+\frac{(1-a^{k-1})\left(\rho+\frac{\lambda r}{1-a}\right)}{1-(1-a^{k-1})\left(\rho+\frac{\lambda r}{1-a}\right)}(k-1)Q_p, \quad k=1,2,\ldots,N-1; \tag{26}$$

где $M_k(V^2)$ ($k=1, 2,\ldots,N-1$), $M(L^2)$ и γ_{kN} определяются формулами (7), (22) и (23) соответственно.

Рассматривая случай $k=N$, заметим, что все заявки, находящиеся в СМО в момент прибытия заявки класса N, должны быть обслужены до момента начала последнего интервала обслуживания отмеченной заявки. Тогда величина $M(T_{1N})$ определяется как среднее время ожидания заявки.

ния в очереди в СМО, ф. р. времени обслуживания в которой имеет вид:

$$A(t)=P(V^*<t)=$$
$$=\begin{cases}0 & \text{при } (j-1)Q_p<t<r+(j-1)Q_p,\\ 1-e^{-\mu(t-jr)} & \text{при } r+(j-1)Q_p\leqslant t<jQ_p,\\ 0 & \text{при } (N-1)Q_p\leqslant t<(N-1)Q_p+r,\\ 1-e^{-\mu(t-Nr)} & \text{при } t\geqslant(N-1)Q_p+r,\ j=1,2,\ldots,N-1.\end{cases}$$

Так как ф. р. $A(t)$ совпадает с ф. р. $F_н(t)$ для $t<(N-1)Q_p$, то моменты ф. р. $A(t)$ находятся с использованием формул (6), (7) и записываются в виде

$$M(V^*)=\frac{1}{\mu}+r\frac{1-\varepsilon^N}{1-\varepsilon}, \tag{27}$$

$$M(V^{*2})=\frac{2}{\mu^2}(1+Nr\mu\varepsilon^N+\theta_N), \tag{28}$$

где θ_N определяется формулой (8).

Используя (27) и (28), можно найти величину $M(T_{1N})$ как среднее время ожидания в СМО без обратной связи с ф. р. времени обслуживания $A(t)$:

$$M(T_{1N})=\frac{\frac{\rho}{\mu}(1+Nr\mu\varepsilon^N+\theta_N)}{1-\rho-\lambda r\frac{1-\varepsilon^N}{1-\varepsilon}}.$$

Величина $M(T_{2N})$ определяется из формулы (25) при $k=N$. Тогда в соответствии с (21) получаем

$$W_N=\frac{\frac{\rho}{\mu}(1+Nr\mu\varepsilon^N+\theta_N)}{\left[1-\rho-\lambda r\frac{1-\varepsilon^N}{1-\varepsilon}\right]\left[1-(1-\varepsilon^{N-1})\left(\rho+\frac{\lambda r}{1-\varepsilon}\right)\right]}+$$
$$+\frac{(1-\varepsilon^{N-1})\left(\rho+\frac{\lambda r}{1-\varepsilon}\right)}{1-(1-\varepsilon^{N-1})\left(\rho+\frac{\lambda r}{1-\varepsilon}\right)}(N-1)Q_p, \tag{29}$$

где θ_N находится из (8).

Легко видеть, что условие отсутствия перегрузки в модели 2 выполняется при

$$\lambda_{гр}=\left(\frac{1}{\mu}+r\frac{1-\varepsilon^N}{1-\varepsilon}\right)^{-1}<1. \tag{30}$$

Таким образом, окончательные результаты для нахождения среднего времени ожидания заявок класса k в модели 2 даются выражениями (26) и (29). Время отклика СРВ по-прежнему определяется формулой (18).

Следует заметить, что из полученных выражений для W_k в моделях 1, 2 непосредственно следуют результаты частных моделей [8] при $r=0$ и [9] при $r=0$, $N=\infty$.

АНАЛИЗ РЕЗУЛЬТАТОВ И ВЫВОДЫ

Проведенное исследование дает возможность количественного определения времени ожидания и отклика заявок различных классов как функций от λ, μ, Q и r для СРВ с различными алгоритмами планирования. Для иллюстрации вида функциональных зависимостей некоторые из полученных результатов изображены графически. На рис. 3, 4 приведены зависимости времени ожидания k-го класса заявок от ρ для СРВ с циклическим и многоприоритетным алгоритмами планирования ($N=4$) при $\mu=1$ с$^{-1}$; $Q=0{,}5$ с, $r=0{,}05$ с. На этих же графиках с целью сравнения пунктиром изображена аналогичная зависимость для обычной СМО без обратной связи с ф. р. времени обслуживания $G(t)$. Можно видеть,

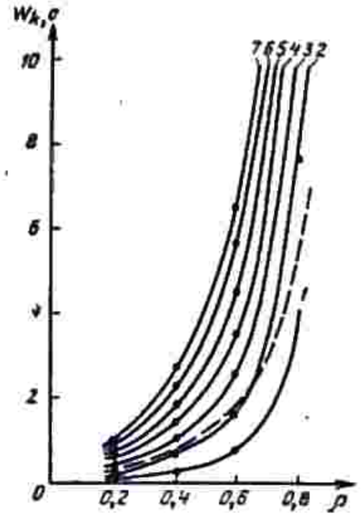


Рис. 3. Зависимость среднего времени ожидания заявок k-го класса ($k=1, 2, \ldots, 7$) от ρ для модели 1 ($\mu=1$ с$^{-1}$, $Q=0{,}5$ с, $r=0{,}05$ с).

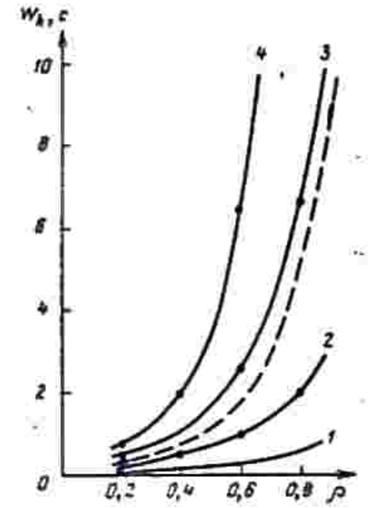


Рис. 4. Зависимость среднего времени ожидания заявок k-го класса от ρ для модели 2 при $N=4$ ($\mu=1$ с$^{-1}$, $Q=0{,}5$ с, $r=0{,}05$ с)

как уменьшается время ожидания коротких задач ($S<1/\mu$) в модели 2 по сравнению с моделью 1. При $\rho\geqslant\lambda_{гр}/\mu$ СРВ входят в режим перегрузки. Для одно- и четырехочередной СРВ $\lambda_{гр}$ составляют соответственно 0,89 и 0,904 с$^{-1}$.

На рис. 5 показана зависимость среднего времени ожидания задачи с $S=1$ с от величины кванта для модели 1 при $\lambda=0{,}65$ с$^{-1}$ и $\mu=1$ с$^{-1}$. График претерпевает разрывы при $Q=S/n$ ($n=1, 2, \ldots$) и сходится к величине $W^*=S\rho/(1-\rho)$. Рис. 6 иллюстрирует факт резкого изменения зависимости $W=f(Q)$ для той же задачи при учете потерь времени на квантование ($r=0{,}1$ с). В этом случае кривая асимптотически приближается к вертикали при $Q_{гр}=0{,}206$ с (граничной величине кванта).

Точность полученных результатов была проверена с помощью моделирования рассмотренных систем. Программная модель СРВ была составлена на языке СЛЭНГ и реализована на ЦВМ БЭСМ-4. Величины отклонений экспериментальных результатов от теоретических иллюстрируются данными для двух экспериментов, приведенными в таблице.

Результаты моделирования при значениях загрузки в диапазоне (0,3—0,75) совпали с расчетными с точностью (0—2) %, превосходящей, в частности, точность результатов в [7—8]. При этом, погрешность, возрастающая с увеличением ρ, зависела, в основном, от длительности моделирования. Было также подтверждено предположение о возможности использования полученных формул для приближенного расчета

Теоретические и экспериментальные значения среднего времени ожидания в очереди заявок k-го класса

Алгоритм планирования	λ, c^{-1}	μ, c^{-1}	Q, с	r, с	k	W_k расч., с	W_k машин., с
Циклический	0,5	1	0,5	0,1	1	0,703	0,678
					2	1,514	1,446
					3	2,354	2,311
					4	3,218	3,252
					5	4,104	4,374
					6	5,008	5,100
С четырьмя уровнями приоритета					1	0,341	0,345
					2	0,965	1,047
					3	2,240	2,192
					4	5,060	5,051

характеристик СРВ с произвольными законами распределения времени решения задач при коэффициентах вариации, не превосходящих единицы с точностью до 18%).

В заключение необходимо отметить, что полученные зависимости, связывающие характеристики обслуживания абонентов СРВ с параметрами задач, могут быть использованы для оптимизации проектируемых

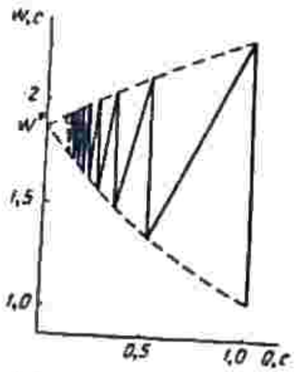


Рис. 5. Зависимость среднего времени ожидания задачи длительности $S=1$ с от величины кванта для модели 1 при $r=0$ ($\lambda=0{,}65$ с$^{-1}$, $\mu=1$ с$^{-1}$).

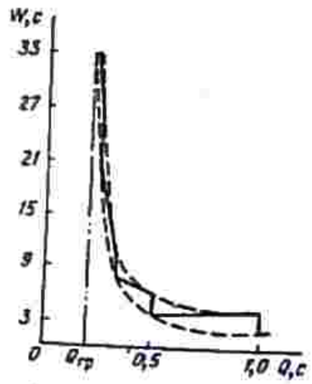


Рис. 6. Зависимость среднего времени ожидания задачи длительности $S=1$ с от величины кванта для модели 1 при $r=0{,}1$ с ($\lambda=0{,}65$ с$^{-1}$, $\mu=1$ с$^{-1}$).

систем. Это можно интерпретировать, например, как выбор разумного компромисса между противоречивыми требованиями уменьшения времени ожидания коротких задач и увеличения загрузки системы при некоторых стоимостных ограничениях. Данная проблема будет служить предметом дальнейших исследований.

Выражаю глубокую благодарность докт. техн. наук Липаеву В. В. и канд. техн. наук Шигину А. Г. за полезные советы, способствовавшие выполнению настоящей работы.

ЛИТЕРАТУРА

1. «Системы с разделением времени». В сб. статей под ред. Карплюса У. Изд-во «Мир», 1969.
2. Саати Т. Л. Элементы теории массового обслуживания и ее приложения. Изд-во «Советское радио», 1965.

3. Хинчин А. Я. Работы по математической теории массового обслуживания. Физматгиз, 1963.
4. Авен О. И., Коган Я. А. Математические модели сложных вычислительных систем (обзор). «Автоматика и телемеханика», 1971, № 1.
5. Adiri I., Avi—Itzhak B. A time-sharing queue. Management Science, 1969, v. 15, № 11.
6. Coffman E. G. Markov Chain Analysis of Multiprogrammed Computer Systems. Naval Research Logistics Quarterly, 1969, v. 16, № 2.
7. Rasch P. J. A queueing theory study of round-robin scheduling of time-shared computer systems. Journal of A.C.M., 1970, v. 17, № 1.
8. Coffman E. G., Kleinrock L. Feedback Queueing Models for Time-Shared Systems. Journal of A.C.M., 1968, v. 15, № 4.
9. Schrage L. E. The queue M/G/1 with feedback to lower priority queues. Management Science, 1967, v. 13, № 7.

УДК 681.3.06.519.152

ОБ ОДНОЙ ЗАДАЧЕ ОПРЕДЕЛЕНИЯ РЕЖИМА ПРОФИЛАКТИКИ

Ю. П. ГОРОХОВ, Г. А. СОКОЛОВ

ПОСТАНОВКА ЗАДАЧИ

Во многих работах по теории надежности, например [1—4], рассматриваются задачи назначения сроков проведения профилактических мероприятий (контроль параметров, тестовая проверка, ремонт, замена устройств и т. д.). Общим в этих задачах является утверждение того факта, что указанные меры предупредительного характера должны проводиться периодически. Определение периода T (или, что то же самое, количества профилактик $N+2$ на заданном интервале времени $[a_0, a_{M+1}]$) осуществляется методами, существенно зависящими от назначения системы, ее особенностей, характеристик надежности, выбранного критерия и т. д. Мы в дальнейшем будем предполагать, что число $N+2$ известно. Практическая реализация режима профилактики [1] с периодом $T=(a_{M+1}-a_0)/(N+1)$ далеко не всегда осуществима.

Действительно, в сложных информационных системах непрерывного действия (например, управляющих технологическим процессом), когда на ЭВМ осуществляется решение повторяющихся циклов взаимосвязанных задач, проведение, скажем, контроля ЭВМ с помощью тестов возможно лишь в моменты окончания работы одной программы перед началом работы следующей. Возникает задача выбора из заданного множества $M+2$ возможных моментов проведения профилактик на интервале решения цикла задач $N+2$ моментов, наилучшим образом согласующихся в смысле некоторого критерия с предписанием теории надежности.

Итак, дано множество A из $M+2$ ($M\geqslant 0$) действительных чисел $a_0, a_1, \ldots, a_M, a_{M+1}$ ($a_0<a_1<\ldots<a_{M+1}$), соответствующих возможным моментам проведения профилактических мероприятий. Требуется из этого множества выбрать $N+2$ ($N\geqslant 0$) числа ($N<M$) так, чтобы они «равномерно», в смысле некоторого критерия, покрывали отрезок $[a_0, a_{M+1}]$. Другими словами, требуется определить значения переменных $x_0, x_1, \ldots, x_N, x_{N+1}$, минимизирующих функцию

$$\sum_{i=1}^{N+1}\varphi_i(x_i-x_{i-1}), \tag{1}$$

где $\varphi_i(\cdot)$ — выпуклые вниз функции, при условии

$$x_i\in A,\ i=0,\ 1,\ \ldots,\ N+1, \tag{2}$$

$$x_0<x_1<\ldots<x_{N+1}, \tag{3}$$

$$x_0=a_0;\ x_{N+1}=a_{M+1}. \tag{4}$$

В дальнейшем для краткости будем называть задачу (1)—(4) при дополнительном предположении

$$a_j - a_{j-1} = \text{const}, \quad j = 1, 2, \ldots, M+1 \tag{5}$$

задачей I, а саму задачу (1)—(4) — задачей II.

Некоторые полезные модификации данных постановок будут указаны в ходе последующего изложения. Заметим, что задача построения плана «равномерного» проведения серии экспериментов формально также сводится к задачам I и II.

Решение задачи I. В силу условия (5) мы не ограничим общности рассуждений, если под множеством A будем понимать совокупность чисел $0, 1, 2, \ldots, M+1$.

Назовем совокупность точек $x_{i+1}, \ldots, x_{i+\bar{k}}$ пакетом длины $\bar{k}$, если: а) $x_{i+k} - x_{i+k-1} = 1$ для $k = 2, 3, \ldots, \bar{k}$; б) $x_{i+1} - x_i > 1$ либо $x_{i+1} = a_1$, в) $x_{i+\bar{k}+1} - x_{i+\bar{k}} > 1$ либо $x_{i+\bar{k}} = a_{M+1}$. Пусть $x_{i+1}, \ldots, x_{i+\bar{k}}$ и $x_{j+1}, \ldots, x_{j+\bar{m}}$ — два пакета ($x_{j+1} - x_{j+\bar{k}} > 1$) Совокупность точек $x_{i+\bar{k}}+1, \ldots, x_{j+1}-1$ назовем промежутком длины $x_{j+1} - x_{i+\bar{k}} - 1$. Совершенно ясно, что задачу I выбора точек x_i можно эквивалентным образом сформулировать в терминах назначения длин промежутков y_i между соседними точками x_i и x_{i+1}. Для этого заметим, что в силу условия (4) максимально возможное число промежутков равно $N+1$, а их суммарная длина равна $M+2-(N+2) = M-N$.

Подставляя в (1)

$$x_i - x_{i-1} = y_i + 1, \tag{6}$$

приведем задачу I к виду

$$\Phi = \sum_{i=1}^{N+1} \varphi_i(y_i + 1) \Rightarrow \min, \tag{7}$$

$$\sum_{i=1}^{N+1} y_i = M - N, \tag{8}$$

$$y_i \geqslant 0 — \text{целое число}. \tag{9}$$

Задача (7)—(9) для некоторых частных видов функции $\varphi_i(\cdot)$ широко известна в исследовании операций [5]. Мы сформулируем метод решения для произвольных выпуклых вниз функций $\varphi_i(\cdot)$, записав временно задачу (7)—(9) в виде

$$\Phi(z_1, \ldots, z_{N+1}) = \sum_{i=1}^{N+1} \varphi_i(z_i) \Rightarrow \min, \tag{7'}$$

$$\sum_{i=1}^{N+1} z_i = M + 1, \tag{8'}$$

$$z_i \geqslant 0 — \text{целое число (цч)}. \tag{9'}$$

Вектор-решение задачи минимизации функции $\Phi(z_1, \ldots, z_{N+1})$ при условиях $\sum_{i=1}^{N+1} z_i = m$, $z_i \geqslant 0$ — цч обозначим $\bar{z}^m = (\bar{z}_1^m, \ldots, \bar{z}_{N+1}^m)$, а допустимый вектор — $z^m = (z_1^m, \ldots, z_{N+1}^m)$.

Лемма. Пусть p и r — любые натуральные числа или нуль, такие, что $p + r + 1 \leqslant M + 1$, тогда, если функция $\varphi_i(\cdot)$ выпукла вниз, то справедливо неравенство

$$\varphi_i(p+r) - \varphi_i(p+r+1) \leqslant \varphi_i(p) - \varphi_i(p+1). \tag{10}$$

Утверждение леммы следует непосредственно из определения выпуклости, т. е. из неравенства

$$\varphi_i(m-1) + \varphi_i(m+1) \geqslant 2\varphi_i(m),$$

поскольку

$$\varphi_i(p+r) - \varphi_i(p+r+1) \leqslant \varphi_i(p+r-1) - \varphi_i(p+r) \leqslant \ldots \leqslant \varphi_i(p) - \varphi_i(p+1).$$

Теорема 1. Вектор $\bar{z}^{m+1}$ имеет вид:

$$\bar{z}_i^{m+1} = \begin{cases} \bar{z}_i^m & \text{для всех } i \neq k, \\ \bar{z}_k^m + 1 & \text{при } i = k, \end{cases}$$

где номер k определяется условием

$$\max_{1 \leqslant i \leqslant N+1} \Delta\varphi_i(z_i) = \Delta\varphi_k(z_k), \tag{11}$$

а

$$\Delta\varphi_i(z_i) = \varphi_i(z_i) - \varphi_i(z_i + 1).$$

Доказательство. Рассмотрим произвольный допустимый вектор z^{m+1}, отличный от $\bar{z}^{m+1}$. Очевидно, существует такой индекс j, что $z_j^{m+1} > \bar{z}_j^m$ или, что то же самое,

$$z_j^{m+1} - 1 \geqslant \bar{z}_j^m. \tag{12}$$

Представим далее функции $\Phi(z_1^{m+1}, \ldots, z_{N+1}^{m+1})$ и $\Phi(\bar{z}_1^m, \ldots, \bar{z}_{N+1}^m)$ в виде

$$\Phi(z_1^{m+1}, \ldots, z_{N+1}^{m+1}) = \sum_{\substack{i=1 \\ i \neq j}}^{N+1} \varphi_i(z_i^{m+1}) + \varphi_j(z_j^{m+1} - 1) - [\varphi_j(z_j^{m+1} - 1) - \varphi_j(z_j^{m+1})].$$

$$\Phi(\bar{z}_1^m, \ldots, \bar{z}_{j-1}^m, \bar{z}_j^m + 1, \bar{z}_{j+1}^m, \bar{z}_{N+1}^m) = \sum_{i=1}^{N+1} \varphi_i(\bar{z}_i^m) - [\varphi_j(\bar{z}_j^m) - \varphi_j(\bar{z}_j^m + 1)].$$

Покажем, что при всех допустимых $z_1^{m+1}, \ldots, z_{N+1}^{m+1}$

$$\Phi(z_1^{m+1}, \ldots, z_{N+1}^{m+1}) \geqslant \Phi(\bar{z}_1^m, \ldots, \bar{z}_{j-1}^m, \bar{z}_j^m + 1, \bar{z}_{j+1}^m, \ldots, \bar{z}_{N+1}^m). \tag{13}$$

Действительно,

$$\sum_{\substack{i=1 \\ i \neq j}}^{N+1} \varphi_i(z_i^{m+1}) + \varphi_j(z_j^{m+1} - 1) \geqslant \sum_{i=1}^{N+1} \varphi_i(\bar{z}_i^m)$$

по предположению, кроме того, в силу (12) и леммы

$$\varphi_j(z_j^{m+1} - 1) - \varphi_j(z_j^{m+1}) \leqslant \varphi_j(\bar{z}_j^m) - \varphi_j(\bar{z}_j^m + 1).$$

Применяя к обеим частям неравенства (13) оператор минимизации сначала по множеству $\sum_{i=1}^{N+1} z_i^{m+1} = m+1$, $z_i^{m+1} \geqslant 0$ — ц ч., а затем по множеству $1 \leqslant j \leqslant N+1$, получим

$$\Phi(\bar{z}_1^{m+1}, \ldots, \bar{z}_{N+1}^{m+1}) \geqslant \min_{1 \leqslant j \leqslant N+1} \Phi(\bar{z}_1^m, \ldots, \bar{z}_{j-1}^m, \bar{z}_j^m + 1, \bar{z}_{j+1}^m, \ldots, \bar{z}_{N+1}^m). \tag{14}$$

В силу определения $(\bar{z}_i^{m+1})$ справедливо неравенство, обратное приведенному выше, т. е. (14) есть равенство, откуда и следует утверждение теоремы.

На основании доказанной теоремы можно построить следующий алгоритм определения оптимального решения задачи (7)—(9). Рассмотрим ряд из приращений функций $\Delta\varphi_i$, соответствующих значению $y_i=0$, $i=\overline{1, N+1}$:

$$\Delta\varphi_1(1),\ \Delta\varphi_2(1),\ \ldots,\Delta\varphi_{N+1}(1). \tag{15}$$

Увеличивая значение аргумента максимального из приращений на единицу, получим новый ряд:

$$\Delta\varphi_1(1),\ \Delta\varphi_2(1),\ \ldots,\Delta\varphi_{k-1}(1),\ \Delta\varphi_k(2),\ \Delta\varphi_{k+1}(1),\ \ldots,\Delta\varphi_{N+1}(1)$$

и т. д. $M-N$ раз. Полученные при этом значения y_i определяют оптимальное решение *).

В соответствии с изложенным легко получить решение $y_1,\ldots,y_{N+1}$ задачи (7)—(9), тогда решение задачи 1 будет иметь вид

$$x_0=0,\ x_1=x_0+y_1+1,\ x_2=x_1+y_2+1,\ \ldots,\ x_{N+1}=$$
$$=x_N+y_{N+1}+1=M+1. \tag{16}$$

Предположим далее, что $\varphi_i(y_i+1)=\varphi(y_i+1)$. Из выпуклости функции $\varphi(z)$ следует, что $\varphi(l+1)-\varphi(l)\geqslant\varphi(l)-\varphi(l-1)$ для любого натурального l. Кроме того, функции $\varphi(y_i+1)$ не зависят от индекса i, следовательно, переход в процессе решения от значения $y_{i'}=l$ к $y_{i'}=l+1$ хотя бы при одном значении индекса $i=i'$ возможен лишь в том случае, когда для всех остальных значений $i\neq i'$ величины y_i также достигли значения l, но $(N+1)l<M-N$.

Из сказанного можно сделать ряд практических выводов.

1. При $M-N\geqslant N+1$ (или $2N<M-1$) в оптимальном решении будут присутствовать либо промежутки равной длины $\frac{M-N}{N+1}$, если**)

$r\left(\frac{M-N}{N+1}\right)=0$, либо двух типов: $r\left(\frac{M-N}{N+1}\right)$ промежутков длины $l^*=\frac{M-N-r\left(\frac{M-N}{N+1}\right)}{N+1}+1$ и $N+1-r\left(\frac{M-N}{N+1}\right)$ промежутков длины l^*-1.

Все пакеты (в количестве $N+2$) имеют длину, равную единице.

2. При $M-N<N+1$ (или $2N>M-1$) в оптимальном решении будут присутствовать лишь промежутки единичной длины в количестве, равном $r\left(\frac{M-N}{N+1}\right)=M-N$. Общее количество пакетов, очевидно, равно $M-N+1$ и их суммарная длина $N+2>M-N+1$, т. е. длина хотя бы одного пакета больше единицы.

Другими словами, любое решение, состоящее из $M-N+1$ пакетов, разделенных $M-N$ промежутками единичной длины, является оптимальным в смысле критерия (7) независимо от длин пакетов, если в сумме они составляют $N+2$. Для многих практически важных задач подобный произвол нежелателен и тогда рекомендуется рассмотреть дополнительную задачу, регламентирующую структуру пакетов так же, как при $2N\leqslant M-1$ задача (7)—(9) регламентировала структуру промежутков.

*) Впервые данный алгоритм был обоснован Е. Гольштейном и И. Медведовской с помощью аппарата динамического программирования. Наше обоснование опирается только на понятие выпуклой функции и, по-видимому, проще.

**) Здесь и далее $r\left(\frac{x}{y}\right)$ — остаток от деления x на y.

Итак, пусть z_i — длина i-го пакета, тогда

$$\sum_{i=1}^{M-N+1} z_i=N+2, \tag{17}$$

$$z_i\geqslant 0 \text{ — целое число,} \tag{18}$$

$$\sum_{i=1}^{M-N+1}\varphi(z_i)\Rightarrow\min. \tag{19}$$

Задача (17)—(19) полностью аналогична задаче (7)—(9), поэтому, перефразируя первый вывод, имеем: при $M-N<N+1$ в оптимальном решении будут присутствовать либо пакеты равной длины $\frac{N+2}{M-N+1}$ в количестве $M-N+1$, если $r\left(\frac{N+2}{M-N+1}\right)=0$, либо пакеты двух типов: $r\left(\frac{N+2}{M-N+1}\right)$ пакетов длиной $l^*=\frac{N+2-r\left(\frac{N+2}{M-N+1}\right)}{M-N+1}+1$ и $M-N+1-r\left(\frac{N+2}{M-N+1}\right)$ пакетов длиной l^*-1.

Правило (16) модифицируется очевидным образом: x^*_i принимают последовательно значения $0, 1, 2, \ldots, M+1$, за исключением

$$z^*_1,\ z^*_1+z^*_2+1,\ z^*_1+z^*_2+z^*_3+2,\ \ldots,$$
$$z^*_1+z^*_2+\ldots+z^*_{M-N}+M-N-1=M+1-z^*_{M-N+1}. \tag{20}$$

Здесь $z^*_1,\ \ldots,z^*_{M-N+1}$ — решение задачи (17)—(19).

В силу симметрии задач (7)—(9) и (17)—(19) правила (16) и (20) позволяют осуществлять перестановку значений $y^*_1,\ \ldots,\ y^*_{N+1}$ и $z^*_1,\ \ldots,z^*_{M-N+1}$ для получения симметричного решения $x^*_0,\ \ldots,x^*_{N+1}$.

3. Задача 1 легко обобщается на случай $N>M$. В содержательных терминах это означает возможность проведения нескольких профилактических мероприятий в один и тот же момент времени. При этом в силу принципа равномерности $\frac{N+2-r\left(\frac{N+2}{M+2}\right)}{M+2}+1$ мероприятий должно проводиться в $r\left(\frac{N+2}{M+2}\right)$ моментов времени и $\frac{N+2-r\left(\frac{N+2}{M+2}\right)}{M+2}$ в $M+2-r\left(\frac{N+2}{M+2}\right)$ моментов времени. Выбор подмножества множества A, состоящего из $r\left(\frac{N+2}{M+2}\right)$ точек, осуществляется с помощью задачи 1.

Примеры.

Ниже мы рассмотрим с целью иллюстрации четыре примера. Для всех примеров $M+2=32$. Результаты решения, как и само множество A, сведены в табл. 1, где в левом столбце указаны номера примеров, в первой строке — элементы множества A, а в последующих строках единицами отмечены выбранные элементы A.

1. $N+2=22$.

Имеем $2N=40>M-1=29$, кроме того, $r\left(\frac{N+2}{M-N+1}\right)=r\left(\frac{22}{11}\right)=0$, следовательно, оптимальное решение задачи 1 содержит $M-N+1=11$ пакетов длины $\frac{N+2}{M-N+1}=2$ и $M-N=10$ промежутков единичной длины.

2. $N+2=27$.

Таблица 1

A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1
2	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0
3	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
4	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0

A	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
2	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
3	0	1	0	0	0	1	0	1	0	1	0	0	1	0	0	1
4	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Как и выше, $2N > M-1$, но $r\left(\frac{N+2}{M-N+1}\right) = r\left(\frac{27}{6}\right) = 3 \neq 0$, следовательно, оптимальное решение содержит три пакета длиной $l^* = 5$, $M-N+1-r\left(\frac{N+2}{M-N+1}\right) = 3$ пакета длиной $l^*-1=4$ и $M-N=5$ промежутков единичной длины.

3. $N+2=15$.

В данном примере $2N=26 < M-1=29$, кроме того, $r\left(\frac{M-N}{N+1}\right) = r\left(\frac{17}{14}\right) = 3 \neq 0$, следовательно, оптимальное решение содержит три промежутка длиной $l^*=2$, $N+1-r\left(\frac{M-N}{N+1}\right) = 11$ промежутков длиной $l^*-1=1$ и $N+2=15$ пакетов единичной длины.

4. $N+2=8$.

В данном случае также $2N < M-1$ и $r\left(\frac{M-N}{N+1}\right) = r\left(\frac{24}{7}\right) = 3 \neq 0$, следовательно, оптимальное решение содержит три промежутка длиной $l^*=4$, четыре промежутка длиной $l^*-1=3$ и восемь пакетов единичной длины.

Решение задачи II. Задача II может рассматриваться как обобщение задачи I, требующее и более общего метода решения. По своему смыслу она близка к задаче о ближайшем соседе [6] и для ее решения с высокой эффективностью может быть применен метод динамического программирования [6, 7]. Универсальность этого метода позволяет нам рассмотреть задачу II, записав функцию цели в более общем виде, чем (1).

$$\Phi = \sum_{l=1}^{N+1} \varphi_l(x_{l-1}, x_l). \tag{21}$$

Введем обозначения:

$$\Phi_p(y) = \sum_{l=p+1}^{N+1} \varphi_l(x_{l-1}, x_l) + \varphi_p(y, x_p), \tag{22}$$

$$\Phi^*_p(y) = \min_{y=x_{p-1}<x_p<\ldots<x_N<x_{N+1}=a_{M+1}} \Phi_p(y), \tag{23}$$

$$x_l \in A,\ l = p-1,\ p, \ldots, N.$$

Очевидно

$$\min \Phi = \Phi^*_1(a_0).$$

Функция $\Phi^*_p(y)$ есть оценка качества оптимального („равномерного") выбора $(N-p+1)+2$ точек из подмножества тех точек множества A, которые удовлетворяют условию $a_l \geq y$, $y \in A$. При $p=N+1$ из (22) и (23) имеем $\Phi^*_{N+1}(y) = \varphi_{N+1}(y, x_{N+1}) = \varphi_{N+1}(y, a_{M+1})$. Для произволь-

ного p, принимающего значения $1, 2, \ldots, N$, получим рекуррентное соотношение, связывающее $\Phi^*_p(y)$ с $\Phi^*_{p-1}(y)$. Справедлива следующая цепочка равенств:

$$\Phi^*_p(y) = \min_{a_{M-(N-p)} \geq x_p > y = x_{p-1}} \min_{x_p < x_{p+1} < \ldots < x_{N+1} = a_{M+1}} \Phi_p(y) =$$

$$= \min_{a_{M-(N-p)} \geq x_p > y} \left[\varphi_p(y, x_p) + \min_{x_p < x_{p+1} < \ldots < x_{N+1} = a_{M+1}} \sum_{l=p+1}^{N+1} \varphi_l(x_{l-1}, x_l) \right] =$$

$$= \min_{x_{p-1}=y<x_p \leq a_{M-N+p}} [\varphi_p(y, x_p) + \Phi^*_{p+1}(x_p)].$$

Ограничение сверху значений x_p связано с тем, что при $x_p > a_{M-N+p}$ оставшихся точек множества A (больших x_p) не хватит для выбора требуемого числа $N-p+1$ точек (две точки x_{p-1} и x_p уже выбраны).

Итак, имеем систему рекуррентных соотношений динамического программирования

$$\Phi^*_{N+1}(y) = \varphi_{N+1}(y, a_{M+1}),\ a_0 \leq y \leq a_M;\ y \in A, \tag{24}$$

$$\Phi^*_p(y) = \min_{y < x_p \leq a_{M-N+p}} [\varphi_p(y, x_p) + \Phi^*_{p+1}(x_p)].$$

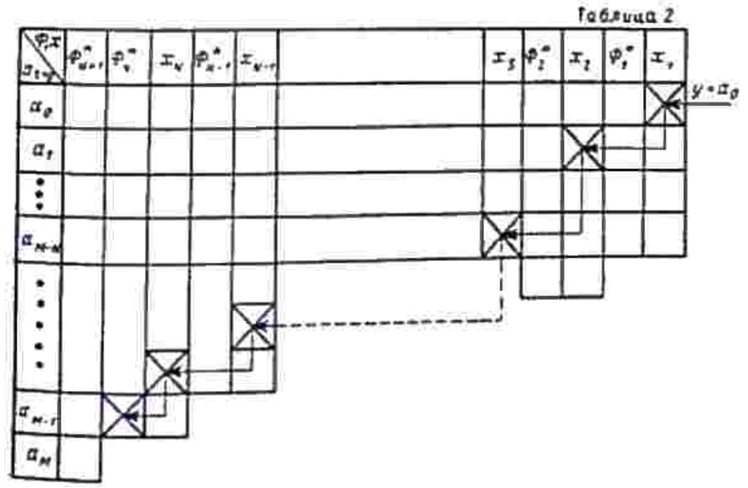
$$a \leq y \leq a_{M-(N-p+1)},\ p = N,\ N-1, \ldots, 1,$$

с помощью которой может быть решена задача II. Более того, если функции (24) построены, то мы имеем оптимальный план выбора $(N+2)$ точек из любой последовательности A, обрезанной слева величиной $y \in A$ и содержащей не менее $N+2$ точек.

Для определения самого плана $\{x_l\}$ служат функции $x_l(a_j)$, $a_j \in A$, которые строятся параллельно с функциями $\Phi^*_l(y)$. По значению y определяется $x_1(y)$, далее $x_2(x_1(y))$, $x_3(x_2(x_1(y)))$ и, наконец, $x_N(x_{N-1}(\ldots(y))\ldots)$, $x_0 = y$, $x_{N+1} = a_{M+1}$. Если в качестве y взять a_0, то мы получим непосредственно оптимальное решение задачи II. Значение функции цели, соответствующее оптимальному решению, есть $\Phi^*_1(y)$.

Таблица функций (табл. 2) $\Phi^*_p(y)$ и $x_p(y)$, $y \in A$ имеет вид:

Таблица 2



Схематически в таблице представлена последовательность построения оптимального решения задачи II. Следует заметить, что практически более удобно работать с системой рекуррентных соотношений (24), записанной в несколько ином виде:

$$\Phi^*_p(y) = \min_{y<x_p\leqslant a_{M-N+p}} [\varphi_p(y, x_p) + \Phi^*_{p+1}(x_p)], \tag{24'}$$

$$a_0 \leqslant y \leqslant a_{M-(N-p+1)},\ p = N, N-1, \ldots, 1,$$

где $\Phi^*_{N+1}(y) = \varphi_{N+1}(y, a_{M+1})$.

Интересно сравнить объем вычислений метода динамического программирования и метода полного перебора. Под объемом вычислений будем понимать количество раз вычислений функций $\varphi_i(x_{i-1}, x_i)$.

Полный перебор связан с вычислением значения функции Φ для C^N_M вариантов, каждый вариант требует однократного вычисления значения $N+1$ функций $\varphi_i(x_{i-1}, x_i)$, т. е. объем вычислений

$$R_1 = (N+1)\,C^N_M.$$

При применении метода динамического программирования построение функции $\Phi^*_p(y)$ требует вычисления значения функции $\varphi_p(y, x_p)$ в

$M-N+p$ точках при $y = a_0$,
$M-N+p-1$ точках при $y = a_1$, и т. д.
1-й точке при $y = a_{M-(N-p+1)}$,

т. е. всего в $\frac{(M-N+p+1)(M-N+p)}{2}$ точках. Поскольку p меняется от 1 до N, имеем ([8], стр 243, формула в. 2):

$$\frac{1}{2}\sum_{p=1}^{N}(M-N+p+1)(M-N+p) = \frac{1}{2}\sum_{k=1}^{N}[(M+1)-k+1][M-k+1] =$$

$$= \frac{1}{12}N[6M(M+1) - (N-1)(6M-2N+4)].$$

Введя дополнительное слагаемое $\frac{M(1+M)}{2}$, учитывающее тот факт, что при $p=N$ под знаком min стоит сумма двух функций $\varphi_N(y, x_N)$ и $\varphi_{N+1}(x_N, a_{M+1})$, окончательно получим

$$R_2 = \frac{1}{12}[6M(M+1)(N+1) - N(N-1)(6M-2N+4)].$$

В приведенной ниже таблице приведены для сравнения величины R_1 и R_2 при некоторых значениях M и N. На пересечении каждой строки и столбца указано в верхнем слева углу R_1 и в нижнем справа — R_2.

Из рассмотрения данной таблицы следует, что при малых N и M, а также при $N \approx M$, т. е. в случаях, не представляющих практического интереса, метод полного перебора предпочтительнее метода динамического программирования. В остальных же случаях метод полного перебора вообще нереализуем, в то время как метод динамического программирования представляется весьма экономичным.

Следует заметить, что в случае $\varphi_i(x_{i-1}, x_i) = \varphi(x_i - x_{i-1})$, где $\varphi(x_i - x_{i-1})$ — выпуклая вниз функция, объем вычислений может быть зна-

M \ N	2	5	10	20	40	60
5	30 / 40	6 / 50				
10	135 / 155	1512 / 240	11 / 275			
50	3670 / 3775	$12{,}7\cdot10^6$ / 7160	$11{,}3\cdot10^{20}$ / 11895	$98{,}9\cdot10^{33}$ / 18415	$42{,}1\cdot10^{49}$ / 23155	
100	14800 / 15050	$46{,}2\cdot10^6$ / 20310	$15\cdot10^{21}$ / 51170	$11{,}5\cdot10^{35}$ / 88150	$5{,}9\cdot10^{52}$ / 138230	$3{,}8\cdot10^{67}$ / 186450

чительно снижен. Прежде всего покажем, что построение функции

$$\Phi^*_N(y) = \min_{\substack{y<x_N\leqslant a_M \\ x_N\in A}} [\varphi(x_N - y) + \varphi(a_{M+1} - x_N)] \tag{25}$$

при фиксированном значении y из A, $y<a_{M+1}$ можно свести к вычислению функций $\varphi(x_N - y)$ и $\varphi(a_{M+1} - x_N)$ в одной точке $x^*_N - y$ и $a_{M+1} - x^*_N$ соответственно.

Теорема 2. При любом значении y из A, $y<a_{M+1}$

$$\Phi^*_N(y) = \varphi(x^*_N - y) + \varphi(a_{M+1} - x^*_N),$$

где x^*_N принимает значение a_i^*, удовлетворяющее условию

$$\Delta a^* = \min_{y<a_i\leqslant a_M} \Delta a. \tag{26}$$

Здесь $\bar{a} = (a_{M+1} + y)/2$, $\Delta a^* = |a_i^* - \bar{a}|$; $\Delta a = |a_i - \bar{a}|$.

Доказательство. Для любого x_N из A, $y<x_N\leqslant a_M$, и любого натурального k имеем

$$\Phi_N(x_N, y) = \varphi(x_N - y) + \varphi(a_{M+1} - x_N) = \varphi(a + (-1)^k\Delta a - y) +$$
$$+ \varphi(a_{M+1} - a - (-1)^{k+1}\Delta a) = \varphi(a + (-1)^k\Delta a) + \varphi(a - (-1)^{k+1}\Delta a) =$$
$$= \varphi(a + \Delta a) + \varphi(a - \Delta a),$$

где $a = a - y = a_{M+1} - a$.

Отсюда, при $x_N = a_i$,

$$\Phi_N(a_i, y) = \varphi(a + \Delta a^*) + \varphi(a - \Delta a^*).$$

Введем в рассмотрение два числа $\alpha_1 = \frac{\Delta a + \Delta a^*}{2\Delta a}$ и $\alpha_2 = \frac{\Delta a - \Delta a^*}{2\Delta a}$. Очевидно, $\alpha_1 + \alpha_2 = 1$ и в силу (26) $0\leqslant\alpha_1,\ \alpha_2\leqslant 1$. Воспользуемся далее свойством выпуклости вниз функции $\varphi(\cdot)$:

$$F(\alpha_1) = \alpha_1\varphi(a + \Delta a) + \alpha_2\varphi(a - \Delta a) \geqslant \varphi(\alpha_1(a + \Delta a) +$$
$$+ \alpha_2(a - \Delta a)) = \varphi(a + \Delta a^*).$$

Аналогично

$$F(\alpha_2) = \alpha_2\varphi(a + \Delta a^*) + \alpha_1\varphi(a - \Delta a) \geqslant \varphi(a - \Delta a^*).$$

Итак,

$$\Phi_N(x_N, y) = \varphi(a + \Delta a) + \varphi(a - \Delta a) = F(a_1) + F(a_2) > \varphi(a + \Delta a^*) + \varphi(a - \Delta a^*) = \Phi_N(a_{i^*}, y).$$

Отсюда в силу произвольности x_N следует

$$\Phi_N(a_{i^*}, y) = \Phi^*_N(y),$$

что и требовалось доказать

Значительное уменьшение объема вычислений можно получить также сокращением множества возможных значений x_p при вычислении функций $\Phi^*_p(y)$.

Теорема 3. При любых значениях p и y_1 и y_2, удовлетворяющих условию

$$a_0 \leqslant y_1 < y_2 \leqslant a_{M-(N-p+1)},$$

справедливо

$$x_p(y_1) \leqslant x_p(y_2),$$

если в (24') $\varphi_p(y_1, x_p) = \varphi(x_p - y)$ — выпуклая вниз функция.

Доказательство Прежде всего заметим, что при $p = N$ утверждение настоящей теоремы следует из предыдущей, поскольку при $y_1 < y_2$

$$(a_{M+1} + y_1)/2 < (a_{M+1} + y_2)/2.$$

Итак, пусть $p < N$ и $x_p(y_1)$ реализует $\Phi^*_p(y_1)$. Представим $\Phi^*_p(y)$ в виде

$$\Phi^*_p(y) = \min_{y < x_p \leqslant a_{M-N+p}} \{\varphi(x_p - y) + \min_{x_p < x_{p+1} \leqslant a_{M-N+p+1}} [\varphi(x_{p+1} - x_p) + \Phi^*_{p+2}(x_{p+1})]\} = \min_{a^* < x_{p+1} \leqslant a_{M-N+p-1}} \{\min_{y < x_p < x_{p+1}} [\varphi(x_p - y) + \varphi(x_{p+1} - x_p)] + \Phi^*_{p+2}(x_{p+1})\},$$

где a^* — наименьшее в A число, бо́льшее y.

Рассмотрим задачу построения функции

$$\min_{y < x_p < x_{p+1}} \{\varphi(x_p - y) + \varphi(x_{p+1} - x_p)\},$$

которая, как легко видеть, сводится к задаче построения функции $\Phi^*_N(y)$, если заменить a_{M+1} на x_{p+1} и a_M на x_{p+1}. Относительно же функции $\Phi^*_N(y)$ утверждение теоремы справедливо, т. е. при $y_1 < y_2$ $x_{p+1}(y_1) \leqslant x_p(y_1)$ для любого фиксированного значения x_{p+1}, в том числе оптимального. Теорема доказана.

Из теоремы следует практическое правило: при расчете функции $\Phi^*_p(y)$ для $y = a_0, a_1, \ldots, a_{M-(N-p+1)}$ множество возможных значений x_p: $y < x_p \leqslant a_{M-N+p}$, $x_p \in A$ может быть сокращено до $y < x'_p \leqslant x_p \leqslant a_{M-N+p}$, где x'_p — оптимальное значение x_p, соответствующее предыдущему значению y. Наконец, для $p = N-1, N-2, \ldots$ из условия $x_p < x_{p+1}$ следует возможность сокращения множества $x'_p \leqslant x_p \leqslant a_{M-N+p}$ до множества $x'_p \leqslant x_p < x_{p+1}$.

Оценим объем памяти V, необходимый для хранения таблицы функций $x_p(y)$, $p = 1, 2, \ldots, N$. Для p-го столбца таблицы требуется

$$V_p = M - N + p$$

ячеек*), следовательно,

$$V = \sum_{p=1}^{N} V_p = \sum_{p=1}^{N} M - N + p = N(M-N) + \frac{1+N}{2} N = \frac{1}{2} N(2M - N + 1).$$

Пример. Проиллюстрируем сказанное выше на примере с $N=5$, $M=15$. В качестве функции $\varphi(x_{i-1}, x_i)$ примем $(x_i - x_{i-1})^2$, а множества A — набор чисел: 0, 1, 3, 6, 10, 11, 12, 18, 20, 22, 26, 35, 36, 40, 41, 45, 50

Значения функций $\Phi^*_p(y)$ и $x^*_p(y)$ сведены в табл. 3. Экстремальное значение функции цели есть $\Phi^*_1(0) = 426$, а само решение имеет вид:

$x_0 = 0$, $x_1 = 10$, $x_2 = 18$, $x_3 = 26$, $x_4 = 35$, $x_5 = 41$, $x_6 = 50$

Табл. 3 позволяет определить оптимальное решение для любого подмножества A_y множества A вида

$$A_y = \{a_i \in A \mid a_i \geqslant y\},$$

где $y \in A$, если число элементов подмножества A_y больше или равно $N+2$. Например, для $y = 10$ минимальное значение функции цели есть $\Phi^*_1(10) = 284$, а само решение имеет вид: $x_0 = 10$, $x_1 = 18$, $x_2 = 26$, $x_3 = 35$, $x_4 = 40$ $x_5 = 45$, $x_6 = 50$. Аналогично для $y = 20$ $\Phi^*_1(20) = 176$, а $x_0 = 20$, $x_1 = 22$, $x_2 = 26$, $x_3 = 35$, $x_4 = 40$, $x_5 = 45$, $x_6 = 50$

Таблица 3

$a=y$	Φ_1, x_1		Φ_2^*	x_2^*	Φ_3^*	x_3^*	Φ_4^*	x_4^*	Φ_5^*	x_5^*
0	1252	26	838	18	636	11	524	10	426	10
1	1201	26	803	18	613	12	505	10	407	10
3	1105	26	739	20	573	12	487	12	375	11
6	976	26	646	20	504	18	406	18	334	12
10	832	26	538	22	424	18	336	18	284	18
11	801	26	515	22	407	22	311	18	269	18
12	754	35	492	26	396	22	298	20	256	20
18	514	35	360	26	262	26	220	26	188	22
20	450	35	332	26	234	26	192	26	176	22
22	394	36	286	35	214	26	172	26	164	26
26	296	40	198	35	156	35	148	35	140	35
35	117	41	75	40	67	36	59	36		
36	106	41	66	41	58	40				
40	50	45	42	41						
41	41	45								
45										

В дальнейших работах предполагается продолжить рассмотрение задач определения режимов профилактики в случаях, когда факт проведения профилактики в момент a_i есть случайное событие, когда ограничение (4) отсутствует и т. д.

Пользуемся возможностью выразить благодарность В. М. Рахвальскому за ценные беседы и И. В. Афанасьевой за большую помощь при проведении расчетов и оформлении рукописи

ЛИТЕРАТУРА

1. Гернбах И. Б. Модели профилактики. Изд-во «Советское радио», 1969.
2. Барзилович Е. Ю. Об оптимальном управлении контролируемым монотонно возрастающим случайным процессом. «Известия АН СССР», Техническая кибернетика, 1966, № 3.

*) При хранении в ячейке одного числа.

3. Барзилович Е. Ю., Захаренко С. К. О сравнении двух оптимальных методов управления случайным процессом. В сб. «О надежности сложных систем». Изд-во «Советское радио», 1966.
4. Волков И. П. Вероятности систем с ожиданием и плановая профилактика их стареющих элементов. Труды ЛИВТ, вып. 39, 1967.
5. Гурич Л. С. и др. Задачи и методы оптимального распределения ресурсов. Изд-во «Советское радио», 1968.
6. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования. Изд-во «Наука», 1965.
7. Хедли Дж. Нелинейное и динамическое программирование. Изд-во «Мир», 1967.
8. Рыжик И. М. Таблицы интегралов, сумм, рядов и произведений. Гостехиздат, 1943.

УДК 681.3—001

НЕКОТОРЫЕ СВОЙСТВА ЗВУКОВЫХ КОДОВ

Г. Д. ФРОЛОВ

В настоящее время решение проблем анализа и синтеза речи занимает важное место в создании автоматизированных систем управления. В связи с этим большое внимание уделяется изучению свойств речевого сигнала.

В настоящей статье приведены некоторые свойства так называемого клиппированного речевого сигнала. Однако, прежде чем перейти к рассмотрению этих свойств, укажем те предполагаемые заранее заданными объекты и отношения, которые служат исходным материалом для описания свойств клиппированного сигнала.

Исходный материал. Будем предполагать, что:

1. Имеется совокупность конкретных звуков данного языка.

2. Всякое исходное произнесение *) представляет собой линейную последовательность конкретных звуков.

3. Каждому исходному произнесению поставлен в соответствие двоичный код $\varkappa_1^{(0)}$, который представляет собой последовательность значений функции

$$\varphi(t)=\begin{cases}1, & \text{если } \varphi(t)\geq 0,\\ 0, & \text{если } \varphi(t)<0,\end{cases}$$

вычисленных с некоторым шагом Δt, где $\varphi(t)$ — функция, определяющая речевой сигнал данного исходного произнесения.

4. Указан транскрипционный алфавит (т. е. список знаков, называемых транскрипционными), содержащий столько же различных знаков, сколько имеется звуков языка. Между транскрипционным алфавитом и совокупностью звуков языка установлено взаимно однозначное соответствие. Транскрипционный знак, соответствующий звуку, объявляется именем этого звука.

5. Каждому исходному произнесению поставлено в соответствие слово, состоящее из букв транскрипционного алфавита, расположенных в том же порядке, в котором расположены соответствующие им звуки в произнесении. Транскрипционную запись исходных произнесений будем называть исходным выражением, а каждое исходное произнесение — произнесением исходного выражения.

6. Предполагается, что на рассматриваемом множестве произнесений, имеющих различные выражения, задано отношение эквивалентности, называемое отношением смысловой эквивалентности. При этом об эквивалентных произнесениях говорят, что они имеют одинаковый смысл.

7. На совокупности конкретных звуков можно определить отношение свободного варьирования. Под отношением свободного варьирования понимается следующее. Пусть P и Q — некоторые цепочки звуков. Будем говорить, что звуки x и y находятся в отношении свободного варьирования в окружении (P, Q), если цепочки PxQ и PyQ либо обе не являются исходными выражениями, либо обе являются исходными выражениями и притом эквивалентными. Точно так же будем говорить, что двоичные коды $\varkappa^{(0)}(x)$ и $\varkappa^{(0)}(y)$, соответствующие звукам x и y, находятся в отношении свободного варьирования в окружении $(\varkappa^{(0)}(P), \varkappa^{(0)}(Q))$, если x и y — звуки, находящиеся в отношении свободного варьирования в окружении (P, Q). Отношение свободного варьирования является отношением типа эквивалентности.

По двоичному коду $\varkappa^{(0)}$ исходного произнесения мы можем построить последовательность $\varkappa=(\varkappa_i)$, каждый элемент которой представляет собой целое положительное число $\varkappa_{2i+1}(\varkappa_{2i+2})$, определяющее количество единиц (нулей), непосредственно следующих друг за другом в коде $\varkappa^{(0)}$ между двумя «соседними» нулями (единицами). Наиболее точное представление о способе построения последовательности $\varkappa=(\varkappa_i)$ можно получить в результате выполнения алгоритма $\mathfrak{A}$.

Пусть $\varkappa^{(0)}=(a_p)_{p=1}^{*}$, где a_p совпадает с одним из чисел 1 или 0.

Алгоритм $\mathfrak{A}$.

1. Выполнить $p:=0$, $i:=0$. Перейти к п. 2.
2. Выполнить $A:=1$. Перейти к п. 3.
3. Выполнить $p:=p+1$. Перейти к п. 4.
4. Проверить условие $p=v+1$. Если условие выполнено, перейти к п. 5, если не выполнено — к п. 6.
5. Закончить процесс.
6. Проверить условие $a_{p-1}=a_p$. Если условие выполнено, перейти к п. 7, если не выполнено — к п. 8.
7. Выполнить $A:=A+1$. Перейти к п. 3.
8. Выполнить $\varkappa_i:=A$, $i:=i+1$. Перейти к п. 2.

Из $\varkappa=(\varkappa_i)$ выделим подпоследовательности $\varkappa^+=(\varkappa_{2i+1})$ и $\varkappa^-=(\varkappa_{2i+2})$. Среди элементов $\varkappa^+(\varkappa^-)$ могут быть равные по величине. Составим множество $M_1(M_2)$ всех различных между собой по величине элементов в $\varkappa^+(\varkappa^-)$. Элементы этого множества будем обозначать символом $\tau_i^+(\tau_i^-)$, а последовательность

$\{(\tau_i^-, \rho_i^+)\}$ ($\{(\tau_i^-, \rho_i^-)\}$), где $\rho_i^+(\rho_i^-)$ — частота **) элемента $\tau_i^+(\tau_i^-)$ в $\varkappa^+(\varkappa^-)$, будем называть частотным спектром кода $\varkappa^+(\varkappa^-)$.

Сегментирование кода $\varkappa$. Задача сегментирования исходного произнесения является одной из центральных в проблемах анализа и синтеза речи. Существо этой задачи заключается в разбиении заданного исходного произнесения (или, что то же самое, двоичного кода $\varkappa^{(0)}$ или, наконец, кода $\varkappa$) на составные части таким образом, чтобы каждая такая часть соответствовала конкретному звуку, из которых состоит данное исходное произнесение (двоичный код $\varkappa^{(0)}$ или код $\varkappa$).

Ниже приведено описание приближенного способа решения этой задачи.

*) Как и в [1], под исходным произнесением понимается некоторый отрезок устной речи. Тем самым под исходным произнесением мы также понимаем физическое явление, имеющее некоторую вполне определенную локализацию во времени и пространстве.

**) Частотой мы в данном случае называем количество одинаковых элементов в данной последовательности.

Код x разобьем на участки $\Delta_k^{(l)}$ ($k=1, 2, \ldots$) длиной $2l$ (кроме последнего, длина которого может быть меньше величины $2l$). Для каждого такого участка вычислим значение $f_1^{(k)}$ функции

$$f_1^{(k)} = \frac{1}{l} \sum_{i=(k-1)l}^{kl-1} x_{2i+2}.$$

Затем вычислим величины $\Delta f_1^{(k)} = |f_1^{(k)} - f_1^{(k+1)}|$. Все участки $\Delta_k^{(l)}$, непосредственно следующие друг за другом в x, объединим в сегменты (сегменты первого типа), если для каждой последовательной пары этой последовательности $\Delta f_1^{(k)} < P$. Точно так же объединим в сегменты (сегменты второго типа) участки $\Delta_k^{(l)}$, непосредственно следующие друг за другом в коде x, для которых $\Delta f_1^{(k)} > P$.

Значения параметров P и l примем равными соответственно числам 2 и 16. Эти значения параметров выбраны в результате статистического исследования большого количества кодов x различных исходных произнесений (для дикторов обоего пола).

Описанный способ сегментирования, как указывалось выше, является приближенным. Среди ошибок, порождаемых этим способом, могут быть и такие, которые определяются параметрами l и P. Ошибки первого типа могут быть устранены путем выполнения следующей процедуры. Пусть граница двух смежных сегментов проходит между участками $\Delta_k^{(l)}$ и $\Delta_{k+1}^{(l)}$.

Возможны два случая: 1) когда $\Delta_k^{(l)}$ принадлежит сегменту первого типа, 2) когда $\Delta_k^{(l)}$ принадлежит сегменту второго типа. В соответствии с этим уточнение указанной границы возможно двояко. Рассмотрим участок ∇_p, полученный из $\Delta_{k+1}^{(l)}$ в первом случае (и $\Delta_k^{(l)}$ — во втором случае) путем отбрасывания последних (первых) $2p$ точек. Вычислим величину

$$\Delta f_p = |f_1^{(k)} - f_p| (\Delta f_p = |f_1^{(k+1)} - f_p|),$$

где $f_p = \frac{1}{l-2p} \sum_{x_{2i+1} \in \nabla_p} x_{2i+2}$, и проверим выполнение условия $\Delta f_p < 2$. Если окажется, что $\Delta f_p < 2$, то сегмент, содержащий $\Delta_k^{(l)}$ ($\Delta_{k+1}^{(l)}$), расширим, включив в него участок ∇_p, а сегмент, содержащий $\Delta_{k+1}^{(l)}$ ($\Delta_k^{(l)}$), соответственно уменьшим, исключив из него участок ∇_p. Если же $\Delta f_p > 2$, то повторим указанную процедуру для следующего значения величины p ($p = 1, 2, \ldots, 7$).

О частичном устранении ошибок второго типа будет сказано ниже.

Анализируя эмпирические данные о результатах сегментирования указанным способом применительно к совокупности исходных звуков, определенной исходя из задач лингвистической практики, следует отметить, что всю совокупность исходных звуков можно разбить на три группы. К первой группе относятся все те звуки, каждому из которых в результате сегментирования исходного произнесения ставится в соответствие один сегмент первого типа, ко второй группе — все те звуки, каждому из которых в результате сегментирования будет поставлен в соответствие лишь один сегмент второго типа и, наконец, к третьей группе — все остальные звуки. Таким образом указанный способ сегментирования дает отклонение от нормы сегментирования для звуков третьей группы. Причем для разных дикторов эти звуки разные. Чаще всего третью группу составляют согласные звуки: [ш], [ш'], [ч'], [ц], [с], [с'], [х], [х'], [ф], [ф'] и реже звуки [з], [з'], [ж] и совсем редко гласные звуки *).

Некоторые свойства звуковых кодов. Статистические исследования позволили установить ряд свойств кодов x, наиболее интересные из которых приведены ниже.

Пусть PxQ — цепочка звуков, соответствующая некоторому исходному произнесению, а $x(P)$, $x(x)$ и $x(Q)$ — коды, соответствующие звукам P, x и Q. В этих предположениях имеет место:

1°. Если в коде $x(x)$ произвести замену элемента $x_{2i+1}(x_{2i+2})$ на элемент $x_{2i+2}(x_{2i+1})$ для данного i, удовлетворяющего неравенству $0 \leqslant i \leqslant m-1$, где $2m$ — длина кода, то получим код $\bar{x}(\bar{x})$, находящийся с данным кодом в отношении свободного варьирования в окружении $(x(P), x(Q))$.

Если учитывать, что отношение свободного варьирования есть отношение типа эквивалентности, то из 1° следует справедливость следующего значения.

Замечание 1. Коды $x(x)$ и $\bar{x}(\bar{x})$ находятся в отношении свободного варьирования в окружении $(x(P), x(Q))$, если элементы кода $x(\bar{x})$ вычислены следующим образом:

$$\bar{x}_{2i+1} := x_{2i+2} \text{ и } \bar{x}_{2i+2} := x_{2i+1}$$

для всех $0 \leqslant i \leqslant m-1$.

2°. Если в коде $x(x) = x_1, \ldots, x_{2i}, x_{2i+1}, x_{2i+2}, \ldots, x_{2m}$ элемент $x_{2i+1} \geqslant A$, где $A = \text{const}$, то код $x(\bar{x}) = x_1, \ldots, x_{2i}, A, x_{2i+2}, \ldots, x_{2m}$ находится с данным кодом в отношении свободного варьирования в окружении $(x(P), x(Q))$. Величина A определяется также локализацией во времени и пространстве, которая определяет и соответствующее исходное произнесение данного кода $x(x)$.

Замечание 2. Из 1° и 2° следует, что для всякого кода $x(x)$ можно указать код $\bar{x}(\bar{x})$, который находится с данным кодом в отношении свободного варьирования в окружении $(x(P), x(Q))$ и в котором каждый элемент $x_{2i+1} < A$, а $x_{2i+2} \leqslant A$ для всех $0 \leqslant i \leqslant m-1$, где $2m$ — длина кода $x(x)$.

3°. Коды $x(x)$ и $x(\bar{x})$ находятся в отношении свободного варьирования в окружении $(x(P), x(Q))$, если $x(x)$ — код, соответствующий сегменту первого типа, а $\bar{x}(\bar{x})$ — код, элементы которого вычисляются по правилам $\bar{x}_i = x_{2m-i}$ для всех $0 < i < 2m$, где $2m$ — длина кода $x(x)$.

4°. Пусть $x(x)$ соответствует сегменту первого типа со спектром $\{(\tau_i^+, \rho_i^+)\}$ положительной составляющей части. Причем элементы $\tau_1^+, \ldots, \tau_{k_1}^+$ таковы, что $\rho_1^+ \geqslant \ldots \geqslant \rho_{k_1}^+$. Пусть далее $\tau_{i_1}^+ > \tau_{i_2}^+ > \tau_{i_3}^+$, где $i_1, (i_2, i_3)$ совпадает с одним из чисел 1, 2, 3 и $i_p \neq i_q$ при $p \neq q$. Тогда коды $x(x)$ и $x(\bar{x})$ находятся в отношении свободного варьирования в окружении $(x(P), x(Q))$, если элементы кода $x(\bar{x})$ вычислены по правилам: $\bar{x}_{2i+2} := x_{2i+2}$; $\bar{x}_{2i+1} := x_{2i+1}$, если x_{2i+1} совпадает с одним из $\tau_{i_1}^+$, $\tau_{i_2}^+$ и $\tau_{i_3}^+$; $\bar{x}_{2i+1} := \tau_{i_1}^+$, если $x_{2i+1} > \tau_{i_1}^+$; $\bar{x}_{2i+1} := \tau_{i_2}^+$, если $x_{2i+1} < \tau_{i_2}^+$ и, наконец, $\bar{x}_{2i+1} := a$, если $\tau_{i_3}^+ < x_{2i+1} < \tau_{i_1}^+$ и $x_{2i+1} \neq \tau_{i_p}^+$ ($p = 1, 2, 3$), где a совпадает с одним из τ_{i_p} и $|a - x_{2i+1}| = \min |\tau_{i_p} - x_{2i+1}|$.

*) Символ «'» означает смягчение звука.

Замечание 3. Из 1° и 4° следует, что коды $\varkappa(x)$ и $\varkappa(\tilde{x})$ находятся в отношении свободного варьирования в окружении $(\varkappa(P), \varkappa(Q))$, если элементы кода $\varkappa(\tilde{x})$ вычислены по способу, описанному в 4°.

5°. Коды $\varkappa(x)$ и $\varkappa(\tilde{x})$ находятся в отношении свободного варьирования в окружении $(\varkappa(P), \varkappa(Q))$, если $\varkappa(x)$ — код, соответствующий сегменту $[a, b]$ второго типа, а $\varkappa(\tilde{x})$ — код, элементы которого вычисляются по правилам $\tilde{x}_{2i+1} := x_{2i+1}$ для всех $a \leqslant 2i+1 \leqslant b$,

$$\tilde{x}_{2i+2} := \begin{cases} x_{2i_1+2} := \max\limits_{a \leqslant 2i+2 \leqslant b} x_{2i+2}, \\ x_{2i_2+2} := \max\limits_{a \leqslant 2i+2 < 2i_2+2} x_{2i+2} & \text{для всех } i_2 \leqslant 2i+2 < i_1, \\ x_{2i_3+2} := \max\limits_{2i_2+2 < 2i+2 \leqslant b} x_{2i+2} & \text{для всех } i_3 < 2i+2 \leqslant i_2, \\ x_{2i_4+2} := \max\limits_{a \leqslant 2i+2 < 2i_3+2} x_{2i+2} & \text{для всех } i_4 \leqslant 2i+2 < i_3, \\ x_{2i_5+2} := \max\limits_{2i_4+2 < 2i+2 \leqslant b} x_{2i+2} & \text{для всех } i_5 < 2i+2 \leqslant i_4, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{cases}$$

6°. Коды $\varkappa(x)$ и $\varkappa(\tilde{x})$ находятся в отношении свободного варьирования в окружении $(\varkappa(P), \varkappa(Q))$, если $\varkappa(x)$ — код, соответствующий сегменту первого типа, а $\varkappa(\tilde{x})$ — код, элементы которого вычисляются в результате выполнения алгоритма $\mathfrak{B}$.

Алгоритм $\mathfrak{B}$

1. Выполнить $k := 0$. Перейти к п. 2.
2. Выполнить $k := k+1$, $l := 0$. Перейти к п. 3.
3. Проверить условие $k > K$, где K — количество элементов спектра $\{(\tau^k, \rho^k)\}$. Если условие выполнено, то перейти к п. 4; если не выполнено — к п. 5.
4. Закончить процесс.
5. Выполнить $i := 0$. Перейти к п. 6.
6. Выполнить $i := i + \mathrm{E}\left[\left(\sum\limits_{k=1}^{K} \rho_k\right)/\rho_k\right]$, где $\mathrm{E}\left[\left(\sum\limits_{k=1}^{K} \rho_k\right)/\rho_k\right]$ — целая часть числа $\left(\sum\limits_{k=1}^{K} \rho_k\right)/\rho_k$. Перейти к п. 7.
7. Проверить условие $i > m+1$. Если условие не выполнено, перейти к п. 8; если выполнено — к п. 5.
8. Проверить условие: не отмечен ли элемент $\tilde{x}_i$. Если условие не выполнено, перейти к п. 9; если выполнено — к п. 11.
9. Выполнить $\tilde{x}_i := \tau_k$, $l := l+1$. Отметить элемент $\tilde{x}_i$. Перейти к п. 10.
10. Проверить условие $l = \rho_k$. Если условие не выполнено, перейти к п. 6, если выполнено — к п. 2.
11. Выполнить $j := i$. Перейти к п. 12.
12. Выполнить $j := j+1$. Перейти к п. 13.
13. Проверить условие $j > m$. Если условие не выполнено перейти к п. 15, если выполнено — к п. 14.
14. Выполнить $j := 0$. Перейти к п. 12.

15. Проверить условие: не отмечен ли элемент $\tilde{x}_j$. Если условие не выполнено, перейти к п. 16; если выполнено — к п. 12.
16. Выполнить $\tilde{x}_j := \tau_k$, $l := l+1$. Отметить элемент $\tilde{x}_j$. Перейти к п. 10.

7°. Существует такое число B, что если согласный звук x — звук третьей группы и $\{(\tau_i^+, \rho_i^+)\}$ — частотный спектр кода $\varkappa(x)$, входящего в состав кода, соответствующего звуку x, то $\tau_i^+ < B$*), а если x — гласный звук, то $\tau_i^+ > B$ для $i = 1, 2, 3$; (τ_i^+ таковы, что $\rho_1^+ \geqslant \rho_2^+ \geqslant \rho_3^+ \geqslant \ldots$).

8°. Всякий код $\varkappa(x)$, соответствующий взрывному звуку x, как правило, имеет в своем составе элемент $x_{2i+2} \geqslant A$.

9°. Если $\varkappa(x)$ — код, соответствующий сегменту первого типа, то $\tau_i < A$ ($i = 1, 2, 3$; $\rho_1 \geqslant \rho_2 \geqslant \rho_3 \geqslant \ldots$), где τ_i — элемент частотного спектра $\{(\tau_i^+, \rho_i^+)\}$ ($\{(\tau_i^-, \rho_i^-)\}$) кода $\varkappa(x)$.

10°. Пусть $\Phi(z) = E[mf(z)]$, где $f(z)$ — функция вычисления псевдослучайных чисел, решающий алгоритм которой приведен в [2, стр. 414, программа № 1] в качестве стандартной программы, m — целое положительное число ($m \geqslant 1$). Конкретное значение m определяется рядом обстоятельств, о которых будет сказано ниже; $E[mf(z)]$ — целая часть числа $mf(z)$. Пусть далее $\varkappa(x)$ — код, соответствующий сегменту первого типа.

Составим частотный спектр $\{(\tau_i^+, \rho_i^+)\}$ положительной составляющей части кода $\varkappa(x)$. Из элементов τ_i^+ составим конструкции $l_1, \ldots, l_n$, каждая из которых есть либо один из элементов τ_i^+, либо некоторая совокупность элементов τ_i^+, представленная в виде некоторой последовательности, если эта совокупность повторяется в положительной составляющей части кода $\varkappa(x)$ несколько раз. Причем конструкции l_q строятся таким образом, чтобы, во-первых, среди элементов x_{2i+1} кода $\varkappa(x)$ не было таких, которые не принадлежали бы ни одной из конструкций l_q, и, во-вторых, каждой из конструкций l_q ставится в соответствие величина ρ_q^+, называемая частотой конструкции, которая с величиной ρ_i^+ находится в отношении

$$\rho_q^+ = \sum_{\cdot} \rho_{i_\nu}^+ \quad \text{для всех } q = 1, 2, \ldots, \tag{1}$$

где $\rho_{i_\nu}^+$ — частота конструкции l_{j_ν}, содержащая в своем составе элемент τ_q^+.

Задавшись некоторым z_0, вычислим последовательность функции $\Phi(z_0)$, из которой выделяется подпоследовательность $\{y_{k_p}\}$ элементов, удовлетворяющих неравенству $1 \leqslant y_{k_p} \leqslant n$. И, наконец, определим положительную часть кода $\varkappa(x)$ путем составления последовательности $\{l_{q_\tau}\}$, где $q_\tau = y_{k_p}$. Выбор числа членов этой последовательности, а следовательно, и подпоследовательности $\{y_{k_p}\}$ определяется соотношением (1). Коэффициент m примем равным величине n.

Точно так же вычислим отрицательную составляющую кода $\varkappa(\tilde{x})$. В таком случае коды $\varkappa(x)$ и $\varkappa(\tilde{x})$ находятся в отношении свободного варьирования в окружении $(\varkappa(P), \varkappa(Q))$.

*) Величина B, как и A, определяется теми же обстоятельствами, которыми определено и данное исходное произвесение.

Некоторые из указанных свойств звуковых кодов могут быть использованы и для уточнения самой процедуры сегментирования.

Как уже указывалось, при сегментировании наибольшее отклонение от нормы наблюдается для звуков третьей группы. С другой стороны, элементы кодов $x(x)$, именно звуков третьей группы, обладают свойством 7°. В таком случае мы можем, используя это свойство, ввести процедуру объединения двух смежных сегментов кода $x(x)$, если элементы этих сегментов обладают свойством 7° Вероятно, этим же свойством обладают и элементы звуковых кодов некоторых взрывных звуков, но элементы кодов последних звуков обладают, как правило, и свойством 8°, которое в большинстве случаев отличает коды взрывных звуков от звуков третьей группы Таким образом, если после сегментирования выполнять вышеуказанную процедуру объединения сегментов, то результаты сегментирования в большинстве случаев улучшаются.

ЛИТЕРАТУРА

1. Успенский В А. Одна модель для понятия фонемы. «Вопросы языкознания», 1964, № 6.
2 Лашенко В. Ф Программирование для цифровых вычислительных машин М 20, БЭСМ 3М, БЭСМ-4, М 220 Изд-во «Советское радио», 1967.

УДК 681.322

ПРИМЕНЕНИЕ ЭВМ ДЛЯ ИССЛЕДОВАНИЯ НЕКОТОРЫХ ХАРАКТЕРИСТИК КЛИППИРОВАННЫХ РЕЧЕВЫХ СИГНАЛОВ

Г. Н. РУСЕЦКИЙ, Ю. П. СКОКАН

Использование электронных вычислительных машин (ЭВМ) в автоматизации определенных областей деятельности человека делает актуальным вопрос о способах обмена информацией между машиной и человеком. Очевидно, что одной из удобных форм такого обмена можно считать передачу информации с помощью устной речи. Поэтому практически интересной становится проблема автоматического опознания речевых сигналов. Несомненно, что для решения этой проблемы необходимо исследование речевых сигналов.

Результаты измерения целого ряда характеристик речевых сигналов приведены в работе [1]. Следует указать, что в этой работе характеристики в основном измерялись не непосредственно на акустическом сигнале, а с помощью фиксирования положений речевого тракта.

Для непосредственного исследования речевых сигналов может быть использован метод спектрального анализа. На результаты исследования данным методом накладываются определенные ограничения, вызываемые применением резонансных фильтров. Ограничения объясняются принципом неопределенности — зависимостью между полосой частот, пропускаемой фильтром и скоростью изменения сигнала на выходе фильтра [2]. Наиболее примечательной чертой методов исследования клиппированных сигналов является то, что результаты исследования не зависят от ограничений, вызываемых принципом неопределенности. Разборчивость клиппированного речевого сигнала лишь немного ниже разборчивости обычной речи [3]. Незначительное влияние операций клиппирования на разборчивость речи позволяет предположить, что значительная часть информации содержится в нулях временной функции речевого сигнала [4].

В силу того, что повторение в одинаковых условиях одного и того же речевого сигнала не приводит к полному совпадению этих сигналов, можно полагать, что в речевых сигналах присутствует элемент случайности. В свою очередь, факт распознавания речевых сигналов человеком дает основание считать, что в них существуют некоторые инвариантные закономерности. Выявление подобных закономерностей, проявляющихся в массе случайных явлений, можно осуществить с помощью статистической обработки количественных характеристик речевого сигнала. Увеличение количества речевых сигналов, подвергнутых статистической обработке, неизбежно приведет к более отчетливому проявлению присущих им закономерностей и позволит с большей точностью определить количественные значения их характеристик.

Накопление и статистическая обработка большого количества информации делают необходимым использование ЭВМ в качестве основного инструмента при экспериментальном исследовании характеристик речевого сигнала. Использование ЭВМ позволяет моделировать любой известный метод обработки речевых сигналов. В отличие от применения специализированной аналоговой аппаратуры или ручных методов обработки информации использование ЭВМ значительно облегчает как сам процесс исследования, так и переход от одной методики исследования к другой.

Для непосредственного исследования речевого сигнала в ЭВМ необходимо обеспечить процесс ввода сигнала в машину.

В данной статье приводится методика и результаты измерений с помощью ЭВМ некоторых характеристик клиппированного речевого сигнала. Способ ввода речевого сигнала в машину и его клиппирование состоит в следующем. С помощью микрофона и усилителя акустический сигнал преобразуется в напряжение, достаточное по величине для управления логическим элементом И. Подача на один вход элемента И такого управляющего напряжения, а на другой — периодической последовательности маркерных импульсов дает возможность получить на выходе данного элемента последовательность пачек импульсов. Пачка импульсов получается при условии, что управляющее напряжение, соответствующее акустическому сигналу, превосходит по своей величине уровень открывания элемента И. Таким образом, пачки импульсов соответствуют положительным полуволнам акустического сигнала. Количество импульсов в пачке отражает длительность этой полуволны. В свою очередь, паузы между пачками импульсов соответствуют отрицательным полуволнам акустического сигнала, а длительности пауз равны длительностям этих полуволн. Паузы можно рассматривать как пачки «нулевых» импульсов. Переходы между пачками соответствуют нулям временной функции речевого сигнала. Процесс записи в запоминающие устройства ЭВМ подобных пачек импульсов, соответствующих положительным и отрицательным полуволнам акустического сигнала, не вызывает принципиальных трудностей [5].

Запись акустического сигнала в ЭВМ представляет собой последовательность целых положительных чисел, равных количеству маркерных импульсов, укладывающихся в соответствующих полупериодах акустического сигнала. При этом все члены такой последовательности с нечетными номерами соответствуют положительным полуволнам акустического сигнала, а с четными — отрицательным. Таким образом, эта последовательность содержит информацию о длительностях полуволн акустического сигнала и поэтому может быть названа числовой формой представления клиппированного сигнала. Процесс получения числовой формы представления клиппированного речевого сигнала проиллюстрирован на рис. 1. В описываемых экспериментах частота следования маркерных импульсов была выбрана равной 6,6 кГц. Таким образом, максимальная частота сигнала, которую можно было записать в машину, составляла 3,3 кГц.

Для исследования характеристик речевого сигнала были выполнены программы, работающие в едином комплексе. Эти программы позволяют:

— вводить в ЭВМ речевой сигнал;
— выводить речевой сигнал на контрольное прослушивание;
— выводить на печать числовую форму клиппированного акустического сигнала;
— осуществлять визуализацию клиппированных сигналов путем их графического представления;
— членить речевой сигнал на машинные слоги;
— определять длительность речевых сигналов и машинных слогов;
— производить статистическую обработку временных характеристик речевого сигнала (определять оценки математического ожидания, дисперсии, среднего квадратического отклонения)

Ввод речевого сигнала в ЭВМ осуществлялся непосредственно у пульта машины через микрофон. При этом момент появления речевого сигнала автоматически фиксировался машиной по первому колебанию поступающему в микрофон акустического сигнала. Таким образом, диктор мог произвольно выбирать момент начала речи, что создавало определенные удобства в работе с описываемым комплексом программ.

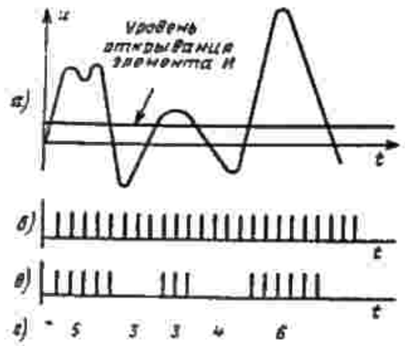


Рис. 1. Процесс получения числовой формы представления клиппированного сигнала
а — напряжение на выходе микрофона; б — последовательность маркерных импульсов; в — напряжение на выходе элемента И; г — последовательность чисел, отображающая числовую форму представления клиппированного сигнала

Вывод речевого сигнала на контрольное прослушивание в процессе проведения исследования осуществлялся с целью проверки качества введенных в машину сигналов. Эта проверка необходима для изъятия из рассмотрения речевых сигналов, записанных на фоне существенных помех. Причиной возникновения помех могут являться условия, в которых проводится эксперимент. Возможность прослушивания создает определенные удобства в работе оператора и с предварительно накопленным в машине материалом.

Контрольное прослушивание необходимо для оценки влияния преобразований в сигнале на их восприятие. Оценка существенности выявленных свойств сигнала довольно эффективно и просто проверяется с помощью контрольного прослушивания. Так, например, выявление и отбор свойств сигнала, на основании которых можно осуществлять его сегментирование, облегчается с помощью контрольного прослушивания выделенных сегментов. В добавление к сказанному следует указать, что ЭВМ, на которой предусмотрено контрольное прослушивание, может быть использована в психоакустических экспериментах в качестве источника акустических сигналов, позволяющего достаточно гибко и точно изменять клиппированный сигнал вплоть до отдельных его колебаний.

Вывод на печать числовой формы клиппированного сигнала для удобства работы выполнялся так, что числа выдавались парами, образующими столбец. Левая колонка столбца соответствовала длительностям положительных полуволн акустического сигнала, правая — отрицательным (рис. 2). Принципиально выдача может быть осуществлена в любой форме.

Визуализация клиппированного речевого сигнала выполнялась с целью облегчения исследования структуры сигналов. Слуховое восприятие человека — это скорее инструмент для опознания слуховых образов в целом, а не средство для исследования их элементов. Зрительное распознавание видимых объектов составляет значительную часть повседневной деятельности человека. Понимаемое в этом смысле познавание включает в себя как необходимую часть выявление закономерностей в различных совокупностях данных. Синтетические и творческие аспекты зрительного человеческого восприятия делают его гибким и эффективным инструментом исследования визуализированного речевого сигнала. Поскольку человек достаточно совершенен в области распознавания зрительных образов, результаты его зрительного восприятия могут оказать существенную помощь в выявлении закономерностей, присущих речевым сигналам.

6	10	4	2	5	7	5	5	4	6	4	6	4	6	4	5	3	8
5	28	1	6	5	7	4	5	4	6	4	6	4	5	4	6	2	23
7	11	2	13	5	8	5	6	4	6	3	8	4	6	3	7	3	8
1	31	5	8	2	3	4	6	4	2	1	5	3	8	1	15	2	11
2	47	4	8	4	7	5	5	2	5	4	6	4	6	3	7	2	16
2	138	4	11	4	7	4	6	5	5	4	5	4	6	4	5	2	39
2	43	4	8	4	7	4	7	5	6	5	5	4	5	3	7	1	
4	16	4	7	3	4	4	6	5	4	3	7	4	6	1	26		
1	22	5	7	4	6	4	6	5	11	2	7	2	12	4	7		
4	9	2	2	5	6	4	7	4	6	4	6	4	6	5	8		

Рис. 2. Числовое представление клиппированного сигнала.

Числовая форма речевого сигнала отображает поэлементную запись этого сигнала в машине, однако для более полного использования возможностей зрительного восприятия человека речевые сигналы целесообразно представлять в виде некоторых контурных объектов, сохраняя при этом детальную информацию о речевом сигнале. Визуализация числовой формы клиппированного речевого сигнала выполнялась с помощью двухкоординатного регистрирующего прибора. При этом величина каждой положительной полуволны объединялась в пару с величиной следующей за ней отрицательной полуволны. На графике длительности полуволн представлялись ординатами пар точек, соединенных прямой. Причем, длительность положительной полуволны представлялась положительной ординатой, отрицательной полуволны — отрицательной. Пример графика слова «два» (числовое представление которого приведено на рис. 2) изображен на рис. 3.

Членение речевого сигнала на машинные слоги производилось на основании содержания числовой последовательности, представляющей речевой сигнал. Согласно выводу Чистович [1] была принята гипотеза о том, что на артикуляторном уровне слог оканчивается на гласный, группа согласных, предшествующих гласному, относится к тому же слогу, что и гласный.

При разработке алгоритма членения речевого сигнала на слоги был проведен целый ряд экспериментов с целью выявления свойств сигнала, на основании которых можно проводить такое членение. Как показали эксперименты, в клиппированном речевом сигнале большинство согласных характеризуются большими величинами длительностей отрицательных полуволн. На рис. 4—7 представлены результаты визуализации участков клиппированных речевых сигналов, содержащих в своем составе некоторые согласные. Рис. 4 и 5 иллюстрируют свойства звука «р» и его ближайшего окружения, взятых из двух реализаций слова «орел». Свойства звука «в» и его ближайшего окружения, взятые из двух реализаций слова «Москва», приведены на рис. 6 и 7.

Указанные свойства согласных в клиппированном речевом сигнале были положены в основу алгоритма автоматического членения сигнала слова на слоги. Контрольное прослушивание речевого сигнала, расчлененного автоматически на слоги, показало, что звучание полученных таким образом машинных слогов в большинстве случаев соответствует звучанию слогов, получаемых согласно гипотезе слогового деления на артикуляторном уровне

С помощью описываемого комплекса программ производилось измерение некоторых временных характеристик клиппированных речевых

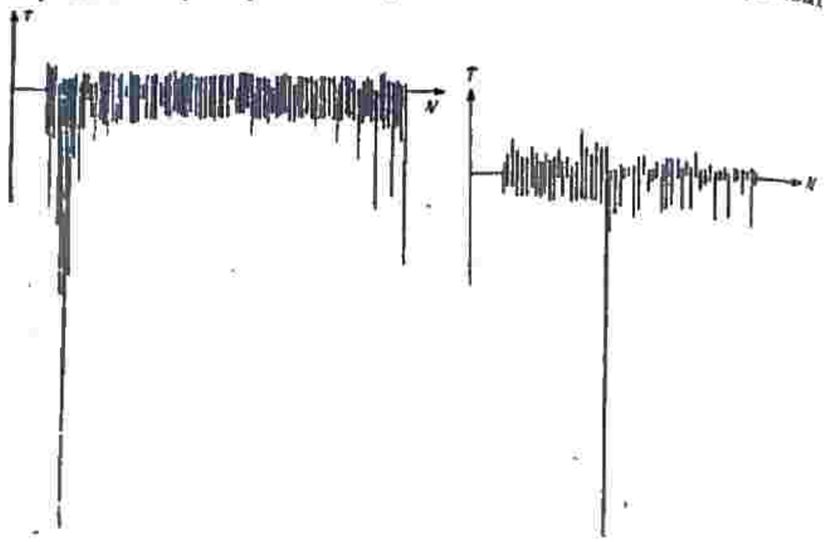


Рис. 3.

Рис. 4.

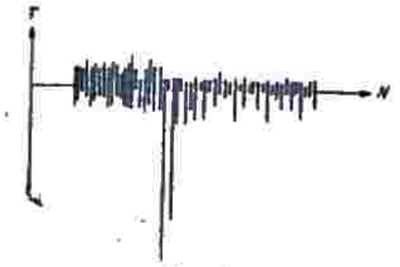


Рис. 5.

Рис. 6.

сигналов. При этом делалась попытка найти наиболее стабильные характеристики, которые можно с помощью машины выделить из речевого сигнала и использовать в качестве признаков при автоматическом опознании. С этой целью результаты измерений подвергались статистической обработке

Материал, подлежащий статистической обработке, разбивался на l групп по n_j реализаций в j-й группе. Для количественной временной характеристики A клиппированного речевого сигнала определялись:

1. Оценка математического ожидания A в j-й группе:

$$\bar{M}[A]_j = \frac{1}{n_j} \sum_{i=1}^{n_j} A_{ij},$$

где A_{ij} — значение характеристики A в i-й реализации j-й группы; n_j — количество реализаций в j-й группе.

2. Оценка дисперсии относительных отклонений характеристики A:

$$D[\delta A] = \frac{\sum_{j=1}^{l} \sum_{i=1}^{n_j} \delta A_{ij}^2}{l\left(\sum_{j=1}^{l} n_j\right) - 1},$$

$$\delta A_{ij} = \frac{A_{ij} - \bar{M}[A]_j}{\bar{M}[A]_j}.$$

В качестве количественных характеристик использовались:

а) длительность речевого сигнала;

б) количество составляющих в числовой форме клиппированного речевого сигнала;

Рис. 7.

в) длительность машинных слогов;

г) относительная (по отношению к длительности речевого сигнала) длительность машинных слогов.

Для статистической обработки была использована выборка по пяти дикторам. Каждый диктор произносил в один сеанс десять различных слов (фраз), повторяя каждое слово (фразу) по 10 раз. Каждый диктор провел по 5 сеансов на протяжении полугода. Речевые сигналы, произнесенные дикторами, записывались и хранились на магнитных лентах. Для ввода в машину и записи на ленту 100 речевых сигналов требовалось не более 15 мин машинного времени, а время членения на слоги и подсчета их длительностей для одного сигнала не превышало 1 с. Следует указать, что дикторы произносили слова в разговорном стиле и произвольным темпом. Статистическая обработка полученного таким образом материала проводилась на ЭВМ по приведенным выше формулам. Для статистической обработки были использованы 2500 реализаций речевого сигнала различных слов. Результаты статистической обработки приведены в таблице. Как видно из величин дисперсий, боль-

Таблица результатов статистической обработки

В группу объединены однотипные речевые сигналы	Дисперсия длительности речевого сигнала	Дисперсия количества составляющих в числовой форме клиппированного речевого сигнала	Дисперсия длительности машинных слогов	Дисперсия относительной длительности машинных слогов
Одного диктора в один сеанс . .	0,0017	0,0060	0,0080	0,0055
Одного диктора в разных сеансах	0,0066	0,0195	0,0146	0,0082
Разных дикторов в разные сеансы	0,0108	0,0395	0,0288	0,0161

шей стабильностью обладает длительность слова. В предположении нормального закона распределения для длительностей речевого сигнала максимальное отклонение длительности слова (фразы), произнесенного диктором в один сеанс, не превысит 12—13%. Это же отклонение, но в разных сеансах одного и того же диктора увеличится до 24—25%. Максимальное отклонение длительности слова (фразы) разных дикторов в разные сеансы составляет 31—33%. Большинство разных речевых сигналов имеет различную длительность, поэтому длительность слова может служить одним из признаков автоматического распознавания речевых сигналов. Возможность автоматического членения на слоги позволяет также использовать их количественных характеристики в качестве признаков опознания.

ВЫВОДЫ

1. Для всестороннего исследования характеристик речевого сигнала с целью разработки алгоритма его машинного распознавания целесообразно использовать электронную вычислительную машину.

2. Простота ввода в машину и отсутствие ограничений в результатах исследования, накладываемых принципом неопределенности, а также скорость машинной обработки делают актуальным исследование, а может быть даже и опознавание клиппированных речевых сигналов на ЭВМ.

3. Результаты статистической обработки клиппированных речевых сигналов показывают, что их количественные характеристики могут быть использованы в качестве количественных признаков опознавания.

ЛИТЕРАТУРА

1. Чистович Л. А. и др. Речь: артикуляция и восприятие. Изд-во «Наука», 1965.
2. Gabor D. Theory of communication, pt. 2. The analysis of hearing. J. Inst. Electr. Engrs. Nov. 1946, v. 93, pt. 3, № 26.
3. Licklider J. C. R., Pollack I. Effects of Differentiation, Integration and Infinite Peak Clipping Upon the Intelligibility of Speech. J. Acoust. Soc. Am. 20, 1948.
4. Фланаган Д. Л. Анализ, синтез и восприятие речи. Изд-во «Связь», 1968.
5. Скокан Ю. П., Сухов А. М., Фролов Г. Д. Запись и воспроизведение последовательности прямоугольных импульсов при помощи машины М-20. «Сб. научных трудов МО», 1967, № 78.

УДК 681.3.51

ОПТИМАЛЬНЫЕ СТРУКТУРЫ ОПТОЭЛЕКТРОННЫХ СИСТЕМ ПОИСКА И РАСПОЗНАВАНИЯ

Г. П. КАТЫС, С. Е. ЗДОР, В. Б. ШИРОКОВ

В настоящее время большинство разработок в области оптоэлектроники связано с улучшением технических характеристик и параметров отдельных элементов и узлов. Достижение высоких показателей работы оптоэлектронных систем в значительной мере определяется использованием результатов, полученных в ряде специальных разделов современной физики. Именно обращение к физической стороне многих явлений и эффектов позволило создать набор оптоэлектронных средств получения, переработки, хранения и представления информации, успешно конкурирующих с электронными и другими устройствами, выполняющими аналогичные целевые функции. Распространение оптоэлектронных методов обусловлено в первую очередь применением лазерной техники, диффузионных свойств и явлений в p-n переходах полупроводников, волоконной техники, голографии, кристаллов и веществ с определенными электрооптическими свойствами и т. д.

Наряду с элементным и физическим подходами в оптоэлектронике возможен и другой подход, который видимо также не лишен перспектив и может составить предмет обсуждения. Этот подход связан с абстрагированием от элементных и схемных решений и заключается в рассмотрении оптимального синтеза оптоэлектронных систем с точки зрения методики построения рациональных структур, оптимизированных по каким-либо информационным критериям.

Среди достоинств и преимуществ оптоэлектроники, заключающихся в быстродействии, помехозащищенности и т. д., вероятно, на одно из первых мест необходимо поставить двумерный характер оптического сигнала, т. е. в оптоэлектронную систему можно заложить не только такие категории как «больше — меньше», «да — нет», но также и категории «выше — ниже», «влево — вправо».

Двумерность оптического сигнала наиболее существенно проявляется в таких областях, как представление информации, преобразование изображения, вычислительные и логические операции, основанные на оптическом принципе, поиск и слежение в оптическом диапазоне, распознавание изображений и др. [1].

Оптимизация оптоэлектронных структур с учетом двумерности оптической информации в настоящей работе иллюстрируется на примерах задач поиска и распознавания.

1. ОПТИМАЛЬНЫЕ ПОИСКОВЫЕ СТРУКТУРЫ

Поиск в оптическом диапазоне может быть представлен как просмотр некоторого непрерывного поля (пространства) излучений, либо поля, на котором имеются изолированные световые пятна, т. е. лучистая энергия является пространственно распределенной. При этом основные задачи поиска могут быть сведены к следующим:

а) поиск (пеленгация) одиночного объекта или группы объектов, размеры которых пренебрежимо малы по сравнению с размерами поля;

б) поиск и определение формы, размеров, ориентации и взаимного расположения контурных и силуэтных объектов;

в) поиск и считывание различных линий (траекторий, графиков и др.);

г) поиск и отслеживание подвижных объектов;

д) определение рельефа поля, поиск, выделение и отслеживание его характерных областей, таких, как динамические зоны, участки, в которых значение параметра достигает максимальной либо минимальной величины, изопараметрические линии и др.

Успешность решения перечисленных задач поиска в значительной степени определяется тем, в какой мере поисковая система удовлетворит таким требованиям, как большая пространственная разрешающая способность, высокая энергетическая чувствительность, способность воспринимать спектр полезного излучения и подавлять спектр мешающего излучения и т. д. Эти требования являются в известной мере классическими и их выполнение зависит в основном от характеристик имеющихся элементов и узлов. Однако расширение класса задач, решаемых системами поиска, приводит к необходимости, кроме удовлетворения вышеупомянутым требованиям, учитывать также количество поисковых усилий, которые при этом расходуются.

Под поисковыми усилиями обычно подразумевается время, необходимое для получения определенного количества информации, ее стоимость, затрачиваемая энергия и т. д.

Очевидно, что процесс получения информации о поле излучений должен происходить таким образом, чтобы поисковые усилия расходовались наиболее рационально.

В общем случае критерии эффективности просмотра распределенного в пространстве источника излучения могут быть сведены к двум основным

а) максимизация вероятности обнаружения события при заранее заданном ограниченном количестве поисковых усилий;

б) минимизация затрат на поиск при условии достижения заданных вероятностных показателей обнаружения.

Для количественной оценки эффективности просмотра поля можно воспользоваться понятием относительного выигрыша. Очевидно, что результаты оптимизированного просмотра должны сравниваться с результатами равномерного безразличного просмотра при одних и тех же затратах, если эффективность просмотра оценивается по первому критерию. В этом случае относительный выигрыш будет определяться выражением

$$f = \frac{R_{опт} - R}{R},$$

где R — количество информации, получаемое при равномерном безразличном просмотре; $R_{опт}$ — количество информации, получаемое при использовании оптимальной стратегии просмотра.

Если равномерный безразличный просмотр и оптимальный просмотр обеспечивают с одинаковым успехом требуемое количество информации, а оценка эффективности производится по второму критерию, то для относительного выигрыша может быть записано выражение

$$f = \frac{\Phi - \Phi_{опт}}{\Phi},$$

где Φ — количество поисковых усилий при равномерном просмотре, $\Phi_{опт}$ — количество оптимально расходуемых усилий.

Совокупность требований, предъявляемых к системам, осуществляющим просмотр пространства, обусловливает разнообразие заложенных в них методов просмотра. Представленная на рис. 1 классификация методов просмотра пространства позволяет сгруппировать их по важным функциональным признакам.

Двумерность получаемой информации делает очевидным тот факт, что получение, преобразование и обработка информации также должны иметь двумерный характер. Практически любые устройства и системы, имеющие дело с двумерной информацией, обладают конечной пространственной разрешающей способностью. Поэтому устройства и системы, осуществляющие двумерное (параллельное) восприятие и обработку информации, могут быть представлены как состоящие из большого количества параллельных каналов, работающих одновременно.

Наиболее широкий класс таких систем составляют устройства, использующие твердотельные светочувствительные матрицы, волоконно-оптические преобразователи, позиционно-чувствительные полупроводниковые фотоприемники и др.

Просмотр поля по многим параллельным каналам позволяет свести практически к нулю время, необходимое для просмотра, значительно повышает информационные возможности систем и в ряде случаев является единственным методом, позволяющим успешно решать некоторые задачи.

В реальных условиях ценность информации, полученной с различных участков поля, может быть различной. Поэтому большое значение имеет приведение информационных возможностей отдельных каналов в соответствие с информационной ценностью просматриваемых участков. Такое приведение может обеспечиваться неодинаковой энергетической либо спектральной чувствительностью отдельных каналов, неравномерной пространственной разрешающей способностью и др. Причем информационные свойства каналов могут быть определены заблаговременно

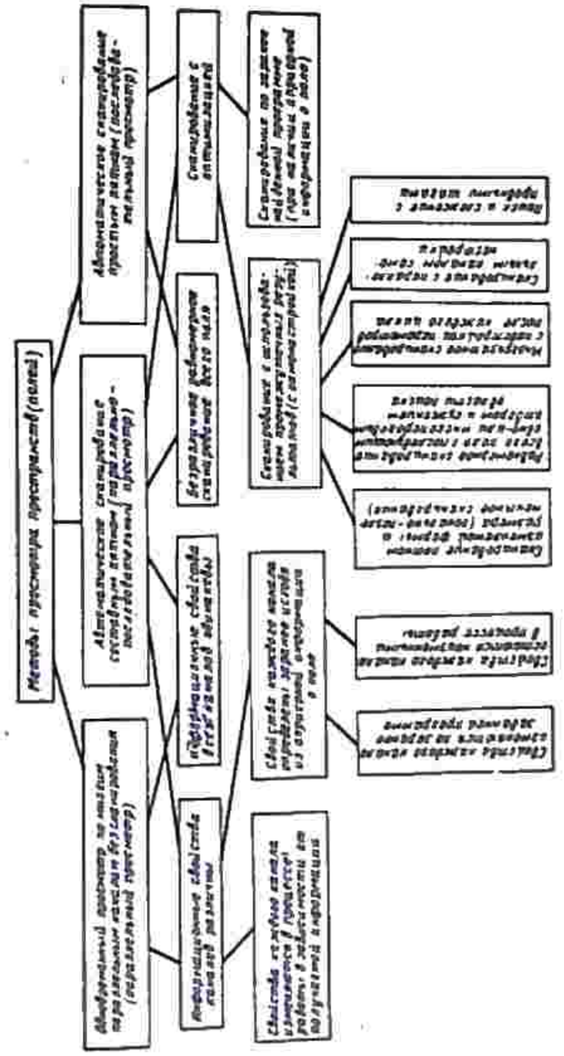


Рис. 1. Классификация методов просмотра полей излучений

и оставаться неизменными в процессе работы или изменяться по заранее заданной программе. Для обеспечения таких режимов работы системы необходимо располагать априорными данными о просматриваемом поле.

Информационные свойства каналов могут также изменяться в процессе работы в зависимости от получаемой информации, т. е. в системе со многими параллельными каналами может осуществляться самонастройка, позволяющая оптимизировать ее работу по некоторому критерию в случае, если предварительные сведения о поле отсутствуют. Однако наряду с ценностью и перспективностью многоканальных систем необходимо отметить, что наличие многих каналов приводит к значительным усложнениям таких систем и затрудняет их аппаратурное выполнение.

Обеспечение оптимальных режимов работы многоканальных систем, заключающееся в изменении информационных свойств отдельных каналов в процессе работы, при реализации связано со значительными трудностями, а часто и совсем невозможно.

Необходимо также упомянуть о двух таких важных и трудносовмещаемых в многоканальных системах требованиях, как высокая пространственная разрешающая способность и большие угловые размеры просматриваемого поля.

Указанные причины, а также ряд других обстоятельств объясняют тот факт, что наиболее широкое распространение в настоящее время получили сканирующие системы, осуществляющие последовательный просмотр поля излучений. В этих системах наряду с элементами, производящими параллельные действия над двумерными сигналами, обязательно присутствует элемент, превращающий двумерное сообщение в одномерное.

Некоторые задачи поиска удовлетворительно решаются при помощи безразличного равномерного сканирования. При этом результативность поиска определяется в основном техническими параметрами элементов и блоков сканирующей системы. Результативность поиска может быть значительно повышена путем оптимизации стратегии поиска. Выше отмечалось, что оптимизация в большой мере связана с наличием предварительных сведений о поле. Если последние имеются, а также известна функция выигрыша, то можно заранее найти такое распределение поисковых усилий по просматриваемому полю, при котором выполняется один из двух ранее упомянутых критериев.

Наиболее распространенным видом предварительных сведений о поле является распределение априорной вероятности события в различных точках поля. Функция выигрыша обычно характеризует успешность просмотра некоторого элементарного участка поля при размещении в нем определенного количества поисковых усилий.

Оптимизация просмотра при наличии априорных сведений о поле сводится либо к определению наиболее перспективного участка поля и к равномерному безразличному просмотру этого участка (остальные участки поля не просматриваются совсем), либо к определению программы неравномерного просмотра исследуемого поля [2]. В случае детерминированного просмотра двумерного поля программа сканирования обычно задается на отрезке $[0, T]$ в виде двух непрерывных или квантованных во времени функций $x_1(t)$ и $x_2(t)$, представляющих собой текущие координаты сканирующего пятна. Время T равно одному полному циклу сканирования всего поля. При случайном поиске находится наиболее рациональное распределение вероятности попадания сканирующего пятна в определенные точки поля. В любом из этих случаев в систему вводится программный блок 3 (рис. 2,*а*), управляющий параметрами сканирования.

Как и в многоканальных системах, оптимизация поиска в случае отсутствия предварительных сведений о поле излучений приводит к необходимости введения в сканирующую систему самонастройки. Характерным признаком самонастройки является то, что среди таких параметров сканирующей системы, как тип траектории, размер и форма сканирующего пятна, энергетическая и спектральная чувствительность, есть хотя бы один, поведение которого невозможно задать заранее и его изменение называется исключительно получаемой информацией о поле.

При этом в качестве исходной информации могут использоваться как сведения, поступающие по основному каналу сканирования 2 (рис. 2,*б*), так и сведения, поступающие по специальному дополнительному каналу 4 (рис. 2,*а*).

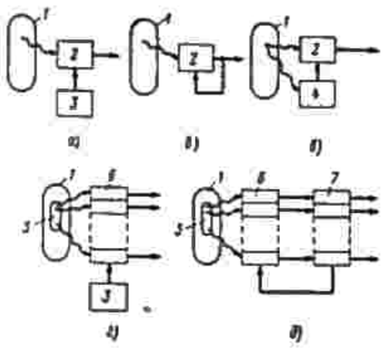


Рис. 2. Оптимальные структуры систем просмотра полей

1 — просматриваемое поле излучений, *2* — устройство сканирования, *3* — программный блок, *4* — дополнительный канал, *5* — составное сканирующее пятно, *6* — многоканальное устройство съема оптической информации, *7* — многоканальное устройство обработки информации.

Введение самонастройки требует выполнения ряда вычислительных и логических операций. Поэтому выигрыш, получаемый от использования самонастройки влечет за собой усложнение системы в аппаратурном отношении, так как канал самонастройки обычно содержит ряд блоков для запоминания и обработки получаемой информации, а также элементы, управляющие параметрами сканирования. Усложнение устройства в большой мере зависит от алгоритма самонастройки. Очевидно, что наиболее выгодно использовать алгоритмы, позволяющие сократить расход поисковых усилий при незначительном усложнении устройства.

Рассмотренные выше параллельный и последовательный методы получения информации о полях излучений обладают как определенными достоинствами, так и явными недостатками. Объединение обоих методов в один, заключающееся в том, что просмотр поля по многим параллельным каналам сопровождается одновременным сканированием этими каналами, позволяет значительно расширить возможности поисковых систем. В общем случае такой параллельно-последовательный просмотр можно представить как сканирование составным пятном 5, перекрывающим сразу целую область исследуемого поля. При этом способы оптимизации как параллельного, так и последовательного просмотра могут быть использованы для улучшения информационных характеристик параллельно-последовательного просмотра (рис. 2,*г*, *д*).

2. ОПТИМАЛЬНЫЕ СТРУКТУРЫ РАСПОЗНАЮЩИХ ОПТОЭЛЕКТРОННЫХ СИСТЕМ

Задача распознавания образов в общих чертах может быть поставлена в следующем виде.

Свойства какого-либо предмета можно представить как совокупность некоторых измерений или координат. Рассматривая пространство

X координат $x_1, x_2, \ldots, x_n$, можно отметить, что каждому предмету в этом пространстве будет однозначно соответствовать какая-либо точка. Совокупность отдельных предметов, которые каким-либо способом отнесены к одной группе, будем называть образом. Задача распознавания заключается в отнесении распознаваемого предмета к определенному образу

Структурно любая распознающая система содержит два блока, анализатор и классификатор (рис. 3).

Анализатор осуществляет измерение физических характеристик (признаков) объекта и рациональное представление признаков классификатору. Классификатор осуществляет разделение предметов (точек в многомерном пространстве X) по принятому правилу решения. Если

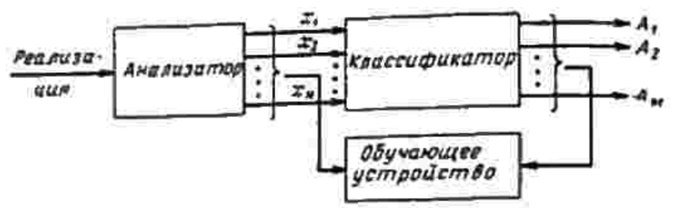


Рис. 3. Структурная схема распознающей системы.

рассматривать распознающую систему как информационно-измерительную, то необходимо различать следующие блоки:

1. Воспринимающее устройство — фоторецепторное поле с оптимально организованной системой развертки изображения.

2. Устройство сравнения, определяющее степень сходства описания предмета с эталонным описанием. Конструктивно устройство сравнения представляют собой функциональные электронолучевые преобразователи, оптические корреляторы и т. д.

3. Блок памяти для хранения программ, промежуточных данных и исходного описания. Блок памяти можно выполнить в виде оптических масок, на электронолучевых трубках, специальных фоточувствительных материалах.

4. Решающее устройство, осуществляющее в соответствии с принятым правилом решения отождествления предъявленного для распознавания изображения с определенным классом. Конструкция решающего устройства зависит от выбранного решающего правила.

Будем понимать под оптоэлектронным анализатором устройство первичной обработки распознаваемого изображения, выполняемое в большей части на твердом теле, имеющее оптическое звено и осуществляющее переключение, преобразование и коммутацию сигналов в оптическом диапазоне.

Важно отметить здесь отличие оптоэлектронных анализаторов от телевизионных, в которых наибольшая часть обработки информации о изображении заключается в преобразовании видеосигнала.

Основная задача оптоэлектронного анализатора может быть определена следующим образом.

При реализации устройства, распознающего большой класс изображений, осуществление полного и развернутого плана принятия решения при каждом новом предъявлении объекта приводит к увеличению памяти устройства и времени опознавания. Следовательно, необходимо в системе иметь блок, который бы позволил находить общие черты большого многообразия распознаваемых предметов. С другой стороны, естественные источники визуальной информации не могут учитывать свойств логических сетей системы, поэтому требуется устройство переработки или согласования информационных свойств большого многообразия внешних источников и логических сетей. Такая переработка (согласование) осуществляется оптоэлектронным анализатором путем образования порций информации и последовательностей из этих порций. Эти порции можно считать простейшими признаками, которые выделяет анализатор. В дальнейшем, на более высоких уровнях, из таких порций признаков синтезируются более сложные признаки и понятия, характеризующие образ.

Указанная совокупность простейших признаков может быть измерена анализатором либо одновременно, т. е. параллельным способом, либо в результате поочередного осмотра отдельных частей изображения. Во втором случае подразумевается анализатор последовательного принципа действия. Перспективы его использования вытекают из усложнения требований, предъявляемых к современным системам — распознаванию сложных изображений, отдельные части которых взаимосвязаны, обладают определенной «осмысленностью», организованностью, выражающейся в пространственных связях внутри изображений.

Примерами такого класса изображений могут служить: несколько рядом написанных букв и фигур (в этом случае возможно машинное чтение по слогам), изображение звездного поля, радио-, гидро-, теплолокационные изображения, в которых в отличие от телевизионных существенна информация о распределении яркости по полю кадра и малосущественна — о количестве градаций яркости.

Оптимальные структуры подобного класса систем распознавания требуют, очевидно, расширения функций и понятий оптического сканирования. При этом определяющим фактором в такой сканирующей системе становится гибкость сканирующего (развертывающего) элемента, который должен обеспечить приспособление развертки к различным изменениям оптических параметров изображения.

В общем случае оптимально организованный осмотр изображения позволяет из заданного набора признаков N выделить $P (P<N)$, обеспечивающих либо заданную надежность распознавания, либо получение максимального количества информации при минимальном числе шагов осмотра.

Как и в случае поиска, при распознавании составление оптимальной стратегии осмотра может основываться: а) на использовании априорных данных о предъявляемых изображениях, б) на использовании промежуточных результатов осмотра изображений.

В первом случае все методы представляют собой задачу так называемого «минимального маршрута» обследования признаков, приводящего к программной развертке изображения. Средняя длительность маршрута определяется так

$$L = \sum_{i=1}^{T} k_i P(B_i),$$

где T — количество распознаваемых объектов, k_i — количество признаков, которые необходимо измерить для принятия решения.

Программная развертка изображения вычисляется, как правило, на быстродействующих ЭЦВМ при подготовке системы к распознаванию данного алфавита классов по введенному критерию минимизации. Если вводится информационный критерий, то часто пользуются вычислением неопределенности решения при отнесении признака X_k к классу A [3]. При этом признаки упорядочивают таким образом, что первым будет признак с минимальным значением $H(A/X_k)$, вторым — признак, обеспечивающий минимальную неопределенность решения совместно с первым признаком, третьим — признак, обеспечивающий минимальную неопределенность решения совместно с первыми двумя и т. д. В результате — составляется оптимальная программа осмотра изображения.

Следует отметить, что правила минимизации здесь могут быть различными в зависимости от принципов, в соответствии с которыми функционирует распознающая система, однако приведенный пример является достаточно общим для многих типов систем. При этом структура анализатора соответствует рис 2,*а*. Технически это означает сопряжение сканирующего устройства с ЭЦВМ, которая осуществляет программное управление лучом развертывающей системы

Существенный интерес представляет также постановка задачи о распределении поисковых усилий для целей распознавания, рассматриваемая выше в связи с вопросами повышения эффективности оптического поиска. Оптоэлектронный анализатор изображения, построенный с учетом минимизации расходуемых усилий, приобретает большую гибкость и организованность, позволяет производить распознавание изображения в условиях ограниченного времени показа, что чрезвычайно важно для ряда специальных применений

Задача распределения усилий при распознавании имеет в сравнении с поиском ряд особенностей, связанных со спецификой вопроса Основная из них заключается в необходимости предусмотреть расход усилий как на этапе составления эталонов (памяти), т. е. до распознавании, так и в процессе самого распознавания. На этапе, предшествующем распознаванию, при составлении оптимальной стратегии осмотра, эта задача сходна с задачей распределения поисковых усилий в том смысле, что здесь оказывается пригодной методика, изложенная в [4], однако с учетом того, что отыскивается не один признак, что само по себе характерно для этого класса задач, а некоторая совокупность, каждый элемент которой обладает различной информационной емкостью (весом). Обобщение этой методики на случай распознавания приводит к взвешенному суммированию стратегий для нахождения отдельных признаков для одного класса с весовыми коэффициентами, равными априорной вероятности появления этого признака в данном классе При этом следует учитывать априорную вероятность появления самого класса изображений

Итак, при распределении усилий распознавания на этапе формирования эталонов (памяти) решаются следующие задачи

1. Нахождение «интересных» областей, внутри которых необходимо производить поиск признаков.

2 Определения количества требуемых усилий распознавания для поиска внутри областей.

3. Составление оптимальной процедуры осмотра всего изображения, исходя из имеющегося ресурса усилий.

4 Запоминание маршрута обследования.

5 Информационная оценка выбранных маршрутов для каждого класса.

На втором этапе, в процессе собственно распознавания выбирается наилучшая стратегия в смысле минимума затрачиваемых усилий. В том случае, когда априорные знания о вероятности появления изображения известны и распределения его признаков определены, возможно из имеющегося набора стратегий, полученных на предыдущем этапе, выбрать лучшую в соответствии с теми исходами, которые будут иметь место при их применении, и составить таблицу соответствий «совокупность признаков изображения ⟶ стратегия осмотра с минимальным количеством поисковых усилий, или сокращенно $x_i \longrightarrow \varphi_i$». Эта таблица находится в вычислительной машине и выбор из нее осуществляется при предъявлении конкретного X_i

Таким образом, структура такой системы, так же как и в предыдущем случае, включает сочленение вычислительной машины со сканирующим устройством, поскольку на основе принятого критерия

оптимальности функции $x_i \longrightarrow \varphi_i$ осуществляется управление луча в развертывающем устройстве

Здесь возможен и более сложный случай, когда после осмотра распознаваемого изображения принимается решение о его принадлежности с определенным риском В этом случае для выбора оптимальной процедуры осмотра из всех имеющихся необходимо ввести критерий «минимум среднего риска», который определяется как разность усилий распознавания при выборе некоторой конкретной процедуры Π_i и минимальных усилий, которые тратятся при выборе некоторой другой процедуры при распознавании одного и того же изображения. Тогда риск представляет собой некоторое количество дополнительных усилий по сравнению с неизбежными усилиями. Для всех изображений B средний риск определяется: $R_{ср} = \Sigma P(B_i) R_{ii}$, где R_{ii} — некоторое среднее значение риска при применении процедуры Π_i к изображению B_i

Выражение может быть вычислено для каждой процедуры, и в соответствии с этим критерием можно выбирать такую процедуру осмотра, которая обеспечивает минимальную величину среднего риска $R_{ср}$

Еще большую гибкость проявляет анализатор, предназначенный для локального анализа изображения, в ходе которого обеспечивается максимум или минимум некоторой предварительно введенной локальной оценки, связанной с оптической структурой изображения. Естественно, что для таких структур характерно более полное использование оптического канала анализатора. Закон осмотра определяется в этом случае исходя из вычисления введенной локальной оценки Поэтому в зависимости от ее вида сканирующее устройство воспроизводит различные траектории осмотра распознаваемого изображения Оценочные функции могут определяться либо значением почерненностей в окрестностях рассматриваемой области, либо степенью почерненности отдельного участка изображения, наблюдаемого через окошко переменной прозрачности, либо запомненными в предыдущие моменты времени оптическими параметрами, по совокупности которых делается следующий шаг осмотра, и т. д. При этом учитывается как содержание локальных признаков, так и последовательность их возникновения

Когда закон развертки заранее определить трудно, в системе должно быть предусмотрено обучение по известным методикам и развертку сканирующего устройства называют обучаемой, приспособленной в процессе осмотра вырабатывать оптимальную стратегию на основе накопленного до этого момента опыта

Структура рассматриваемого класса сканирующих систем распознавания показана на рис. 2,*б*. т. е. она обязательно предусматривает обратную связь по информации

Таким образом в области поиска и распознавания при наличии априорных сведений возможный подход к оптимизации просмотра поля излучений связан с составлением некоторой оптимальной программы. Отсутствие каких-либо данных о просматриваемом поле приводит к необходимости введения самонастройки либо обучения, заключающихся в использовании промежуточных результатов просмотра для коррекции процедуры поиска или распознавания

ЛИТЕРАТУРА

1. Катыс Г. П. Автоматический обзор и поиск в оптическом диапазоне. Изд-во «Наука», 1966
2 Здор С. Е. О синтезе поисковых сканирующих систем В сб «Современные проблемы кибернетики» Изд-во «Наука», 1970.
3. Барабаш Ю. Л. и др. Вопросы статистической теории распознавания Изд-во «Советское радио», 1967
4. Koopman B. O. The theory of search. Operations Research, 1957, v. 5, № 5.

УДК 681.3.06

ДИАГНОСТИЧЕСКИЙ АНАЛИЗ ОДНОТАКТНЫХ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ СХЕМ

Г. А. МИРОНОВ, М. А. ОСТРОВИДОВ

1. ТИП РАССМАТРИВАЕМЫХ СХЕМ

В схемах ЦВМ наибольшее применение находят логические элементы, реализующие функции конъюнкции, дизъюнкции, отрицания. Из этих элементов составляются комбинационные схемы, реализующие заданные сложные функции. Число элементов, входящих в схемы, бывает достаточно велико и связи элементов настолько сложны, что поиск неисправного элемента представляется весьма кропотливым делом.

В настоящей работе под диагностическим анализом понимается анализ комбинационной схемы, имеющей неисправный логический элемент, с целью указания этого неисправного элемента или указания некоторого подмножества из множества элементов, составляющих схему, которая включает в себя неисправный элемент. Во всяком случае правильнее говорить о подмножестве подозреваемых элементов.

Поскольку предполагается, что поиск неисправного элемента должен вестись только «логическим» методом, т. е. без физического вмешательства человека в работу диагностируемой схемы путем изменения ее или подключения дополнительных приборов, то, очевидно, речь может идти только о правильных *) схемах [1].

Для проведения диагностического анализа необходимо иметь исходную информацию. Такой информацией являются фиксируемые схемным или программным контролем результаты проявления неисправностей на одном или нескольких выходах схемы. Кроме того, необходимо иметь входную информацию, т. е. тот набор входных сигналов, при котором проявилась неисправность.

При дальнейшем изложении блоком будем называть часть схемы ЦВМ, для которой можно указать исходную информацию. Например, в настоящей работе блоком считается совокупность логических элементов, реализующих элементарную машинную операцию (ЭМО) [2], для которой схемным или программным контролем устанавливается факт наличия внутри нее неисправного элемента.

Для упрощения как дальнейшего изложения, так и алгоритма диагностического анализа будем полагать, что все элементы конъюнкции и дизъюнкции имеют два входа и один выход. Такое допущение не нарушит общности дальнейших рассуждений [1]. Если в реальной схеме имеются элементы с числом входов более двух, то, применяя правила эквивалентных преобразований, можно привести рассматриваемую схему к схеме, реализующей ту же функцию, но имеющую только элементы с двумя входами.

ЭМО могут выполняться за один или несколько тактов работы ЦВМ. Мы ограничимся рассмотрением только таких ЭМО, которые выполняются за один такт.

Итак, ниже рассматриваются правильные комбинационные схемы, реализованные на элементах конъюнкции, дизъюнкции и отрицания, реализующие однотактные ЭМО. При этом считается, что каждый элемент конъюнкции и дизъюнкции имеет не более двух входов и один выход, а элемент отрицания — один вход и один выход.

*) Логическая схема устройства называется правильной, если соответствующий ей ориентированный граф не содержит замкнутых контуров.

2. ДИАГНОСТИРУЕМЫЕ НЕИСПРАВНОСТИ

Предлагаемый диагностический анализ основан на проверке правильности выполнения каждым логическим элементом присущей ему функции при заданных входных сигналах. Поэтому в дальнейшем нас не будет интересовать физическая реализация того или иного логического элемента (ламповый, полупроводниковый и т. д.).

С целью повышения эффективности ЦВМ оснащены некоторым количеством запасных сменяемых элементов. В случае выхода из строя какого-либо сменяемого элемента он заменяется на аналогичный, заведомо исправный из имеющегося запаса. Взятый из машины неисправный элемент с помощью вспомогательного оборудования и приборов подвергается тщательному анализу с целью определения возникшей физической неисправности [3] и ее устранения.

Такая практика эксплуатации ЦВМ представляется вполне разумной, так как поиск и устранение физической неисправности требует применения дополнительной аппаратуры и, главное, значительно большего времени, чем замена сменяемого элемента. Поэтому диагностика должна быть направлена на указание элемента, имеющего логическую неисправность.

По аналогии с [4] логической неисправностью элемента будем называть такую неисправность, которая вызывает изменение логической функции, присущей данному элементу в исправном состоянии.

Элементы отрицания, имеющие один вход, могут выполнять четыре различные функции одного переменного: $f_0=x$, $f_1=\bar{x}$, $f_2=0$, $f_3=1$. Из этих функций только одна f_1 является присущей исправному элементу, а остальные три функции являются следствиями логических неисправностей. Для элементов с двумя входами существуют 16 функций алгебры логики, из которых только одна является присущей исправному элементу, а 15 являются следствиями логических неисправностей.

По характеру проявления неисправностей во времени обычно различают сбой, перемежающийся отказ и постоянный отказ. В данной работе рассматривается возможная при любом из этих трех случаев ситуация: при выполнении ЭМО в блоке появилась логическая неисправность, ее появление зафиксировано схемным или программным контролем.

В общем случае в блоке может существовать любое количество неисправных элементов. В настоящей работе ограничим возможное количество неисправных элементов в блоке числом 1, что характерно для нормального режима эксплуатации ЦВМ. При хорошо организованной эксплуатации возникшая в вычислительной машине ошибка выявляется и устраняется за интервал времени, значительно меньший, чем среднее время безотказной работы машины, т. е. с большой вероятностью устраняется до появления новой ошибки.

3. ИДЕЯ МЕТОДА

Идея выполнения диагностической процедуры заключается в следующем. Пусть при выполнении ЭМО имеющимся контролем зафиксировано наличие в блоке логической ошибки. Носителем этой ошибки может быть любой элемент блока. Однако при той входной информации, при которой зафиксирована неисправность, в работе могут участвовать и влиять на результат не все, а лишь некоторые элементы блока. Очевидно, что носителем зафиксированной неисправности могут быть только эти элементы. Поэтому целью диагностической процедуры является выборка из множества элементов, составляющих блок, некоторого подмножества. Список элементов этого подмножества называется таблицей предположений и может содержать любое количество элементов от 1 до N, составляющих блок.

Таблица предположений представляет собой форму представления результата диагноза при одном наборе входных сигналов. Составляя таблицы предположений для разных наборов значений входных сигналов и определяя пересечения этих таблиц, можно сузить область предположений.

Таблицу предположений удобно составлять, просматривая схему от некоторого выхода ко входам. Первым подозреваемым элементом, записываемым в таблицу, естественно, следует считать выходной элемент блока, т. е. элемент высшего яруса схемы. Далее последовательно просматриваются все остальные элементы блока и те из них, функции которых не влияют на результат данного выхода схемы, при входных данных, породивших ошибку, исключаются из рассмотрения вместе с элементами, подключенными только к их входам.

Для определения того, влияет ли функция элемента на функцию выхода блока при данной входной информации, необходимо предварительно просчитать значения функций всех элементов блока для этого набора входной информации, считая исправными все элементы блока. Полученные значения назовем действующими. Вся диагностическая процедура выполняется на основе анализа действующих значений функций.

Рассмотрим подробнее возможность влияния различных типов логических элементов на получение неверного результата.

1. **Инвертор.** Если значение функции инвертора влияет на значение выходной функции блока, то, очевидно, и значение функции элемента нижнего яруса, подключенного к входу инвертора, также влияет на выходную функцию блока. Следовательно, следующим элементом, записанным в таблицу предположений после рассматриваемого инвертора, должен быть элемент, выход которого подключен ко входу инвертора.

2. **Конъюнктор.** Если его функция влияет на функцию выхода блока, то необходимо проанализировать действующее значение функции конъюнктора. Если действующее значение функции конъюнктора равно 0 и действующие значения функций элементов, подключенных ко входам конъюнктора, тоже равны 0, то последние элементы в таблицу предположений заносить не надо. Это следует из того, что при исправности конъюнктора изменение значения функции (1 вместо 0) любого из элементов, подключенных к его входам, не вызовет изменения значения функций конъюнктора. Поэтому функции этих элементов не влияют в данном случае на выходную функцию блока.

Если действующее значение функции конъюнктора равно 0, а действующее значение функции одного из элементов, подключенных к его входам, равно 1, то на значение функции блока влияет функция того элемента, у которого действующее значение функции равно 0. Следовательно, в таблицу предположений надо занести этот элемент, поскольку изменение значения функции этого элемента вызовет изменение значения функции конъюнктора (при условии исправности последнего).

Если действующее значение функции конъюнктора равно 1, то изменение значения функции любого из подключенных к его входам элементов (0 вместо 1) вызовет изменение значения функции конъюнктора. Поэтому оба элемента, подключенных к входам конъюнктора, подлежат занесению в таблицу предположений.

3. **Дизъюнктор.** Если действующее значение функции дизъюнктора равно 1 и действующие значения функций элементов нижних ярусов, подключенных ко входам дизъюнктора, также равны 1, то эти последние элементы в таблицу предположений не вносятся. В самом деле, в случае исправности дизъюнктора изменение значения функции одного из этих элементов (0 вместо 1) не изменит значения функции дизъюнктора.

Если действующее значение функции дизъюнктора равно 1, а действующее значение функции одного из элементов, подключенных к его входам, равно 0, то на значение функции дизъюнктора влияет только значение функции элемента, у которого действующее значение функции дизъюнктора равно 1. Следовательно, этот элемент должен быть занесен в таблицу предположений.

Если действующее значение функции дизъюнктора равно 0, то изменение значения функции любого из элементов, подключенных к его входам, вызовет изменение значения функции дизъюнктора (если последний исправен). Поэтому оба элемента, подключенные ко входам дизъюнктора, должны быть включены в таблицу предположений.

Описанным образом просматриваются все элементы блока по схеме «сверху вниз», т. е. от выбранного выхода блока к его входам. В результате такого просмотра в таблице предположений оказываются все те элементы, неисправность которых может вызвать неверный результат при той входной информации, при которой схемным или программным контролем зафиксирована ошибка.

В общем случае диагностируемый блок может иметь несколько выходов. Если имеющийся контроль фиксирует не только факт наличия ошибки в блоке, но и неверный результат на входах блока, то диагностическую процедуру следует выполнять только для тех выходов блока, на которых зафиксирован неверный результат.

4. АЛГОРИТМ СОСТАВЛЕНИЯ ТАБЛИЦЫ ПРЕДПОЛОЖЕНИЙ

Приведенный ниже алгоритм предполагает, что: а) все элементы (сквозным образом) и входы блока (для каждого в отдельности) предварительно пронумерованы, б) имеется описание исследуемой схемы, в котором приведены номера элементов, указан порядок соединения их друг с другом и указаны их типы. Все выходные элементы блока должны быть размещены в таблице T_a.

Особенностью алгоритма является поочередный анализ ветвей схемы. В случае, если оба элемента, подключенные ко входам рассматриваемого элемента, должны быть включены в таблицу предположений T_p, то один из них (подключенный к правому входу) вначале записывается во вспомогательную таблицу T_m на очередное свободное место, а другой становится рассматриваемым. Такая процедура повторяется до тех пор, пока просмотренным окажется входной элемент блока. После этого производится переход к рассмотрению элемента, записанного в последней строке вспомогательной таблицы. После выборки этого элемента его строка в таблице T_m стирается.

Алгоритмический процесс прекращается, когда просмотрены все ветви схемы (в таблице T_a не осталось непросмотренных элементов) и таблица T_m пуста. Алгоритм составления таблицы предположений может быть представлен в следующем виде.

1. Задать начальные условия $m_* = 0$, $p = 0$, $a = l$ (m — номер строки вспомогательной таблицы, p — номер строки таблицы предположений, l — число выходных элементов схемы). Перейти к п. 2.

2. Выбрать из таблицы входных элементов T_a номер элемента, записанный в строке a, и считать элемент с этим номером рассматриваемым, уменьшить текущий номер строки таблицы T_a (т. е. значение величины a) на единицу. Перейти к п. 3.

3. Увеличить номер строки таблицы предположений T_p на единицу и записать в эту строку номер рассматриваемого элемента. Перейти к п. 4.

4. Определить тип рассматриваемого элемента. Если рассматриваемый элемент — конъюнктор, то перейти к п. 5; если — дизъюнктор, то перейти к п. 11; если — инвертор, то перейти к п. 8.

5 Проверить, равно ли 0 действующее значение функции рассматриваемого элемента. Если да — перейти к п. 6, в противном случае — к п. 10

6 Найти элемент B_1, подключенный к левому входу рассматриваемого элемента и проверить, равно ли 0 действующее значение функции элемента B_1. Если да — перейти к п. 7, в противном случае — к п. 9.

7. Найти элемент B_2, подключенный к правому входу рассматриваемого элемента и проверить, равно ли 1 действующее значение функции элемента B_2. Если да — перейти к п. 8, в противном случае — к п. 16.

8. Найти элемент B_1, подключенный к левому входу рассматриваемого элемента. Перейти к п. 14.

9. Найти элемент B_2, подключенный к правому входу рассматриваемого элемента. Перейти к п. 14

10. Найти элемент B_2, подключенный к правому входу рассматриваемого элемента; увеличить номер m строки вспомогательной таблицы T_m на единицу и записать, в эту строку номер элемента B_2. Перейти к п. 8

11. Проверить условие: «действующее значение функции рассматриваемого элемента равно 1». При выполнении условия перейти к п. 12, в противном случае — к п. 10.

12. Найти элемент B_1, подключенный к левому входу рассматриваемого элемента и проверить, равно ли 1 действующее значение функции элемента B_1. Если да — перейти к п. 13, в противном случае — к п. 9

13. Найти элемент B_2, подключенный к правому входу рассматриваемого элемента и проверить, равно ли 1 действующее значение функции элемента B_2. Если да — перейти к п. 16, в противном случае — к п. 8

14. Проверить условие: «элемент B не является входом блока». При выполнении условия перейти к п. 15, в противном случае — к п. 16.

15. Считать элемент B рассматриваемым. Перейти к п. 3.

16. Проверить условие: «вспомогательная таблица T_m пуста». При выполнении условия перейти к п. 17, в противном случае — к п. 19

17. Проверить условие: «просмотрены все выходные элементы блока». При выполнении условия перейти к п. 18, в противном случае — к п. 2.

18. Процесс прекратить.

19. Выбрать для рассмотрения элемент, номер которого записан в m-й строке вспомогательной таблицы T_m; стереть эту строку, уменьшить номер строки m на единицу; проверить условие: «выбранный элемент не является входом блока». При выполнении условия перейти к п. 3, в противном случае — к п. 16.

ЛИТЕРАТУРА

1. Миронов Г. А., Федотова Д. Э. Составление тестов для логических сетей. В сб. «Цифровая вычислительная техника и программирование», вып. 3. Изд-во «Советское радио», 1967.
2. Криницкий Н. А., Миронов Г. А., Фролов Г. Д. Описание систем команд ЭЦМ с помощью элементарных машинных операций. В сб. «Цифровая вычислительная техника и программирование», вып. 4. Изд-во «Советское радио», 1968.
3. Миронов Г. А. Испытательные программы для контроля электронных цифровых машин. Изд-во «Наука», 1964.
4. Федотова Д. Э., Миронов Г. А. Применение контролирующих тестов для диагностики неисправностей логических сетей. В сб. «Цифровая вычислительная техника и программирование», вып. 4. Изд-во «Советское радио», 1968.

ИМЕННОЙ УКАЗАТЕЛЬ

СОДЕРЖАНИЕ

ЦИФРОВАЯ ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРОГРАММИРОВАНИЕ

СБОРНИК СТАТЕЙ ПОД РЕДАКЦИЕЙ *А. И. КИТОВА*

Редактор Т. М. Любимова
Художественный редактор З. Е. Бендрова
Технический редактор Г. З. Кузнецова
Корректоры Е. П. Озерецкая Л. А. Максимова

Сдано в набор 2/XI 1971 г. Подписано в печать 27/II 1972 г. Т-04092
Формат 70×108¹/₁₆ Бумага типографская № 2 Объем 16,8 усл. п. л. Уч.-изд. л. 16,79
Тираж 12 500 экз. Зак. 478
Издательство „Советское радио". Москва, Главпочтамт, п/я 693. Цена 1 руб.

Московская типография № 10 Главполиграфпрома
Комитета по печати при Совете Министров СССР,
Шлюзовая наб., 10.

УДК 681.3—+003

О некоторых формальных языках. Криницкий Н. А. «Цифровая вычислительная техника и программирование». 1972, вып. 7, стр. 5—12.

Введено определение языков как множеств слов, порождаемых индуктивными грамматиками. Установлено соотношение между дедуктивными и индуктивными порождающими грамматиками, а также между грамматиками, заданными в форме Бекуса, и двумя ранее указанными видами грамматик. Делается попытка формализации понятия семантики.

Библ. 3 назв.

УДК 681.322

Американские автоматизированные информационные системы для медицины. Китов А. И. «Цифровая вычислительная техника и программирование», вып. 7, 1972, стр. 12—23

Статья представляет собой описание принципов работы систем MEDLARS и MSIS. В статье приведены некоторые данные, которые будут полезны для разработчиков информационных систем вообще и медицинских в частности.

УДК 681.3-52

Общие принципы и структура математического обеспечения автоматизированных систем управления. Тараканов К. В. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 23—29

В статье излагаются некоторые принципы, которых следует придерживаться при разработке специального математического обеспечения автоматизированных систем управления. Их выполнение позволит создать условия для более успешной разработки комплекса программ при меньшей трудоемкости и в более короткие сроки.

Обращается внимание на разработку автоматизированных информационных систем и в особенности на необходимость создания базовых информационных систем как основы всего специального математического обеспечения.

Рис. 1.

УДК 681.3.51

К вопросу оценки вероятности ошибки в выдаваемых вычислительным центром данных. Семаков В. В. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 30—37

Излагается подход к проблеме определения вероятности появления случайной ошибки в выдаваемых вычислительным центром данных при решении экономических задач на ЭВМ. Приводятся формулы для оценки этой вероятности. Статья представляет интерес для широкого круга специалистов, занимающихся организацией вычислительных центров.

Рис. 2, библ. 4 назв.

УДК 681.3.06

О расчете времени исполнения машинных программ. Герман В. А. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 37—50

В работе излагаются методы расчета времени исполнения машинных программ на ЦВМ. Исследуется частота исполнения каждого блока программы. Анализируемые программы представляются стохастическими граф-моделями.

Предлагаемые алгоритмы могут применяться для расчета моделей большой размерности, так как они широко используют структурные свойства исследуемых граф-моделей. Для установления различных структурных свойств граф моделей используется аппарат булевых матриц.

Рис. 9, библ. 9 назв.

УДК 681.3.04—+003

Проблемно-ориентированный язык и система генерирования программ для задач обработки данных. Павельев В. А., Степаниченко Д. А. «Цифровая вычислительная техника и программирование», вып. 7, 1972, стр. 50—57

Описана система программирования RPG, разработанная американской фирмой IBM. Кратко описана сущность программирования с использованием системы RPG. Статья представляет интерес как для программистов, эксплуатирующих ЭВМ, так и для специалистов, разрабатывающих математическое обеспечение ЭВМ.

Библ. 3 назв.

УДК 681.306

Об использовании грамматических средств в ИПС для больших массивов документов. Китов А. И., Грачева Е. К. Сб. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 57—68

Описывается информационный язык с грамматикой, используемой как для повышения точности описания и поиска документов, так и для выдачи на печать формализованных фраз естественного языка, а также практически реализованные алгоритмы построения лексического словаря и формирования формализованных фраз.

Табл. 1, рис. 6, библ. 3 назв.

УДК 681.306

Организация памяти для выполнения обращений по признакам. Кожурин Ф. Д., Антоненко М. Г., Швец Н. Я. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 68—75.

Статья посвящена вопросам организации памяти с последовательным доступом для выполнения поиска объекта по признакам.

Предлагается метод «отрезков», который позволяет осуществить поиск по максимально возможной группе признаков. Показано, что предложенный метод является достаточно удобным, дается оценка этого метода с помощью предложенного автором коэффициента избыточности. Метод отрезков целесообразно применять в тех случаях, когда объем хранимых данных об объектах значительно превышает объем признаковой информации об этих объектах.

Рис. 2, библ. 5 назв.

УДК 681.3.065

О некоторых принципах организации фактографических информационно-поисковых систем с табличной формой представления информации и библиотечным математическим обеспечением. Горизонтова М. Б. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 75—83.

Описаны принципы построения фактографической информационно-поисковой системы с представлением информации в виде объектно-характеристических таблиц. Дается краткое описание состава хранилища информации, перечисляются процедурные средства системы, обеспечивающие поиск и обработку информации, рассматривается актуальная для динамических систем проблема отсутствия информации и предлагаются конкретные пути ее решения.

Библ. 6 назв.

УДК 681.3.51

Об одном методе объединения массивов. Матэр Е. А., Фельдман Г. Л. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 83—93

Предлагается алгоритм объединения массивов разной длины, записанных на магнитной ленте, в один упорядоченный массив. Алгоритм основан на идее Хаффмана построения кодов с минимальной избыточностью и соответствующего дерева объединения. Приводятся оценки эффективности модификаций алгоритма по числу проходов лент и сравнение с методом слияния.

Табл. 2, рис. 3, библ. 7 назв.

УДК 681.3.62-523.8

Принцип организации информационной системы (ИС) отраслевой автоматизированной системы управления (ОАСУ). Аскеров Т. М., Фейзуллаев А. П. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 93—104.

В статье описывается принцип организации информационной системы (ИС) отраслевой автоматизированной системы управления (ОАСУ). В основу положен метод логических шкал. Система является универсальной как с точки зрения областей использования, так и с точки зрения типов ЭВМ, на которых она может быть реализована. Система описывается на примере задачи сбора информации для машиностроительного министерства.

Табл. 1, рис. 5, библ. 3 назв.

УДК 681.3.053

Метод простых чисел для кодирования поисковых образов объектов в информационно-поисковых системах. Вуль Ю. И., Инякин В. Ф. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 104—107

В статье описан оригинальный метод кодирования дескрипторов при построении информационно-поисковой системы. Авторы рекомендуют в качестве кодов дескрипторов использовать простые числа. Описанный метод может представлять интерес в некоторых специальных случаях.

Библ. 1 назв.

УДК 681.306

Об одном способе организации поискового массива в библиографических ИПС. Арнаудов Д. Д. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 107—111

Рассматривается вопрос построения поискового массива и дескрипторного словаря, дающих возможность создать эффективную информационно-поисковую систему. Предполагается большой объем массива документов и словаря дескрипторов и ограниченные объемы внешней и оперативной памяти.

Табл. 1, рис. 1.

УДК 681.3.065.4

Организация хранения и поиска информации о химических соединениях. Керимов С. К. и Мехтиев А. А. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 111—117

В работе рассматривается способ алгоритмического кодирования, химических соединений, разработанный в НИИТЭХИМе (г. Москва), а также алгоритм поиска информации об этих соединениях. Хранение кодов химических соединений в памяти ЭВМ на магнитных лентах осуществляется прямым способом.

Описываются критерии смыслового соответствия и алгоритм поиска химических соединений.

Принципиальной особенностью данной ИПС фактографического типа является сочетание дескрипторного способа описания документов с символьным специализированным языком линейных кодов.

Табл. 1, библ. 4 назв.

УДК 681.3.06

Об одной программе анализа графов. Гаазе-Рапопорт М. Г., Лукьянова С. П. «Цифровая вычислительная техника и программирование», 1962, вып. 7, стр. 117—120

В статье описана программа для ЦВМ, решающая задачу представления графа в прусно-параллельной форме с выделением «остатка» графа, содержащего циклы.

Рис. 1, библ. 1 назв.

УДК 681.3.519.2

О составе операций и статистике их использования в программах управляющих ЦВМ. Липаев В. В., Колин К. К. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 120—126.

Приводятся результаты статистического анализа состава операций в программах четырех ЦВМ, используемых в сложных системах автоматизированного управления объектами, которые свидетельствуют о высокой разветвленности программ и о преобладании в них логических операций над арифметическими. Даются рекомендации по использованию в ЦВМ данного класса специальных команд для выполнения операций с частью слова и операций с константами. Делается вывод о целесообразности применения в управляющих ЦВМ систем команд, специально ориентированных на решение информационно-логических задач.

Табл. 2, библ. 3 назв.

УДК 681.3.06-52

Система автоматизации программирования и выпуска технической документации на программу для управляющих ЦВМ (ЯУЗА-1). Серебровский Л. А., Сибиряков П. Г., Липец И. Е., Панова Л. А. Сб. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 126—133.

Излагается структура и основные технические и эксплуатационные характеристики системы автоматизации программирования, входным языком которой служит покомандный универсальный автокод ЯУЗА. Система обеспечивает изготовление программ для управляющих ЦВМ широкого класса. Система настраивается на конкретную ЦВМ путем задания ее параметров и системы команд.

Система реализована на М-220. Она позволяет автоматически стыковать программы в единую программу большого объема (порядка сотен тысяч команд). Изготовление программы сопровождается автоматическим выпуском всей технической документации, которая оформляется с учетом требований ЕСКД.

Рис. 4, библ. 2 назв.

УДК 681.3-51

К вопросу оценки функционирования звена «оператор—пульт управления» с помощью моделирования на ЦВМ. Исаев В. П., Попов М. Ф., Попов Р. А. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 133—139.

Рассматриваются вопросы функционирования звена «оператор—пульт управления». Для оценки качества функционирования звена предлагается использовать модель, реализуемую на ЦВМ. Приводится блок-схема и результаты моделирования, позволяющие производить оценку качества функционирования звена «оператор—ПУ» путем интегральной оценки времени, затрачиваемого на набор, контроль и выдачу команды.

Рис. 5, библ. 13 назв.

УДК 681.3.06.519.162

Некоторые математические модели систем с разделением времени. Яшков С. Ф. «Цифровая техника и программирование», 1972, вып. 7, стр. 140—149.

Рассматриваются принципы построения алгоритмов планирования в системах с разделением времени (СРВ) и исследуются модели СРВ методами теории массового обслуживания. Получены аналитические выражения для определения среднего времени ожидания и времени отклика задач различных классов в СРВ с произвольным количеством очередей при учете потерь времени на квантование обслуживания. Подтверждена точность предложенных оценок с помощью моделирования СРВ на языке СЛЭНГ.

Табл. 1, рис. 6, библ. 9 назв.

УДК 681.3.51

Об одной задаче определения режима профилактики. Горохов Ю. П., Сокотов Г. А. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 149—160.

Рассматриваются две задачи, связанные с определением режима профилактики, наименее уклоняющегося в смысле принятого критерия от «равномерного».

Табл. 3, библ. 8 назв.

УДК 681.3—003

Некоторые свойства звуковых кодов. Фролов Г. Д. Сб. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 160—166.

В статье приводится описание приближенного способа автоматического сегментирования речевого сигнала и некоторые свойства звуковых кодов, которые могут быть использованы при создании систем автоматического опознания речевых образов или синтеза речи.

Библ. 2 назв.

УДК 681.322

Применение ЭВМ для исследования некоторых характеристик клиппированных речевых сигналов. Русецкий Г. Н., Скокан Ю. П. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 166—172.

Приводится методика и результаты измерений с помощью ЭВМ некоторых характеристик клиппированного речевого сигнала.

Табл. 1, рис. 7, библ. 5 назв.

УДК 681.3.51

Оптимальные структуры оптоэлектронных систем поиска и распознавания. Катыс Г. П., Здор С. Е., Широков В. Б. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 172—181.

Рассматриваются структуры оптоэлектронных систем поиска и распознавания и исследуются возможности их оптимизации по ряду критериев на основе использования априорной и текущей информации с учетом наложенных ограничений. Показывается, что оптимизация приводит к введению в системы специальных программных блоков либо каналов самонастройки. Обсуждаются вопросы разработки программы сканирования и алгоритмов самонастройки в системах поиска и распознавания.

Рис. 3, библ. 4 назв.

УДК 681.3.08

Диагностический анализ одновходных комбинационных логических схем. Миронов Г. А., Островидов М. А. Сб. «Цифровая вычислительная техника и программирование», 1972, вып. 7, стр. 182—186.

Излагается метод определения элементов, могущих быть неисправными. Метод использует информацию, получаемую в произвольном алгоритме обнаружения факта наличия неисправности.

Библ. 4 назв.