МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Министерство высшего и среднего специального образования Латвийской ССР
Латвийский ордена Трудового Красного Знамени государственный университет имени Петра Стучки
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Республиканский межведомственный сборник научных трудов

Под ред. Г.Л.Ионина

Латвийский государственный университет им.П.Стучки
Рига 1977

УДК 519.2; 681.142.2:

Настоящий выпуск сборника "Математическое моделирование, I" посвящен статистическому моделированию систем массового обслуживания. Рассмотрены вопросы математического обеспечения, разработки моделирующих программ и применения результатов моделирования при проектировании коммутационных систем.

Сборник предназначен для научных работников и инженеров, аспирантов и студентов, занимающихся или интересующихся статистическим моделированием или применением результатов моделирования.

Печатается по решению редакционно-издательского совета ЛГУ им. П.Стучки от 25 марта 1977 года

М 30501-057у 223-77
М 812(II)-77

АЛГОРИТМИЧЕСКИЙ ЯЗЫК А4

Седол Я.Я.
(ВЦ ЛГУ им.П.Стучки)

I. ВВЕДЕНИЕ

Алгоритмический язык как система правил и обозначений для записи алгоритмов необходим на всех этапах работы, связанной с вычислительными процессами. Особое значение алгоритмический язык имеет как средство записи программ для ЭВМ. Для этого язык дополняется транслятором - программой для перевода (трансляции) на машинный язык.

Алгоритмический язык А4 разработан в Вычислительном центре Латвийского государственного университета им. П. Стучки на основе ранее разработанного там же языка АЗ [I]. А4 предназначается для записи алгоритмов вычислительного характера (большой объем вычислений над небольшим количеством данных) и отличается от других языков подобного назначения (Алгол, Фортран и др.) простотой строения и высокой экономичностью. По таким показателям, как длина программы, расход перфокарт, объем транслятора и время трансляции, А4 экономичнее других известных языков в 2-7 раз. Язык А4 близок к общепринятой системе математических обозначений, в нем полностью устранены словесные выражения. Сравнительно небольшой объем "грамматики" языка А4 делает его легко изучаемым и уменьшает вероятность ошибок в программах. Простота и лаконичность А4 делает его удобным и для "внемашинных" применений - разработки, хранения и публикации алгоритмов.

Язык А4 машинно-независим и допускает реализацию практически на любой универсальной ЭВМ. В настоящее время единственный транслятор А4TI с языка А4 разработан на БЭСМ-4.

По сравнению с языками Алгол и Фортран язык А4 имеет некоторую ограниченность средств, в частности, в нем не предусмотрены действия с многомерными массивами, комплексными числами и символьной информацией. В перспективе возможно расширение языка А4 путем добавления перечисленных и некоторых других средств.

Структура языка А4 допускает возможность построения на его основе более специализированных языков. Пример такого построения - применение А4 для статистического моделирования [2].

2. ОСОБЕННОСТИ ОПИСАНИЯ

Понятия, связанные с алгоритмическим языком, делятся на формальные и содержательные. Формальные объекты предоставляют собой последовательности символов из специального набора - алфавита языка. Содержательные объекты составляют интерпретацию формальных, раскрывают их смысл.

В тексте описания применяются обобщенные формальные объекты, содержащие метасимволы. Метасимволы рассматриваются как переменные, принимающие в качестве значений конкретные формальные объекты. В качестве метасимволов используются не входящие в алфавит А4 буквы, в частности, малые латинские и греческие. Формальные объекты (в том числе обобщенные) отделяются от текста и друг от друга посредством фигурных скобок"{ } ". Если в одном контексте с формальным объектом упоминается входящий в него метасимвол без скобок, то он понимается в содержательном смысле.

В примерах формальных объектов иногда за заключительной скобкой будет дана интерпретация в обычных математических обозначениях. Тогда большим латинским буквам формального объекта соответствуют малые буквы интерпретации.

Сведения относительно реализации языка А4 посредством транслятора А4TI вынесены в абзацы с пометкой "А4TI".

3. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ

Рассмотрим некоторые понятия (содержательные), связанные с записью чисел. Для записи действительных чисел с заданной точностью применяются десятичные дроби, которые изображаются двумя основными способами – с фиксированной запятой (или точкой) и с плавающей запятой (точкой). Для краткости будем говорить о фиксированном и плавающем способе записи. Оба способа по существу представляют действительное число как пару целых чисел.

Фиксированный способ записи выделяет целую часть c и дробную часть d числа x. Пример: $x=-13{,}075$; $c=-13$; $d=075$.

Дробная часть может иметь слева нули, имеющие существенное значение. Знак числа приписывается целой части (возможен случай $c=-0$).

Плавающий способ представляет число x как пару из мантиссы m и порядка p согласно равенству

$$x = m \cdot 10^{p-q}$$

где q – число цифр в записи m (более наглядно, хотя не совсем точно, $x = 0{,}m \cdot 10^{p}$).

Пример: $x=-13{,}075$; $m=-13075$; $p=2$. Будем требовать, что мантисса не начинается нулем, тогда p определяется однозначно (исключение составляет $x=0$, тогда предполагается $p=0$, а m состоит из q нулей).

Длиной представления называется число цифр дробной части (при фиксированном способе) или число цифр мантиссы (при плавающем). Длина определяет точность представления числа. Если задана длина q, то представление числа однозначно с точностью до округления.

В таблице I приведены примеры представления чисел (без округления).

Аналогичное представление чисел возможно в других системах счисления, например в двоичной.

Таблица I. Представление чисел

x	q	c	d	m	p
3,I4I5	2	3	I4	3I	I
"	4	3	I4I5	3I4I	I
-25	3	-25	000	-250	2
$5 \cdot 10^{-7}$	I	0	0	5	-6
-0,03	2	-0	03	-30	-I

4. АЛФАВИТ

Алфавит языка А4 состоит из 64 символов, которые приведены в таблице 2. Одновременно таблица 2 дает способ кодирования символов. Код символа состоит из двух восьмеричных цифр (6 битов), первая цифра берется из строки вверху, вторая - из столбца слева. Пример: Символ {□} имеет код 35.

Таблица 2. Алфавит языка А4.

	0	1	2	3	4	5	6	7
0		H	R	Q	0	8	=	≠
1	A	J	S	W	1	9	+	±
2	B	∃	T	Ø	2	Δ	-	*
3	C	K	U	[	3	(	×	x
4	D	L	V	]	4	)	/	↑
5	E	M	X	□	5	;	>	≥
6	F	N	Y	'	6	:	<	≤
7	G	P	Z	␣	7	.	,	∗

Символ с кодом 00 называется пробелом и применяется как средство внешнего оформления; наличие пробелов не меняет смысл формального объекта. Остальные символы алфа-

вита называются А-символами, их конечные последовательности - А-словами. К А-словам относится также пустое слово, обозначаемое метасимволом Λ .

А-символы разделяются на буквы (коды 01-32), цифры (40-51) и знаки (остальные). Буква {Ø} перечеркивается для отличия от цифры {0}. Среди знаков выделяются знаки действий {+ - × / ≙ # ≙ ↑} и знаки отношений {= ≠ > < ≥ ≤}.

5. ИДЕНТИФИКАТОРЫ

Идентификатором называется А-слово из букв. Идентификаторы используются в качестве обозначений (наименований) некоторых объектов - массивов, функций, блоков.

Примеры идентификаторов: {A}, {BETA}, {LN} .

А4ТТ. Максимальная длина идентификатора - 7 букв.

6. ПЕРЕМЕННЫЕ И МАССИВЫ

Переменные являются основными содержательными объектами языка А4. Для удобства описания будем интепретировать переменные как ячейки некоторой воображаемой ЭВМ. Ячейки образуют линейную последовательность и имеют нумерацию 1,2,... Каждая переменная занимает одну ячейку. Номер ячейки называется адресом, а содержимое ячейки - значением переменной.

Совокупность рядом расположенных переменных называется массивом. Каждый массив имеет точку отсчета для внутренней нумерации, называемую центром или центральной переменной массива. Внутренний номер переменной, указывающий ее расположение относительно центра, называется смещением.

Предполагается, что каждая переменная входит в некоторый массив. В частности, индивидуальная переменная рассматривается как массив, состоящий из одной (обычно центральной) переменной.

7. ТИПЫ ПЕРЕМЕННЫХ

Будем рассматривать ячейку как линейную последовательность битов (двоичных разрядов) неопределенной длины n (рис.I). Нумерация битов начинается с младшего бита, т.е. справа налево.

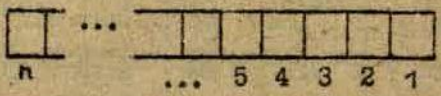


Рис.I. Ячейка.

Способ записи численных значений в ячейку зависит от типа соответствующей переменной. В языке А4 рассматриваются три типа, обозначаемые соответственно буквами {A}, {J}, {S} ;эти буквы называются типовыми буквами.

Тип связывается с массивом - каждый массив имеет свой тип, который распространяется на все переменные массива.

7.I. Тип А

Тип А (арифметический или действительный) представляет действительные числа плавающим способом. В ячейку записываются в закодированном виде двоичные мантисса и порядок представляемого числа. Способ кодирования и записи не уточняется (он зависит от конкретной реализации), поэтому значения типа А не доступны для логических действий на уровне битов. Тип А предназначается для арифметических вычислений.

7.2. Тип J

Тип J (индексный или целый) представляет целые числа фиксированным способом. Неотрицательное значение типа J записывается в двоичной системе и помещается в определенных битах ячейки. Номер v бита единиц, а также способ записи отрицательных значений типа J, не уточняется. Биты, не используемые в записи, заполняются нулями. Тип J пред-

назначается для переменных-индексов и других видов целочисленных переменных.

Пример. На рис.2 показана запись в ячейку числа I8 типа J.

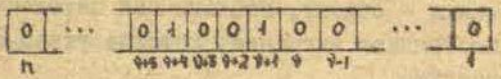


Рис.2. Запись числа типа J.

7.3. Тип S

Тип S (специальный или логический) представляет целые неотрицательные числа в виде их двоичной записи, т.е. в виде строк битов. Запись помещается в младших разрядах ячейки (так же, как тип J в случае $\nu = 1$). Тип S предназначается в основном для логических действий.

А4Т1. Для БЭСМ-4 длина ячейки $n = 45$. Число типа A записывается следующим образом: абсолютная величина мантиссы в битах 36-I, знак мантиссы в 44, порядок, увеличенный на 64, в 43-37. Бит 45 не используется. Для типа J $\nu = I3$, отрицательные значения записываются в дополнительном коде в битах 36-I3. Для записи содержимого ячейки применяется строка из I5 восьмеричных цифр, разделяемая (слева) на признак (I цифра), код (2 цифры) и три адреса (по 4 цифры). В таблице 3 приведены примеры записи чисел в ячейку.

Таблица 3. Запись чисел на БЭСМ-4.

Число	Тип	Запись				
18	A	1	05	4400	0000	0000
"	J	0	00	0000	0022	0000
"	S	0	00	0000	0000	0022
-13	A	3	04	6400	0000	0000
"	J	0	00	7777	7763	0000
0,1	A	0	75	6314	6314	6315

8. ТЕРМЫ

Термами называются А-слова из букв, цифр и знаков $\{\_\}$, $\{\Delta\}$, предназначенные для обозначения переменных и констант.

Константа - объект, аналогичный переменной, но принимающий постоянное значение, которое указывается прямо в записи константы.

Термы, изображающие переменные, называются n-термами. n-терм всегда начинается буквой, константа - цифрой или знаком $\{\Delta\}$.

8.1. Константы

В А4 применяются константы (в формальном смысле) трех видов - целые десятичные, плавающие десятичные и восьмеричные.

Целые десятичные константы, называемые также нумералами, изображают неотрицательные значения в общепринятом виде (не допускаются лишние нули слева), а отрицательные - с нулем вместо знака "-".

Примеры: $\{0\}$ 0, $\{13\}$ 13, $\{01\}$ -1, $\{0301\}$ -301.

Плавающая десятичная константа имеет вид $\{m\,\Delta\,p\}$, где $\{m\}$ и $\{p\}$ - нумералы, изображающие соответственно мантиссу и порядок значения константы.

Примеры: $\{5\,\Delta\,0\}$ 0,5; $\{01\,\Delta\,4\}$ -1000; $\{13\,\Delta\,07\}$ 0,13; $\{1\,\Delta\,06\}$ 10^7.

Восьмеричная константа допускает только целые неотрицательные значения и записывается в виде $\{\Delta\,a\}$, где a - обычная запись числа в восьмеричной системе.

Примеры: $\{\Delta\,10\}$ 8, $\{\Delta\,777\}$ 777_8=511.

Плавающие десятичные константы всегда имеют тип А. В остальных случаях запись константы не определяет ее тип и константа называется неопределенной. Тип неопределенной константы определяется по контексту (см. 18). Если необходимо явное указание типа константы, то применяется определенная константа, которая образуется от неопределенной до-

бавлением справа типовой буквы.

Примеры: {01J} , {125S}, {A37A}.

8.2. Имена

Массивы обозначаются идентификаторами, а обозначение или имя переменной имеет вид {am}, где {a} - идентификатор массива, {m} нумерал, указывающий смещение. Если m = 0, то часть {m} можно опускать - идентификатор массива одновременно служит именем его центральной переменной.

Примеры: {R3}a_3, {F010}f_{-10}, {LAM0}, {LAM}.
Две последние записи эквивалентны.

8.3. Индексаторы

Индексатором называется терм вида {ab} , состоящий из двух имен - основания {a} и индекса {b} . Основание пишется в полной форме (без опускания нуля). Индекс {b} обязательно имеет тип J . Индексатор обозначает переменную, адрес которой равен сумме адреса основания и значения индекса.

Примеры: {A0J}a_i , {F02 2 3}$f_{2_3 2_3}$, {RAT5TAR05}. .

8.4. Группы

Группой называется терм вида {a␣b}, где {a} - имя, {b} - либо нумерал (группа постоянной длины), либо имя (группа переменной длины), либо λ (неполная группа). В первых двух случаях группа называется полной.

Группа служит обозначением совокупности переменных с адресами от α до $\gamma + \beta$, где α - адрес переменной a , γ - адрес центра массива, к которому относится a , β - значение b . В случае неполной группы β устанавливается по контексту (см.I7).

Примеры: {A␣5} $a_0, a_1, \ldots, a_5$; {N2␣3} $n_{2_1}, n_{2_2}, \ldots, n_{2_3}$;
{G␣M2} $g_0, g_1, \ldots, g_{m_2}$; {AM1␣}.

9. ДЕЙСТВИЯ

В А4 предусмотрены следующие действия над переменными (таблица 4). Все действия двухместны, т.е. имеют два операнда и один результат.

Таблица 4. Действия.

	Знак	Название	Типы	Ранг
арифметические	+	сложение	AAA, JJJ SSS	1
	-	вычитание		
	×	умножение	AAA	2
	/	деление		
логические	$\underset{\sim}{+}$	логическое сложение	JJJ SSS	1
	#	сравнение		
	$\underset{\sim}{\times}$	логическое умножение		2
	↑	сдвиг	JJJ , SJS	3

В графе "типы" указаны допустимые типы операндов и тип результата.

Арифметические действия выполняются в общепринятом смысле. Логические операции (кроме сдвига) выполняются над парами соответствующих битов операндов по следующим правилам:

$$0 \underset{\sim}{+} 0 = 0;\ 0 \underset{\sim}{+} 1 = 1 \underset{\sim}{+} 0 = 1 \underset{\sim}{+} 1 = 1;$$
$$0 \# 0 = 1 \# 1 = 0;\ 0 \# 1 = 1 \# 0 = 1;$$
$$0 \underset{\sim}{\times} 0 = 0 \underset{\sim}{\times} 1 = 1 \underset{\sim}{\times} 0 = 0;\ 1 \underset{\sim}{\times} 1 = 1.$$

Сдвиг означает, что передвигается значение первого операнда на число битов, равное значению второго операнда. Положительному значению второго операнда соответствует сдвиг влево, отрицательному - вправо.

Ранги действий используются для установления приоритета действий в выражениях (см. II).

Примеры выполнения логических действий: $5 \underset{\sim}{+} 12 = 13$, $11 \underset{\sim}{\times} 7 = 3$, $15 \# 9 = 6$, $7 \uparrow 2 = 28$, $5 \uparrow 01 = 2$. Последний пример допустим только для типа S; в случае типа J результат не будет записью числа типа J.

10. ФУНКЦИИ

Функцией называется одноместное действие, имеющее один операнд (аргумент) и один результат. В А4 используется фиксированный набор функций, состоящий из элементарных функций и функций перевода. Каждая функция имеет свой стандартный идентификатор.

Список элементарных функций приведен в таблице 5. Аргумент и результат элементарной функции обязательно имеет тип А.

Таблица 5. Элементарные функции

Идентификатор	Название	Обычное обозначение
ABS	абсолютная величина	$\|x\|$
ENT	целая часть	$[x]$
SQR	квадратный корень	$\sqrt{x}$
EXP	экспонента	e^x
LN	натуральный логарифм	$\ln x$
SIN	синус	$\sin x$
COS	косинус	$\cos x$
TG	тангенс	$tg\,x$
ARS	арксинус	$arc\sin x$
ART	арктангенс	$arctg\,x$

Функции перевода преобразуют значения переменных из одного типа в другой. Идентификатор функции перевода состоит из двух типовых букв: первая - тип аргумента, вторая - тип результата. Всего имеется шесть функций перевода с идентификаторами {AJ}, {JA}, {AS}, {SA}, {JS}, {SJ}. Перевести в тип J разрешается только целые, а в тип S - только целые неотрицательные значения.

11. ВЫРАЖЕНИЯ

Выражение M называется A-слово, построенное согласно следующему индуктивному определению:

а) Терм является выражением.

б) Если {a} , {b} - выражения, {x} - знак действия, то {a x b} - выражение.

в) При тех же условиях {(a x b)} - выражение.

г) Если {a} - выражение, {f} -идентификатор функции, то {f(a)} - выражение.

Выражение рассматривается как предписание выполнить определенную последовательность действий над значениями входящих в выражение термов. Результат этих действий называется значением выражения.

Примеры выражений: {A0M} a_m , {0-R/2} - r/2 ,

{SQR(Z×Z'+EXP(H))} $\sqrt{z^2+c^2}$, {LA2␣M/(BA␣+C)} .

Порядок действий в выражении определяется скобками. Если скобок недостаточно, то в первую очередь выполняется условие с большим рангом, а если ранги одинаковы, то в порядке слева направо. Ранги действий даны в таблице 4 (раздел 9).

Пример. Выражение {A/3-B×C/5+1} равносильно {((A/3)-((B×C)/5))+1}.

I2. СПИСКИ

Списком называется А-слово, представляющее собой последовательность разделенных запятыми формальных объектов любого рода. Объекты, составляющие список, называются элементами списка, а их число - длиной списка. Длина списка, в частности, может равняться единице или нулю.

Примеры. {A, RGT, LN} - список идентификаторов, {1,2, 5Δ0,0} - список констант, {A+B} - список выражений, { } - пустой список.

I3. СТРОЕНИЕ ПРОГРАММЫ

Законченный алгоритм на языке А4 называется программой. Программа составляется из операторов. Операторы разделяются знаками {;} , точнее - знак {;} всегда является последним символом оператора.

I3.1. Блоки

Операторы объединяются в блоки. Блок представляет собой автономную часть программы для выполнения некоторой части алгоритма. Первый по порядку блок называется основным блоком программы. Работа программы заключается в выполнении основного блока, который использует (прямо или косвенно) другие блоки в качестве подпрограмм.

I3.2. Метки

Для разделения программы на блоки и для ссылок внутри блока некоторые операторы имеют метки. Метка пишется в начале оператора, после метки ставится точка. Метка представляет собой либо идентификатор (заглавная метка), либо положительный нумерал (внутренняя метка). Оператор может иметь не более одной метки.

Заглавная метка ставится к первому оператору каждого блока и представляет собой идентификатор (наименование) этого блока. Конец блока определяется либо заглавной меткой следующего блока, либо концом программы. Оператор с заглавной меткой называется заглавным оператором блока.

Внутренние метки образуют частичную нумерацию операторов блока. Внутренние метки в пределах блока должны быть различными.

Часть оператора, которая остается после удаления метки (если такая имеется) вместе с точкой и конечного {;}, называется командой.

Пример. Программа {AL. M=1; 1.M×2; M<N*1;} состоит из одного блока AL и трех операторов. Программа содержит три команды {M=1}, {M×2}, {M<N*1}.

I3.3. Управление

Управлением называется воображаемый объект, указывающий выполняемый в данный момент оператор (команду). При входе в блок управление передается заглавному оператору блока.

Будем считать, что в конце каждого блока стоит подразумеваемый оператор выхода, достижение которого рассматривается как завершение выполнения блока. Операторы блока выполняются в порядке их следования, за исключением случаев, когда выполняются команды, передающие управление (управляющие команды).

Работа программы начинается входом в основной блок и кончается достижением его выхода.

I4. СИСТЕМА КОМАНД

В А4 используются команды трех типов – команды определения, вычислительные команды и управляющие команды.

Команды определения задают типы и расположение массивов, вычислительные команды меняют значения переменных, а управляющие команды меняют ход управления.

Формальным признаком команд определения является наличие знака {:}, а управляющих команд – знака {*}. Исключением является пустая команда (см.20.), которая относится к управляющим.

I5. СИСТЕМА ОПРЕДЕЛЕНИЙ

Определением массива называется действие, которое устанавливает тип массива и резервирует ему место, определяя тем самым адрес центра массива. Определения производятся перед выполнением программы во время трансляции. Транслятор просматривает программу один раз в порядке ее написания и выполняет все необходимые определения. Каждый массив, упомянутый в программе, определяется либо посредством команды определения, либо автоматически.

I5.I. Команды определения

Команда определения имеет вид {:t:S}, где {t} – типовая буква, {S} – непустой список, элементами которого могут быть имена и группы постоянной длины.

Каждый элемент списка s определяет один массив типа t и устанавливает адрес его центра следующим образом. Пусть z – наибольший адрес, занятый ранее определенным массивом (в начале трансляции $z=0$). Рассмотрим случай, когда элемент списка s – группа вида $\{an \smile m\}$, где $\{a\}$ – идентификатор массива, $\{n\}$ и $\{m\}$ – нумералы (возможно $\{n\}=\lambda$, что равносильно $n=0$). Тогда адрес α центра массива a и новое значение z_1 для z определяются по формулам

$$\begin{cases} \alpha = z-n+1, \\ z_1 = \alpha+m \end{cases}$$

Если элемент списка – имя вида $\{an\}$, то полагается $m=n$

Пример. Пусть $z=I5$ и выполняется команда $\{:J: L \smile 3, A1, T3 \smile 6\}$. В результате определяются массивы L,A, T типа J с центрами в I6,I9,I8 соответственно; $z_1=24$. Более наглядно пример показан на рис.3.

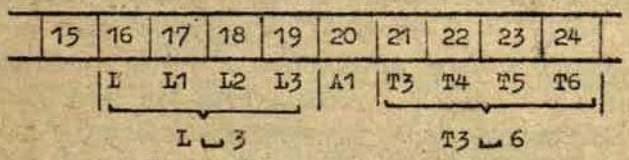


Рис.3. Пример определения .

Отметим, что одна и та же переменная может принадлежать разным массивам. В примере имена $\{L2\}$, $\{T\}$ и $\{A01\}$ обозначают одну и ту же переменную с адресом I8. Это возможно только тогда, когда "нарушаются" установленные границы массивов – имена $\{T\}$ и $\{A01\}$ в этом смысле "незаконные", хотя формально их применение и не запрещено.

Команды определения выполняются во время трансляции и не участвуют в выполнении самой программы. Допускается помещение команд определения в начале программы перед началом основного блока. Команда определения влияет только на ту часть программы, которая следует после нее. Массив может быть определен несколько раз – тогда в каждом месте программы имеет силу последнее из предыдущих определений.

I5.2. Автоматическое определение

Если в программе встречается не определенный ранее идентификатор массива, то происходит автоматическое определение. Появление не определенного идентификатора {a} равносильно выполнению команды {:t:a} , где t зависит от последней буквы идентификатора {a} - если она равна I, J, K, L, M или N, то t=I, в остальных случаях t=A.

Следует учитывать, что автоматическое определение резервирует место только для одной, центральной, переменной массива.

Пример. Оператор {AK1=0;} , написанный там, где идентификатор {AK} не определен, является ошибкой - автоматическое определение резервирует место для AK, но не для AK1.

I6. ВЫЧИСЛИТЕЛЬНЫЕ КОМАНДЫ

Вычислительная команда (в полной форме) имеет вид {a=b} , где {a} - n-терм, {b} - выражение того же типа. При выполнении команды вычисляется значение выражения b и заносится в a (знак {=} используется как знак присваивания). Сокращенная форма вычислительной команды состоит из одного выражения {b} , содержащего по крайней мере один n-терм. В этом случае роль {a} выполняет первый слева n-терм, входящий в {b}.

Примеры: {A=0}, {LA=202-LA}, {J+1}, {S3=1+2/S3} , {1+2/S3} . Последние две команды эквиваленты.

I7. ПРИМЕНЕНИЕ ГРУПП

Группы применяются в вычислительных командах согласно следующему правилу: первая слева группа в команде должна быть полной, все остальные - неполными. Конечные адреса неполных групп устанавливаются автоматически так, чтобы все группы в команде содержали одинаковое число переменных. Команда выполняется над соответствующими перемен-

ными групп в порядке возрастания адресов.

Примеры. {A␣J=0}, {B2␣100↑1/R01␣}, {S↑S1␣N2}. Последняя команда дает в S сумму значений всех переменных группы {S␣N2}.

А4Т1. Не допускается отрицательное смещение первой переменной полной группы. Не допускается наличие в одной команде с группой индексаторов и действий сдвига с отличным от константы вторым операндом.

Примеры ошибочных команд: {H01␣3×2}, {R␣3=TOM}, {N2␣J↑L}.

I8. ОПРЕДЕЛЕНИЕ ТИПОВ КОНСТАНТ

Тип неопределенной константы {m} определяется типом терма {a} той же команды, если выполняются следующие условия:

а) {a} является a-термом или определенной константой;

б) {a} расположен слева от {m};

в) {a} не заключен в скобки, вне которых находится {m} или наоборот;

г) между {a} и {m} нет других термов, выполняющих условия а)-в).

Если терма {a}, выполняющего перечисленные условия, в команде нет, то {m} имеет тип A.

Примеры. Пусть {J}, {S} - имена соответствующих типов. Тогда в команде {S=1↑(J+2)-3} константы {1} и {3} имеют тип S, а {2} - тип J. В команде {10-J} константа {10} имеет тип A, следовательно, команда ошибочна. Правильно: {10J-J} или {J=10-J}

I9. УПРАВЛЯЮЩИЕ КОМАНДЫ

Управляющие команды делятся на команды перехода, передающие управление в пределах блока, и команды обращения, устанавливающие связь между блоками. К управляющим командам относятся также команды вызова встроенных подпрограмм и пустые команды.

I9.I. Переходы

Будем считать, что заглавный оператор имеет подразумеваемую внутреннюю метку 0, а оператор выхода - пустую метку λ. Назовем ссылкой А4-слово, совпадающее с некоторой внутренней (действительной или подразумеваемой) меткой данного блока. Команда перехода имеет вид $\{*m\}$ (безусловный переход) или $\{a \times b *m\}$ (условный переход), где $\{m\}$ - ссылка, $\{a\}$, $\{b\}$ - выражения одного типа, $\{\times\}$ - знак отношения.

Безусловный переход передает управление оператору с меткой m. Условный переход делает то же самое в случае, если соотношение $a \times b$ между значениями двух выражений верно. Если это соотношение ложно, то управление переходит на следующий оператор.

Примеры: $\{A \neq B *3\}$, $\{*2\}$, $\{MOJ=1+R*0\}$, $\{*\}$. Последняя команда означает безусловный выход из блока.

I9.2. Обращения

Команда обращения имеет вид $\{*a\}$, где $\{a\}$ - идентификатор блока. Команда вызывает выполнение блока a, точнее: управление передается на вход блока a, а после достижения выхода блока a управление возвращается на следующий за обращением оператор.

Если блока с идентификатором $\{a\}$ в программе нет, то обращение работает впустую. Если в программе несколько блоков с идентификатором $\{a\}$, то они выполняются последовательно в порядке их следования; возврат происходит после выхода из последнего такого блока.

Не допускается прямое или косвенное обращение из блока к нему самому.

Пример. Программа $\{A.*C; *B; *D; *C; B.S+1; C.S \times R; B.R-1;$ равносильна $\{A.S \times R; S+1; R-1; S \times R;\}$.

I9.3. Встроенные подпрограммы

Встроенные подпрограммы (ВП) представляют собой стан-

дартные алгоритмы, которые имеются в готовом виде и добавляются к программе при трансляции. Каждая ВП имеет обозначение в виде А-слова из букв и (или) цифр. ВП включается в работу программы посредством команды вызова вида {*n*s} где {n} - обозначение ВП, {s} - список элементов информации для ВП. Строение списка {s} уточняется отдельно для каждой ВП.

Рассматриваемый вариант А4 имеет набор ВП для операций вывода, где в качестве обозначений ВП используются цифры 0,1,...,4.

Добавление новых ВП - одна из возможностей расширения языка А4. Пример такого расширения - язык А4М для моделирования [2].

В дальнейшем ВП с обозначением {n} изображается в виде ВП-n (например, ВП-0).

20. ЦИКЛЫ

В программах часто встречаются участки, называемые циклами типа "арифметическая прогрессия". Формально такой участок можно предоставить в виде

$$\{a=b-h;\ m.\ a+h;\ z\ a<c*m;\},\qquad (I)$$

где {a} - имя, {b}, {c}, {h} - имена или константы, {m} - метка, {z} - последовательность операторов. Содержательно: a - параметр цикла, b - начальное значение, c - конечное значение, h - приращение (шаг).

Для более краткой записи цикла в А4 предусмотрены команда открытия цикла вида {a=b,c,h} и пустая команда для закрытия цикла. Формально команда открытия цикла - вычислительная, пустая команда - управляющая.

Цикл (I) записывается в виде {a=b,c,h;z;} (обозначения прежние).

Пустая команда закрывает тот цикл, (если их несколько), который открылся последним (в порядке написания). Если пустая команда стоит на месте, где открытых циклов нет, то она работает впустую. Конец блока автоматически

закрывает все циклы, т.е. писать пустые команды в конце блока не обязательно. К пустой команде может быть метка.

Если $h=1$, то команду открытия цикла можно писать в сокращенном виде $\{a=b,c\}$. Отрицательные значения h не предусмотрены.

Команды открытия и закрытия циклов в А4 рассматриваются лишь как сокращения и не создают формальных ограничений на передачу управления через границы цикла, на изменение a,b,c или h внутри цикла и т.д.

Пример. Блок $\{C.K=1,3;\ L=A,B,R2;*N;;*M;\}$ означает $\{C.K=0;1.K+1;L=A-R2;2.L+R2;*N;L<B*2;*M;K<3*1;\}$.

21. ПРОСТОЙ ВЫВОД

Вывод (печать) результатов производится посредством ВП. Для вывода линейной последовательности значений в некоторой стандартной форме применяется простой вывод посредством ВП-0.

Команда простого вывода имеет вид $\{*0*S\}$, где $\{S\}$ - непустой список из имен и (или) групп, указывающий, что печатать.

Пример. Команда $\{*0*A\sqcup 5,N,T2\sqcup J\}$ печатает значения $a_0, a_1, \ldots, a_r, n, t_1, \ldots, t_i$.

А4Т1. Простой вывод означает "узкую" печать на бумажной ленте (см. 29.I.).

22. ВЫВОД НА АЦПУ

Алфавитно-цифровое печатающее устройство (АЦПУ) позволяет печатать информацию на широкой бумаге в наглядном и удобном для пользования виде. Предполагается, что АЦПУ печатает строки из 128 символов, где набор символов содержит алфавит А4 и большие буквы русского алфавита (это близко к АЦПУ машины БЭСМ-4). Позиции символов в строке нумеруются слева направо числами 1,2,...,128.

Программирование вывода на АЦПУ сводится к применению двух основных действий - занесения и закрытия строки.

Занесение задает символьные значения позициям строки АЦПУ. Закрытие строки означает печать заполненной посредством занесений строки, после чего начинается заполнение новой строки. Новая строка перед заполнением автоматически содержит во всех позициях пробелы.

22.1. Команды занесения

Для заполнения строки АЦПУ применяются ВП-1 и ВП-2. Команда занесения имеет вид {*1*S} или {*2*S}, где S - непустой список элементов информации.

Действия ВП-2 отличаются от ВП-1 тем, что информация заносится в двух экземплярах - в позициях 79-128 автоматически повторяется то, что занесено в позиции 1-50. ВП-2 позволяет получить два экземпляра результатов в случаях, когда ширина печатаемого материала не превышает 50 символов.

Строение списка {S} для обоих ВП одинаково.

22.2. Элемент информации

Элемент списка {S} для ВП-1 и ВП-2 в общем случае имеет вид {a'b'c} или {'a'b'c}, где А-слова {a}, {b} и {c} не содержат апострофов {'}. Части {b} и {c} могут быть пустыми, тогда опускаются и апострофы, оказавшиеся в конце элемента (за исключением случая {'a'}).

Часть {a} указывает объект занесения, {b} - место (позицию) в строке, {c} - способ кодировки (формат) информации.

22.3. Объект занесения

Если элемент информации не начинается апострофом, то он означает занесение для печати численного значения некоторой переменной. В этом случае {a} - имя этой переменной (отличные от имен термы не допускаются).

Если же элемент начинается апострофом, то {a} называется литералом. Литералом может быть любое А-слово, не содержащее апострофов. Элемент информации в этом случае означает занесение символьной информации, которая прямо

указывается в литерале. В следующих случаях литерал при занесении перекодируется:

а) Если литерал содержит знак $\{\square\}$, то вместо него заносится пробел, т.е. знак $\{\square\}$ служит для кодировки пробелов в литералах.

б) Буква $\{P\}$ в формате (см. 22.5) переключает алфавит на русский вариант согласно таблице 6 (первая графа - написано в литерале, вторая - заносится для печати).

Таблица 6. Русский вариант алфавита

B	Б	J	Й	V	В	±	Ш
C	Ц	L	Л	Z	З	#	Ч
D	Д	N	Н	X	Я	×	Щ
F	Ф	P	П	Y	Ю	≥	Э
G	Г	R	Р	Q	Ь	≤	'
H	Х	S	С	W	Ж	≠	¤
I	И	U	У	␣	Ы		

Символы, не указанные в таблице, не перекодируются.

Пример. Команда $\{*1*'$KVADRATN␣J□KØRENQ''P$\}$ заносит русское словосочетание "квадратный корень".

22.4. Позиция

Часть $\{\beta\}$ элемента информации указывает номер ν позиции, с которой начинается занесение (кроме одного, рассмотренного в 22.5 случая, когда ν указывает конец занесения).

В качестве $\{b\}$ может быть нумерал, имя типа J, или λ. Пусть τ - значение b. Если $\tau>0$, то $\nu=\tau$. Если $\tau\le 0$, то $\nu=q+|\tau|+1$, где q - позиция последнего (правого) символа, занесенного при выполнении предыдущего элемента информации. Если $\{b\}=\lambda$, то $\nu=q+2$, т.е. если позиция не указана, то занесение продолжается с пропусканием одной позиции.

Занесение информации на место, где уже что-то занесено, стирает "старую" информацию. Занесение на позиции вне пределов 1-128 символы пропадают.

Пример. Команда $\{*1*'$AB$'125,'1234'126\}$ заносит в

позиции 125-128 символы A123.

22.5. Формат

Часть $\{c\}$ состоит из трех подчастей, точнее $\{c\} = \{\alpha\beta\gamma\}$, где $\{\alpha\}$ называется основанием, $\{\beta\}$ – длиной, а $\{\gamma\}$ – дополнением формата. Каждая из них может быть пуста.

Основание $\{\alpha\}$ состоит (если оно не пусто) из одной буквы A,B,C,E,F или H. Основание пишется только в случае занесения численного значения и указывает форму представления числа согласно таблице 7.

Таблица 7: Основания формата

α	Способ представления числа
A	Плавающий, порядок из одной цифры
B	Плавающий, порядок из двух цифр
C	Целая часть
E	Фиксированный с точкой
F	Фиксированный с запятой
H	Восьмеричный

Длина формата $\{\beta\}$ – неотрицательный нумерал или λ, что равносильно $\beta = 0$. Значение β в случае оснований A,B,E,F означает длину представления (см.3); в случаях E и H берется β последних цифр целой части или восьмеричной записи числа соответственно, добавляя нули, если цифр меньше чем β.

В таблице 8 показано, что именно и в каком порядке заносится для печати численных значений.

В случае $\alpha = H$ допускается только тип s. В остальных случаях значения могут быть любого типа и заносятся в десятичной системе. Знак числа (мантиссы, целой части) добавляется слева только в случае "-", знак порядка заносится всегда. Если $\beta = 0$, то точка или запятая в случаях E и F не заносится.

Для литералов основание формата не пишется, а длина

ρ используется как коэффициент повторения - литерал повторяется $\rho+1$ раз.

Таблица 8. Занесение чисел

α	Что заносится		
A	мантисса из ρ цифр	знак порядка	одна (последняя) цифра порядка
B			порядок из двух цифр
C	ρ последних цифр целой части		
E	целая часть	точка	дробная часть из ρ цифр
F		запятая	
H	ρ последних восьмеричных цифр		

Пример. Команда {*1*'A2*'1'2} заносит в начале строки A2*A2*A2* .

Дополнение формата $\{\gamma\}$ является A-словом из букв P, R, S, причем каждая из них входит в $\{\gamma\}$ не более одного раза.

Буква P при основаниях A, B, E, F означает, что число ν (см. 22.4) рассматривается как позиция последнего, т.е. правого заносимого символа, а не первого, как в остальных случаях.

Для литерала буква P означает русский вариант.

Буква S блокирует округление численных значений, для литералов не применяется.

Буква R после занесения выполняет действие закрытия строки, т.е. строка печатается и открывается для занесений новая строка.

Если при занесении численных значений формат пуст или состоит из одной лишь буквы R, то недостающая часть формата переносится от предыдущего элемента.

Пример. Команда {*2*A'1'E3P,B,C"R} равносильна {*2*A'1'E3P,B"E3P,C"E3PR} .

В таблице 9 показан пример напечатанной на АЦПУ численной информации при выполнении указанной команды. Предполагается, что переменные K, T, U, Q имеют следующие значения и соответствующие им типы: K=10J, T=3141595 Δ 1, U=017A, Q= Δ 12345S.

Таблица 9. Пример печати на АЦПУ

Команда занесения	Позиции																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
*1*T'5'A4R,					3	1	4	2	+	1								
T'K'B4SR,										3	1	4	1	+	0	1		
T'K'E2PR,							3	.	1	4								
T'3'C3R,			0	0	3													
U'K'ER,										-	1	7						
U'K'F2PR,					-	1	7	,	0	0								
Q'' H2R,												4	5					
Q'2'H6R,		0	1	2	3	4	5											
Q'04'R												0	1	2	3	4	5	

22.6. Строки и страницы

Закрытие строки можно делать двумя способами – с помощью рассмотренного в 22.5 дополнения R к формату при занесении и посредством специальной ВП-3. Команда вызова ВП-3 имеет вид $\{*3*n\}$, где $\{n\}$ – положительный нумерал или λ. Выполнение команды закрывает строку и, если $\{n\}$ отлично от λ, то, кроме того, передвигает бумагу (делает интервал) на n строк.

Если требуется разделение печатаемого материала на страницы, то применяется ВП-4, закрывающая страницу. Команда вызова имеет вид $\{*4*\}$ с пустым списком информации. Закрытие страницы означает передвижение бумаги на определенное число строк и печать горизонтальной разделительной черты.

Пример. Пусть А – блок, вычисляющий элементы a_{ij} некоторой матрицы по заданным i, j. Требуется напечатать на АЦПУ матрицу $\|a_{ij}\|$ размера 10x10 с плавающим представлением длины 5 и окончить страницу. Обозначения: $\{J\}i$, $\{J\}j$, $\{A\}$ – результат работы блока А. Решение: {J=1,10; K=1; J=1,10; *A;*1* A'K'A5; K+8;; *3*;; *4*;} .

23. ИСХОДНЫЕ ДАННЫЕ

Перед началом работы программы некоторым переменным присваиваются значения, называемые исходными данными. Исходные данные прилагаются к программе и обрабатываются транслятором после трансляции программы.

Набор исходных данных представляет собой последовательность операторов ввода. Оператор ввода имеет вид $\{a:s;\}$; где $\{a\}$ - имя, $\{s\}$ - непустой список констант. Оператор ввода присваивает переменной a первое значение из списка s , потом - следующей за a переменной второе значение из s и т.д. до конца списка s . Тип вводимых значений совпадает с типом переменной a .

Пример. Требуется ввести исходные значения $i=5$, $a_1=0{,}3$; $a_2=-13$, $a_3=10^7$.

Решение: $\{I:5;\ A1:3\Delta0{,}043,\ 1\Delta8;\}$.

24. НЕФОРМАЛЬНЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ

В настоящем разделе рассматриваются некоторые прямо не связанные с языком А4 средства, повышающие наглядность программирования. Применение этих средств не обязательно, но в случае сложных программ они необходимы в качестве ориентиров для самого программиста и других пользователей программы. Неформальные элементы программирования - таблица массивов и блок-схема - не входят в программу, а составляют сопровождающую документацию, которая разрабатывается вместе с программой.

24.1. Таблица массивов

Одна из первых задач, возникающая при составлении новой программы, заключается в построении "материальной части" программы - набора переменных, используемых программой. Ввиду того, что каждая переменная входит в массив, задача сводится к составлению таблицы массивов (ТМ), в которой перечислены все используемые массивы.

Основную часть ТМ целесообразно заполнять до начала

программирования; в дальнейшем при необходимости ее можно дополнить. На основании ТМ в окончательной программе пишутся команды определения.

ТМ содержит следующие графы: идентификатор, тип, границы "от" и "до", назначение, примечания.

В графах "границы" указываются смещения первой и последней переменных массива. В графе "до" можно писать выражения, составленные из ранее упомянутых в ТМ переменных. При написании команд определения эти выражения заменяются на их максимальные возможные значения. Если массив состоит из одной лишь центральной переменной, то в графах "границы" ставятся черточки.

В графе "назначение" пишется словесное описание роли массива в программе.

В "примечаниях" пишутся сведения, относящиеся к группам массивов, например, указываются массивы исходных данных и результатов.

Пример ТМ приведен в 25.

24.2. Блок-схема

Алгоритмическую часть программы, которая остается после удаления операторов определения, удобно представить в виде блок-схемы. Блок-схема строится отдельно для каждого блока и представляет собой графическое изображение, показывающее пути перемещения управления в блоке.

Блок-схема строится из элементов, начертания которых показаны на рис.4.

Рис.4. Элементы блок-схемы

В прямоугольных клетках прямых участков пишутся вычислительные команды, а также команды обращения и вызова ВП. Каждая команда пишется в своей строке, поэтому разделять их знаками {;} не обязательно.

В клетках-параллелограммах разветвлений пишутся условия - части команд условного перехода вида $\{a \times b\}$, где $\{a\}, \{b\}$ - выражения, $\{\times\}$ - знак отношения. Боковой выход соответствует выполнению условия, нижний - невыполнению. Пустая команда закрытия цикла изображается пустым параллелограммом.

Над черточкой входа пишется идентификатор блока. Каждый блок должен иметь ровно один вход; выходов может быть несколько.

Слияние применяется в случаях, когда управление из разных частей блок-схемы далее следует по одному общему пути.

Метки и безусловные переходы в блок-схемах не изображаются, их роль выполняют стрелки.

Программу целесообразно составлять сначала именно в виде блок-схемы, а потом по ней писать окончательную программу. В начальных стадиях разработки в клетках схемы можно писать неформальные словесные описания частей алгоритма. Пример блок-схемы приведен в 25 (рис.5).

25. ПРИМЕР ПРОГРАММЫ

Составим на языке А4 программу решения следующей задачи. Пусть $x(a)$ - наименьший положительный корень уравнения $\sin x = ax$ $(0 \le a < 1)$. Требуется вычислить и напечатать на АЦПУ таблицу значений $x(a)$, где a меняется в некотором интервале $d \le a < d_1$ с шагом 0,001. Числа d и d_1 задаются с двумя цифрами после запятой. Форма таблицы на примере $d=0{,}50$; $d_1=0{,}53$ показана в таблице 10.

Вместо показанных в таблице 10 линий следует печатать интервалы, а вместо нулей внутри таблицы - значения $x(a)$

Решение уравнения будем программировать способом деления интервала пополам, в качестве исходного интервала

полагается (0; 3,15).

Программирование начинается составлением ТМ (таблица II).

Таблица I0. Форма вывода результатов

	0	1	...	9	
0,50	0,00000	0,00000	...	0,00000	0,50
0,51	0,00000	0,00000		0,00000	0,51
0,52	0,00000	0,00000		0,00000	0,52
	0	1		9	

Таблица II. Таблица массивов

Идентификатор	Тип	Границы от	Границы до	Назначение	Примечания
A	A	–	–	аргумент a	
B	"	–	–	a с точностью до 0,01	
C	"	–	–	цифра тысячных числа a	
D	"	0	1	заданные границы d и d_1	исходные данные
E	"	–	–	верхняя граница для B	
P	"	–	–	$ax - \sin x$	
X	"	–	–	x	результат
G	"	–	–	верхняя граница интервала для X	
H	"	–	–	интервал деления	
K	I	–	–	позиция на АЦПУ	

Далее разрабатывается блок-схема программы. Программа составляется из четырех блоков: A – основной блок, печатающий таблицу, C – печать значений C вверху и внизу таблицы, R – решение уравнений методом деления интервала, F – вычисление функции $f(a,x) = ax - \sin x$. Блок-схема представлена на рис.5.

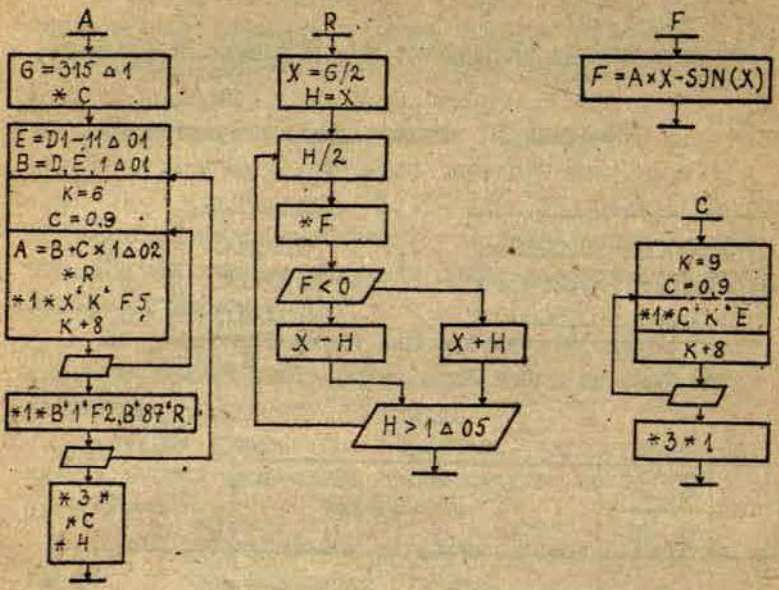


Рис.5. Блок-схема

Теперь на основании ТМ и блок-схемы нетрудно написать программу, которая представлена в приложении I. Команда определения требуется лишь для массива D, для остальных подходит автоматическое определение.

Пример набора исходных данных (d=0,25; d_1=0,50):
{D:25 Δ 0, 5 Δ 0}.

26. ТРАНСЛЯТОР А4Т1

В этом и следующих разделах рассмотрено практическое применение языка А4 на ЭВМ БЭСМ-4.

Транслятор А4Т1 состоит из 1770 машинных слов (МС). МС БЭСМ-4 содержит 45 битов. А4Т1 содержит собственно транслятор с языка А4 (980 МС), дополнение транслятора для статистического моделирования (440 МС) и подпрограммы вывода (350 МС). Во время трансляции А4Т1 вместе с программой помещается в оперативную память ЭВМ. Транслируемая исходная

программа просматривается транслятором всего один раз, поэтому транслятор работает сравнительно быстро - время трансляции программы средней длины(≈ 1000 символов) 20-30 сек. После трансляции программа сразу выполняется, сохранение протранслированной программы для многократного использования не предусмотрено.

А4Т1 помещается на магнитной ленте (МЛ).

27. ЗАПИСЬ НА БЛАНКАХ И ПЕРФОРАЦИЯ

Программа для А4Т1 пишется на бланках, имеющих строки по 42 символа. Образец бланка показан в приложении I.

Записанная на бланке информация переносится на перфокарты (ПК). Для этого используется кодировка символов, показанная в таблице 2 (раздел 4). Перфорация производится в режиме "команда". Одно МС, соответствующее одной строке на ПК, вмещает 7 символов - один в коде и по два в каждом из трех адресов. Признак (старшие три бита МС) не используется. Каждые две строки бланка соответствуют одной ПК. На каждой ПК обязательно пробиваются все 12 строк.

Для внешнего оформления программы на бланке рекомендуется следующие (необязательные)правила.

а) Каждый блок программы начинается на новой ПК.

б) Все метки помещаются в левых концах строк. Заглавные метки пишутся начиная с первого столбца, а внутренние - начиная со второго. В строках, не имеющих меток, первые два-три столбца не заполняются.

в) Каждый оператор помещается по возможности в одной строке.

г) Операторы, а также элементы списков, разделяются пробелами.

Пример программы в приложении I написан в соответствии с этими правилами. В таблице 12 показана кодировка первой ПК этой программы при перфорации.

Таблица I2. Пример кодировки

		0056	0156	0004
	37	4155		
0				
0				
0				
0				
	01	5707	6043	4145
	52	4155	0077	0355
		0560	0441	6241
	41	5240	4155	
0				
0				

28. КОМПЛЕКТАЦИЯ И ЗАПУСК

Вводимая в машину программа комплектуется из ПК, содержащих программу и исходные данные на языке А4, и из специальных обслуживающих карт. Используются следующие обслуживающие карты: L (личная карта), Σ (сигма) и В. В картах Σ и В пробито по одной строке: в Σ - основной и вспомогательный маркеры, в В - основной маркер и 7 в признаке. Строение карты L показано в таблице I3 (N - личный номер программиста в двоично-десятичном коде).

Таблица I3. Личная карта

	50	0420	0002	7200
	70	0144	0001	
	57	4030		0006
	50	0410	0626	7441
	70	7350	0004	
3	42	3623	6776	0763
	56	7360	0143	0143
	01	N		
Σ	01	N		

Вводимый в машину набор ПК имеет следующее строение:
L; программа, Σ данные-1, Σ ;...; данные-n, B, Σ .

Здесь n - количество наборов исходных данных, которые обозначены как данные-1,..., данные-n. Знаками ";" разделены участки, каждый из которых вводится в один прием. Эти участки обязательно отделяются друг от друга посредством 2-3 пустых ПК. Карта B служит признаком того, что введенный набор данных - последний. Если предусмотрена печать листинга (см.29.4), то перед программой ставится еще карта B.

При запуске программы происходит следующая последовательность действий ЭВМ:

а) Вводится карта L и вызывает считывание с МЛ транслятора АЛГЕК.

б) Вводится и транслируется программа.

в) Вводится и обрабатывается набор исходных данных.

г) Выполняется программа.

д) Если набор исходных данных имел карту B, то полний останов. В противном случае возврат к в) для ввода следующего набора.

29. ОСОБЕННОСТИ ВЫВОДА

Результаты работы ЭВМ печатаются на ленте (узкая печать) и на АЦПУ. Рассмотрим каждый вид печати в отдельности.

29.1. Узкая печать

При запуске программы на ленте всегда печатается следующая информация.

а) Заголовок, содержащий личный номер программиста и (возможно) дату запуска.

б) Контрольная сумма (точнее: дополнение контрольной суммы) вводимой программы в окаймление строк из семерок.

в) Таблица блоков (ТБ), рассмотренная в 29.2.

После ТБ печатается информация, выдаваемая программой посредством простого вывода. Рассмотрим простой вывод более

подробно.

БЭСМ-4 имеет два режима узкой печати - восьмеричную (печатается содержимое ячейки в виде строки из I5 восьмеричных цифр) и десятичную (печатается десятичное число в плавающем представлении). Десятичная печать печатает строки вида

$$\pm\alpha\beta pm$$

где α - знак мантиссы, β - знак порядка, p - порядок из двух цифр, m - мантисса из 9 цифр. Например, -3,I5 печатается в виде +-+0I 3I5000000.

Средствами десятичной печати можно напечатать также строки, в которых цифры порядка (кроме первой) и мантиссы заменяются пробелами или знаками "+", "-".

Печать чисел типов A, J посредством простого вывода происходит в десятичном виде, а чисел типа S - в восьмеричном. Тип A печатается в плавающем представлении, а тип J - как целое число в конце строки. Одна команда простого вывода печатает одну "порцию" чисел на ленте, за исключением случаев перехода от десятичной печати к восьмеричной и обратно. Для разделения напечатанного материала по элементам команды вывода печатаются разделители - строки вида ---3 в десятичной и 777777777777777 в восьмеричной печати.

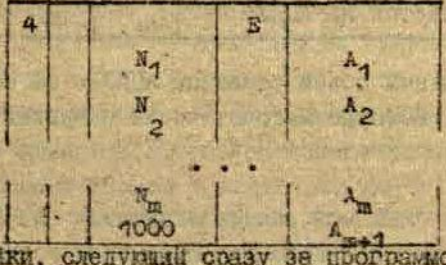
Пример. Пусть A,J,S - массивы соответствующих типов. Выполняется последовательность операторов {A=1/3; A1=A-2; J=3; J1␣2=0120; S␣2=JS(ABS(J␣)); #0# A␣1, J␣3, S, S1␣2;}. В результате напечатается следующая информация

+	+	+	0	0	3	3	3	3	3	3	3	3	3
+	-	+	2	1	1	6	6	6	6	6	6	6	7
-	-	-	3										
													3
											-	1	2	0
											-	1	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
0	0	0	0	0	0	0	0	0	0	0	17		0	

29.2. Таблица Блоков

Печатаемая на ленте ТБ отражает расположение протранслированной программы в памяти машины. Программа располагается в нулевом кубе (БЭСМ-4 имеет два куба - нулевой и первый - по 4096 ячеек) начиная с адреса 0200 (адреса ячеек куба задаются в виде четырехзначных восьмеричных чисел).

ТБ имеет следующее строение:



где E - адрес ячейки, следующей сразу за программой (с адреса E начинается размещение массивов, с которыми работает программа), m - число блоков, N_i - номер, A_i - адрес начала i-го блока программы. Последняя строка ТБ отражает добавляемый в ходе трансляции блок с номером 1000, который связывает программу с подпрограммами вывода (последние размещаются в первом кубе).

Номера блоков вырабатываются транслятором. Если идентификатор блока состоит из одной или двух букв, то номер является дополнением до 7777 кода идентификатора. Например, блок A имеет номер 7776, блок LW - 6346.

Блоки с более длинными идентификаторами нумеруются числами 2001, 2002, ... в порядке появления идентификаторов в программе.

ТБ печатается в ходе трансляции.

29.3. Печать на АЦПУ

Печать информации на АЦПУ происходит только тогда, когда это предусмотрено программой, а также в случае печати листинга (см.29.4). Перед информацией автоматически печатается личный номер программиста.

Ввиду того, что набор символов АЦПУ не охватывает все

символы алфавита А4, в некоторых случаях имеется несоответствие между предусмотренными и фактически печатаемыми символами. Эти случаи показаны в таблице I4.

Таблица I4. Соответствие символов

Символы А4	□	Δ	#	≥	≤
Печатается на АЦПУ	—	↑	=	≥	≤

Нормальный объем страницы АЦПУ - 64 строки (не считая интервалов, которые автоматически оставляются по обе стороны от разделительной черты). Нормальная страница получается только в случае, когда в момент выполнения команды {*Ч*} в заполняемой странице имеется от 56 до 64 оконченных строк. В остальных случаях разделительная черта печатается через три интервала после последней строки.

29.4. Листинг

Листингом называется распечатка программы на АЦПУ, выполняемая во время трансляции. Признаком печати листинга служит карта В, вставляемая непосредственно перед программой.

Листинг печатается в двух экземплярах (в двух параллельных колонках). Расположение символов в листинге соответствует записи программы на бланках. В левой стороне печатается нумерация перфокарт программы в порядке 1,2,... Программа разделяется на страницы по 32 перфокарты.

После программы печатаются исходные данные. Исходные данные печатаются на новой странице и нумерация перфокарт опять начинается с единицы.

Если требуется печать листинга без выполнения программы, то после программы ставится карта В, т.е. комплект перфокарт имеет вид L;B, программа, B, Σ ;

Пример листинга (в одном экземпляре) показан в приложении 2.

30. ОТЛАДКА

На практике первый запуск программы обычно из-за ошибок в программе дает неправильный результат или вообще не дает результата. Иногда программа работает, но нет уверенности в том, что она работает правильно. Доведение программы до несомненно исправного состояния называется отладкой.

В настоящее время нет программных средств контроля "грамматической" правильности программ на языке А4. Грубые ошибки могут вызвать сбой при трансляции - об этом свидетельсвует отсутствие на ленте ТБ и неоконченный листинг. Чаще всего неправильная программа протранслируется до конца и начинает работать.

После того, как программа по тем или другим причинам кончила работу, на пульте машины нажимается кнопка оператора (КО), что вызывает печать (узкую) информации о причинах останова. Рассмотрим эту информацию более подробно. Заголовок информации имеет вид

```
+ + +  0  + + +  + + +
+ + +  0  +   +  +   +
+ + +  0  +   +  + + + i
```

где цифра i указывает причину останова. Наиболее типичными являются случаи $i=1$ (автоматический останов), $i=3$ (блокировка) и $i=6$ (цикл). После заголовка печатаются две восьмеричные строки, из которых первая указывает состояние машины, а вторая - команду, на которой программа остановилась (или остановлена). Пусть эти строки имеют вид

-	-	-	A	-
-	K	-	-	-

(2)

(черточки указывают, что содержание может быть произвольным). Адрес А является адресом команды, на которой программа остановлена, что позволяет с помощью ТБ найти блок, содержащий эту команду. Рассмотрим теперь причины останова в отдельности.

Автоматический останов (авост) происходит в случае,

когда встречается невыполнимая команда, как правило арифметическая. Код операции этой команды может быть 01 (сложение), 02 (вычитание), 04 (деление), 05 (умножение) или 44 (квадратный корень). Причиной аварста может быть деление на нуль, квадратный корень из отрицательного числа или переполнение, т.е. результат, который превосходит по абсолютной величине наибольшее представимое на БЭСМ-4 число (такое число имеет десятичный порядок 19). В случае аварста после (2) печатаются в десятичном виде числа, над которыми производилась невыполнимая операция.

Блокировка означает команду останова с кодом К=77. В случае полного останова А=0175, а вторая строка (2) имеет вид 0 77 7777 7777 7777. Полный останов означает, что программа нормально окончила свою работу.

Блокировка, отличная от полного останова, возникает в случае выполнения действий умножения и деления над числами типа J,S. Блокировка возможна также в случаях выхода управления вне программы.

Цикл означает безрезультатную работу программы "на месте". В этом случае машина останавливается извне. После (2) в этом случае печатается участок (протранслированной) программы по обе стороны от указанной команды.

Может оказаться, что после КО ничего не печатается. Это происходит в случаях, когда программа, обрабатывая некоторый массив, выходит за пределы массива и "съедает" расположенную в конце нулевого куба программу для КО.

Информация, напечатанная после КО, в простейших случаях является достаточной для выявления ошибок. Основным способом отладки, однако, является расстановка наблюдателей - команд простого вывода, которые печатают необходимую для отладки информацию в ходе работы программы. После отладки наблюдатели удаляются. Наблюдатели можно ставить в конце программы, используя свойство обращения к одноименным блокам. Пусть, например, в программе есть блок 1М, после выполнения которого предполагается печатать массив L до индекса M. Тогда достаточно добавить к программе блок-наблюдатель {1М: for L:=L to M;} .

31. ЗАКЛЮЧЕНИЕ

Опыт применения А4 показал, что на языке А4 можно программировать быстро, экономично и практически без ошибок. В более сложных случаях сказывается ограниченность средств языка А4 и в этом плане А4 уступает языкам типа АЛГОЛ и ФОРТРАНА, а также системе ЈА - программирования [I] . Некоторые наиболее существенные недостатки А4 - отсутствие динамического распределения памяти, автоматического синтактического контроля и специальных средств отладки - намечено устранить в ходе дальнейшего совершенствования языка и транслятора. В настоящее время А4 находится в стадии развития и в перспективе может стать языком программирования высокого уровня, сохраняя простоту и экономичность.

Литература

I. Ионин Г.Л., Седол Я.Я. Программирование и статистическое моделирование на БЭСМ-4. Рига, ЛГУ им.П.Стучки, 1976. 204 с.

2. Седол Я.Я. Язык статистического моделирования А4М. Настоящий сборник, с.76 - 95.

Приложение I

Бланк А4 с програмой

```
A-4   БЭСМ-4

1      :A: D_1;
     A. G=315Δ1;  *C;  E=D1-11Δ01;
2      B=D,E, 1Δ01;  K=6;  C=0,9;
       A=B+C×1Δ02;  *R;  *1* X'K'F5;  K+8;  ;
3      *1* B'1'F2,  3'87'R;  ;
       *3*;  *C;  *4*;
4    C. K=9;  C=0,9;  *1* C'K'E;
       K+8;  ;  *3*1;
5    R. X=G/2;  H=X;
       1. H/2;  *F;  F<0*2;  X-H;  *3;
6      2. X+H;
       3. H>1Δ05*1;
7    F. F=A×X-SIN(X);

D    D: 2,5Δ0, 5Δ0;
```

Приложение 2

Пример листинга

```
1     :A: D=1;
    A.G=315*1; *G; E=D1-11=01;
2     B=D,E,1=01;   K=6; C=0,9;
      A=B+C-1=02; *R; *1* X'K'F5; K+8; ;
3     *1* B'1'F2, B'87'R; ;
      *3*; *C; *4*;
4   C.K=9; C=0,9; *1* C'K'F;
      K+8; ; *3*1;
5   R.X=6/2; H=X;
     1.H/2; *F; F<0*2; X-H; *3;
6     2.X+H;
      3,H>1=05*1;
7   F.F=A*X-5|N(X)|
```

```
1   D: 25=0, 3=0;
```

ОПИСАНИЕ И МОДЕЛИРОВАНИЕ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ

Ионин Г.Л., Седол Я.Я.
(ВЦ ЛГУ им.П.Стучки)

I. ВВЕДЕНИЕ

Системы массового обслуживания (СМО) [I] охватывают широкий класс явлений из самых различных сфер человеческой деятельности. В частности, к СМО относятся телефонные и телеграфные системы связи, для исследования которых создан специальный раздел теории массового обслуживания - теория телетрафика.

Применяемые на практике СМО обычно имеют очень сложное строение, что ограничивает возможность их аналитического исследования. Универсальным и во многих случаях единственным методом исследования СМО является статистическое моделирование или метод Монте-Карло. Метод заключается в построении модели СМО в виде программы для ЭВМ. В результате статистического моделирования получаем совокупность оценок величин, характеризующих моделируемую систему.

Составление программы моделирования для сложной СМО - процесс очень трудоемкий, причем основные трудности связаны с переводом на математический язык элементов реальных СМО. Для преодоления этих трудностей необходим способ стандартного описания СМО, некоторый формальный язык, охватывающий, с одной стороны, существенные свойства всего многообразия реальных СМО, а с другой стороны, близкий к алгоритмическим языкам программирования.

Известные в настоящее время языки для моделирования СМО и других сложных систем - Симула [2], Сиискрипт [3] и другие [4,5] - имеют ряд недостатков, препятствующих их широкому практическому применению. Главные из них - очень сложный аппарат построения моделей и отсутствие не-

посредственной наглядной взаимосвязи модели с моделируемой системой.

В настоящей работе изложена разработанная авторами система понятий, которая дает очень простой и наглядный способ описания СМО и, кроме того, является средством перехода от СМО к программе моделирования. Предусмотрены различные уровни формализации, позволяющие постепенно переходить от словесного описания СМО к программе моделирования на алгоритмическом языке. В настоящей работе не рассмотрен самый высокий уровень этой формализации, поэтому она не является руководством по программированию моделей СМО. Цель настоящей работы - описание СМО для последующего статистического моделирования, а также для других, не связанных с моделированием, исследований СМО.

По сравнению с предшествующими описаниями [6,7] сделан ряд усовершенствований для достижения большей универсальности и логической завершенности системы. Некоторые понятия и термины позаимствованы из языка Симула [2] .

2. СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ (СМО) И ЕЕ МОДЕЛИ

Понятие СМО не поддается точному определению ввиду очень большого разнообразия объектов, к которым применим развиваемый в настоящей работе метод описания. Обычно СМО - совокупность элементов (приборов, линий) для удовлетворения (обслуживания) поступающих требований (вызовов). В более широком смысле СМО - некоторое реально работающее, проектируемое или воображаемое устройство, имеющее определенную структуру (строение) и определенный алгоритм функционирования.

Работа СМО в большинстве случаев содержит элементы случайности, поэтому алгоритм функционирования понимается в вероятностном смысле - кроме строго определенных действий допускаются такие, как, например, реализация случайной величины с заданной функцией распределения.

Исследование СМО обычно проводится с целью получить информацию о некоторых свойствах (характеристиках) СМО. Для этого строятся модели СМО - абстрактные системы, сохраняющие в некотором смысле (обычно статистическом) исследуемые свойства СМО. Модели в зависимости от назначения имеют различные степени абстракции (формализации).

При исследовании СМО методом статистического моделирования составляется программа моделирования - модель СМО на алгоритмическом языке, предназначенная для запуска в ЭВМ. Вероятностный характер СМО отображается введением в программу моделирования источников случайных чисел. Результатом моделирования является статистическая информация - оценки некоторых вероятностных характеристик моделируемой СМО.

Программа моделирования имеет наибольшую степень формализации среди моделей СМО. При работе по составлению программы моделирования целесообразно ввести промежуточные модели - функциональную модель и математическую модель СМО, имеющие также и самостоятельный интерес. На практике не всегда эти стадии четко разграничены, возможны модели, имеющие черты одновременно нескольких основных моделей.

2.I. Функциональная модель

Составление функциональной модели - первая степень абстракции, при которой отбрасываются конкретные и несущественные свойства реальной СМО и составляется описание СМО с применением некоторой общей терминологии. Будем придерживаться "телефонной" терминологии - поступающие требования всегда называются вызовами, а обслуживающие устройства - линиями. Например, если рассматриваемая СМО - магазин, то вызовами называются покупатели, а линиями - продавцы.

Рассматриваемые в теории телетрафика основные виды СМО, такие как полнодоступный пучок линий с потерями или

однолинейная система с повторными вызовами - это функциональные модели , каждая из которых охватывает целый класс реальных СМО.

2.2. Математическая модель

Математическая модель отличается от функциональной тем, что для описания СМО применяются лишь чисто математические понятия - числа, переменные, функции, алгоритмы. В математической модели состояние СМО изображается набором значений определенных переменных, а правила функционирования СМО задаются в виде алгоритма, изменяющего эти значения. Вероятностный характер работы СМО изображается введением случайных величин с заданными функциями распределения.

Работа алгоритма дает реализацию некоторого случайного процесса. При соблюдении некоторых условий (экспоненциальные распределения случайных величин) имеем реализацию однородного марковского процесса. В этом случае можно пользоваться моделированием цепи Маркова, что упрощает алгоритм моделирования. Математическая модель, составленная с соблюдением всех формальных правил на некотором алгоритмическом языке, переходит в программу моделирования.

В следующих разделах будут рассмотрены основные понятия, позволяющие формализовать продвижение по цепочке СМО - функциональная модель - математическая модель - программа моделирования.

3. СТРУКТУРНАЯ И ДИНАМИЧЕСКАЯ ЧАСТЬ СМО

При исследовании почти всех СМО можно выделить две части, взаимодействие которых составляет работу СМО. Первая из них - структурная часть - включает те элементы СМО, которые имеются в неизменном количестве и меняют лишь внутренние состояния. К структурной части обычно относятся

составляющие СМО обслуживающие устройства. Вторая - динамическая часть - состоит из объектов, которые возникают (поступают) и взаимодействуют некоторое время со структурной частью, после чего покидают СМО. Такими объектами, как правило, являются поступающие на СМО требования, поэтому элементы динамической части будем называть вызовами.

Будем предполагать, что алгоритм работы СМО связан с динамической частью - вызовы рассматриваются как активные элементы, совершающие действия над пассивной структурной частью.

4. СОБЫТИЯ И ДЕЙСТВИЯ

В настоящем разделе будут изложены понятия, составляющие основу функциональной модели СМО. Если реальная СМО по каким - либо причинам не описывается этими понятиями, то причины несоответствия следует устранить на стадии составления функциональной модели. В большинстве случаев это удается без значительных изменений свойств СМО.

В функциональной модели предполагается дискретность работы СМО (здесь и далее СМО означает не только исходную СМО, но и ее модели) в следующем смысле. Время функционирования СМО разбивается на чередующиеся интервалы активного и пассивного времени. Активное время по сравнению с пассивным имеет пренебрежительно малую длительность, которая в модели приравнивается нулю. Все операции по воспроизведению работы СМО сосредоточены в активное время и, следовательно, совершаются мгновенно. Совокупность этих операций за один интервал активного времени называется событием.

События состоят из элементов, называемых активными действиями. Каждое активное действие связано с одним и только одним вызовом, который совершает это действие. Кроме активных действий, вызовы совершают пассивные дейс-

твия. Пассивным действием называется состояние вызова, имеющее длительность из целого числа интервалов пассивного времени, когда находящийся в СМО вызов не совершает активных действий.

Каждый вызов имеет собственный алгоритм, указывающий, какие действия и в каком порядке вызов должен совершать во время своего пребывания в СМО. Вызов покидает СМО, когда его алгоритм выполнен до конца.

Вызов называется активным, если он совершает активное действие. В каждый момент активного времени активен один и только один вызов. Вызов остается активным, пока он либо переходит в пассивное действие, либо покидает СМО, либо активизирует (т.е. переводит в активное состояние) другой вызов.

Работа всей СМО в целом состоит из переплетения активных действий проходящих через СМО вызовов. Порядок активизации вызовов происходит по определенному алгоритму, который вытекает из совокупности алгоритмов вызовов и называется алгоритмом выбора. Строение алгоритма выбора будет рассмотрено в I2.3.

5. СХЕМА ДЕЙСТВИЙ

Графическое изображение последовательности действий, совершаемых одним, но произвольным вызовом от поступления до выхода из системы, называется схемой действий. Действия вызовов изображаются клетками, а порядок их выполнения с точки зрения вызова - стрелками. Схема действий дает наглядное представление о поведении вызова, полностью отображает работу СМО и является основной частью как функциональной, так и математической модели. Для функциональной модели в клетках схемы действий пишутся словесные описания действий, а для математической модели - запись действий на языке математических обозначений, в частности, на алгоритмическом языке А4 [8].

Работу СМО можно наглядно представить в виде игры типа "кто первый ?". Игровым полем служит схема действий, фишками - вызовы, игроком является алгоритм выбора, один ход соответствует одному событию. В промежутках между ходами фишки, которые введены в игру, стоят на клетках, изображающих пассивные действия. В каждом ходу либо активизируется одна из имеющихся фишек, либо вводится новая, и совершаются ее продвижения по клеткам активных действий, выполняя при этом взаимодействия со структурной частью. Ход кончается либо на клетке пассивного действия, либо выходом фишки из игры. Правила изображения и примеры схем действий будут даны в следующих разделах.

6. ТИПЫ ДЕЙСТВИЙ

Введем классификацию действий, совершаемых вызовов. Каждый тип действий имеет свое изображение в схеме действий. Изображения действий показаны на рис.I. Активные действия изображаются в виде прямолинейных, а пассивные - в виде круглых клеток. Все действия, кроме генерации и запуска, могут иметь несколько входов.

6.I. Акция

Акция - активное действие вызова, в котором нет взаимодействия с другими вызовами и после которого однозначно определено следующее действие вызова. Акция изображается прямоугольником, внутри которого помещается описание совершаемого действия, с входом сверху и одним выходом снизу. К акциям обычно относятся все взаимодействия вызова со структурной частью, например, занятие и освобождение линий.

6.2. АЛЬТЕРНАТИВА

Альтернатива - активное действие, реализующее выбор (разветвление) дальнейшего пути вызова из двух возможных.

Альтернатива изображается параллелограммом, внутри которого пишется либо высказывание (условие), принимающее значение "истинно" и "ложно", либо число p ($0 \le p \le 1$). Имеется вход сверху и два выхода (снизу и в одной из боковых сторон клетки). Выполнение альтернативы заключается в проверке условия - если оно истинно, то вызов уходит по боковому выходу, а если ложно, то по нижнему. В случае числа p вызов с вероятностью p уходит по боковому выходу, а с вероятностью $(1-p)$ - по нижнему.

6.3. Генерация

Генерация - активное действие, в результате которого возникает новый вызов, т.е. из одного получаются два вызова. Генерация изображается в виде треугольника с вертикальным основанием, имеющего один вход сверху и два выхода - снизу (для исходного вызова) и из боковой вершины (для нового вызова). После генерации активизируется новый вызов, а исходный вызов приостанавливает свои действия. После окончания активных действий нового вызова вновь активизируется исходный вызов.

6.4. Запуск

Запуск - активное действие, в результате которого возникает один "первоначальный" вызов. Запуск изображается горизонтальной чертой с одним выходом вниз. Предполагается, что запуск производится всего один раз в начале работы СМО, а все остальные вызовы возникают внутри СМО в результате действий типа "генерация". Такая концепция отличается от общепринятого представления о потоке вызовов, поступающих на СМО извне, однако по ряду соображений целесообразно рассматривать источник (или источники) вызовов как составную часть самой СМО. Строение источника вызовов будет показано в п.6.9.

6.5. Закрытие

Закрытие - активное действие вызова, в результате которого вызов прекращает свое существование (покидает систему). Закрытие изображается горизонтальной черточкой с входом сверху. В отличие от запуска закрытий может быть несколько.

6.6. Активизация

Активизация или просмотр очереди - активное действие вызова (A), в результате которого активизируется другой вызов (B), находившийся до этого в состоянии пассивного действия типа "ожидание" (см.6.8). Действия вызова A при этом приостанавливаются до окончания активных действий вызова B.

Активизация изображается прямоугольником, который отличается от изображения акции лишь тем, что в одной из боковых сторон имеется прерывистая черта, соединяющая активизацию с полем, изображающим ожидание вызова B.

Действие активизации заключает в себе такие алгоритм проверки, устанавливающий, есть ли на указанном поле ожидающие вызовы, подлежащие активизации, и алгоритм для выбора вызова B из совокупности (очереди) ожидающих вызовов. Если проверка даст отрицательный результат, то активизация ничего не меняет. Совокупность упомянутых алгоритмов - алгоритм активизации - указывается отдельно в каждом конкретном случае.

6.7. Задержка

Задержка - пассивное действие вызова, длительность которого является случайной величиной с заданной функцией распределения (не исключается возможность детерминированной длительности). Активизация вызова наступает после исте-

чения времени задержки и не зависит от других вызовов.

Задержка изображается в виде окружности с входом сверху и одним выходом снизу. Внутри окружности можно указать информацию, характеризующую длительность задержки. К действиям типа "задержка" обычно относятся обслуживание вызова и промежутки между вызовами поступающего потока.

6.8. Ожидание

Ожидание - пассивное действие вызова, не имеющее заранее устанавливаемой длительности. Ожидание прекращается в результате действия типа "активизация", совершаемого другим вызовом.

Ожидание изображается окружностью с входом сверху, одним выходом снизу и примыкающей с боковой стороны прерывистой чертой (таких может быть несколько), которая соединяет ожидание с соответствующей клеткой действия типа "активизация". Внутри окружности можно указать информацию о строении совокупности ожидающих вызовов - очереди

В следующих разделах рассматриваются некоторые производные действия, которые можно образовать, комбинируя основные типы действий.

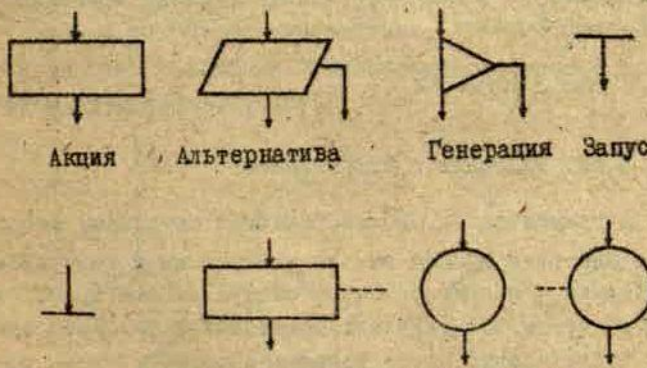


Рис.I. Типы действий

6.9. Источник

Источником называется элемент схемы действий, изображенный на рис. 2 слева. В источнике объединены два действия - задержка и генерация, он имеет один вход и один выход. После начального запуска через вход источник постоянно выпускает поток вызовов, причем расстояния между вызовами соответствуют длительности задержки. Ввиду того, что источник является составной частью почти всех схем действий, для него введено специальное изображение (рис. 2 справа) - полукруг с входом сверху и выходом снизу. Внутри полукруга можно поместить информацию о характере потока. Вход источника, как правило, соединяется с выходом запуска. Если схема действий имеет несколько источников, то они подключаются к запуску так, как показано на рис.3. В схеме действий допускается сокращенное изображение источника (источников) без изображения запуска (например, рис.8).

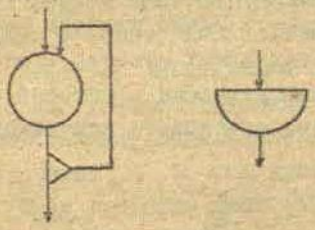


Рис.2. Источник

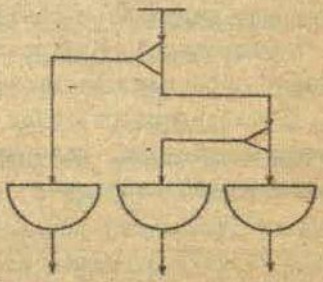


Рис.3. Запуск трех источников

6.10. Задержка с прерыванием

В практических СМО встречаются ситуации, когда пассивное действие вызова нельзя отнести ни к задержкам, ни к ожиданиям - например, вызов обслуживается прибором, который во время обслуживания может выйти из строя или во время обслуживания может появиться другой вызов, который имеет приоритет и "выгоняет" первый вызов. В этих случаях

необходим элемент схемы действия, изображающий задержку с прерыванием - пассивное действие, которое оканчивается по истечении заданного времени, но может и кончиться раньше вследствие действия другого вызова. Такой элемент можно построить из основных типов действий так, как показано на рис.3 слева. Вызов посредством генерации расщепляется на два вызова - первый (основной) вызов поступает на ожидание, а второй (вспомогательный) совершает задержку. После окончания задержки вспомогательный вызов совершает активизацию по следующему алгоритму: проверяется, находится ли на ожидании соответствующий основной вызов, и если да, то он активизируется. Действия вспомогательного вызова этим кончаются и он ликвидируется. Активизацию ожидающего основного вызова может произвести и другой, посторонний вызов, на что указывает черта слева; это соответствует прерыванию задержки. После активизации основной вызов проходит альтернативу с условием: активизацию совершил вспомогательный вызов. Построенный таким образом элемент имеет два выхода - левый соответствует случаю прерывания, правый - окончанию задержки. Сокращенное условное обозначение задержки с прерыванием изображено на рис.4 справа. Левый кружок символизирует ожидание, правый - задержку. Правую и левую стороны можно менять местами, т.е. пользоваться зеркальным изображением рисунка. Пример задержки с прерыванием показывает, как с помощью искусственных приемов преодолеваются затруднения, вызванные ограниченностью набора основных типов действий.

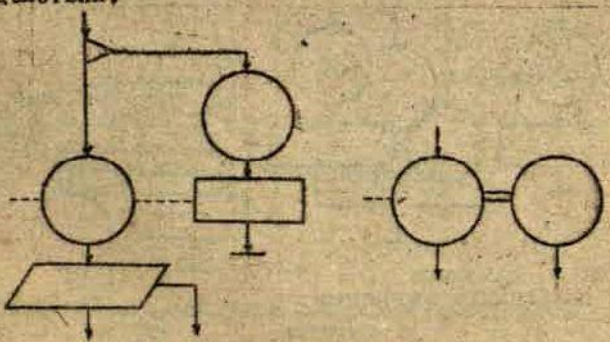


Рис.4. Задержка с прерыванием

6.II. Примеры сложных действий

Рассмотрим некоторые примеры построения сложных действий, которые встречаются в СМО.

В результате действий вызова в СМО может возникать необходимость получить n вызовов ($n=0,1,2,\dots$). Такие задачи возникают, например, при распаде и рождении новых частиц, при распространении эпидемий. Генерация n вызовов представлена на рис.5.

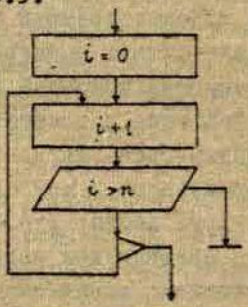


Рис.5. Генерация n вызовов

В качестве другого примера рассмотрим задержку вызова, которая реализует наименьшее или наибольшее значение из задержек $F_1(t)$, $F_2(t)$. Задержка с наименьшим значением представлена на рис.6, а с наибольшим значением на рис. 7. Задержка с наименьшим и наибольшим значениями легко обобщается на случай n задержек $F_1(t), F_2(t), \dots, F_n(t)$.

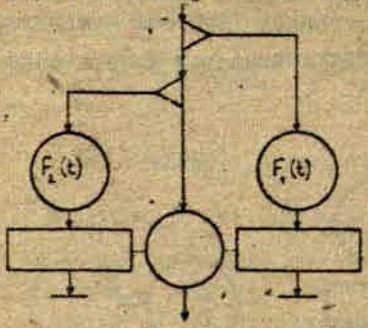


Рис.6. Задержка с наименьшим значением

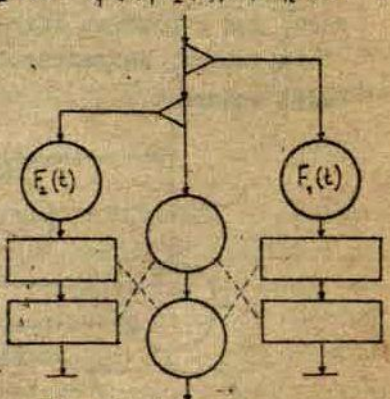


Рис.7. Задержка с наибольшим значением

7. ПРИМЕРЫ ФУНКЦИОНАЛЬНЫХ МОДЕЛЕЙ

Функциональная модель СМО состоит из структурной и динамической части. Структурная часть задается в виде словесного описания и на уровне функциональной модели не формализуется. Динамическая часть представляется в виде схемы действий вместе с необходимыми пояснениями к ней. Рассмотрим примеры функциональных моделей СМО.

7.1. Полнодоступный пучок линий с потерями

Рассматриваемая СМО состоит из V обслуживающих устройств (линий). На СМО поступает поток вызовов с заданной функцией распределения расстояний между вызовами. Если в момент поступления вызова не все линии заняты, то вызов занимает одну из свободных линий и обслуживается случайное время с заданной функцией распределения, после чего линия освобождается и вызов покидает СМО. Если при поступлении вызова все линии заняты, то вызов сразу покидает систему (теряется).

Структурная часть модели состоит из V линий, каждая из которых имеет два состояния - свободно и занято. Динамическая часть в виде схемы действий показана на рис. 8. Так как функции распределения для поступающего потока и времени обслуживания не уточняются, то соответствующие клетки на схеме действий пусты. Надписи вне клеток не входят в схему действий, а являются пояснениями к ней.

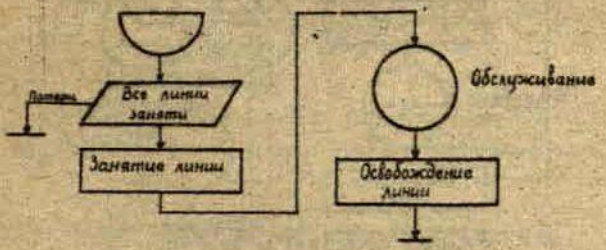


Рис.8. Схема действий полнодоступного пучка с потерями

7.2. Полнодоступный пучок с ограниченным числом мест ожидания

Имеется V линий и n мест для ожидания. Поступающий вызов занимает линию и обслуживается, а если свободной линии нет, то вызов занимает место для ожидания. Если все места для ожидания заняты, то вызов теряется. После обслуживания вызов освобождает линию и покидает СМО; если в этот момент имеются занятые места для ожидания, то один из вызовов переводится из ожидания на обслуживание.

Структурная часть - V линий и n мест для ожидания; каждое из них может быть свободно или занято. Динамическая часть - схема действий на рис. 9. На схеме действий не указан алгоритм активизации, который заключается в следующем: проверяется, есть ли вызов в очереди, и если есть, то активизируется один из них. Порядок становления в очередь и выхода из очереди в рассматриваемой модели не уточняются. Этот пример показывает, какие трудности возникают при словесном описании даже очень простых моделей СМО - описание получается либо громоздким, либо неточным (в настоящем примере описание лишено точности). Построение схемы действий делает описание точным, наглядным.

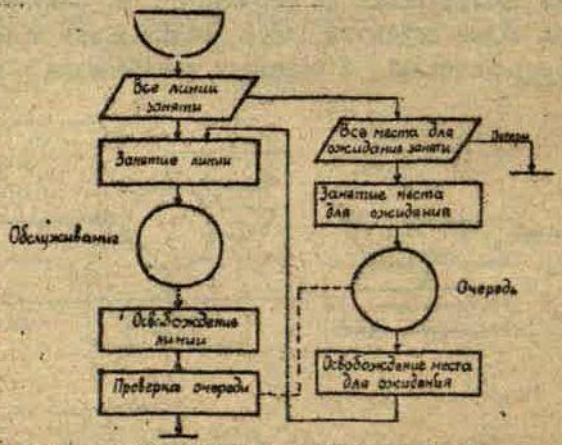


Рис.9. Схема действий для полнодоступного пучка с ограниченным числом мест для ожидания

7.3. Однолинейная система с неисправностями

СМО состоит из одной линии, которая может быть в одном из трех состояний - свободном, занятом и неисправном; в первых двух состояниях линия считается исправной. Периоды исправности и неисправности линии чередуются и имеют случайные длительности, не зависящие от поступающих вызовов. На линию поступает поток вызовов. Если линия свободна, то вызов ее занимает и обслуживается, в противном случае вызов теряется. Если период неисправности наступает при занятой линии, то обслуживание прерывается и вызов теряется.

Структурная часть модели содержит одну линию, имеющую три состояния. Динамическая часть представлена в виде схемы действий на рис.10. Пример показывает применение производных элементов схемы действий - двойного запуска и задержки с прерываниями.

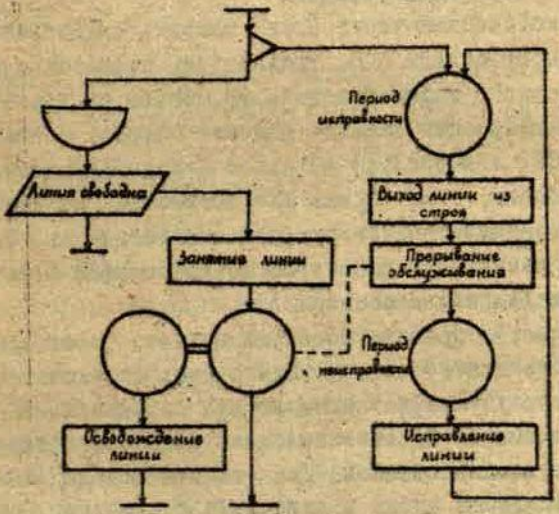


Рис.10. Схема действий однолинейной системы с неисправностями

8. СТАТИСТИЧЕСКАЯ ЧАСТЬ

Конечная цель моделирования СМО - получение информации, обычно статистической, о моделируемой СМО, поэтому в моделях СМО, кроме работы самой СМО, необходимо отображать получение этой информации. Модель СМО, в которой включен сбор статистической информации, называется расширенной моделью СМО. Программа моделирования всегда является расширенной моделью, так как запуск на ЭВМ программы, не выдающей никакой информации, лишен смысла.

Расширенная модель СМО, кроме структурной и динамической частей, имеет еще третью - статистическую часть. Статистическая часть представляет собой совокупность элементов (переменных), способных принимать численные значения. Изменение этих значений включается в динамическую часть модели. В конце моделирования из значений статистической части вычисляется информация, выдаваемая в качестве результатов моделирования.

Статистическая часть может оказать существенное влияние на выбор модели СМО. Если модель строится с целью статистического моделирования, то именно от характера собираемой информации зависит решение вопроса о том, какие свойства СМО и насколько детально отразить в модели. С другой стороны, если модель уже построена, то статистическая часть имеет некоторую самостоятельность - она не оказывает влияния на работу остальной модели и ее в некоторых пределах можно менять.

В качестве примера расширим модель, рассмотренную в 7.I. Пусть требуется определить одну статистическую характеристику - вероятность потерь по вызовам π, которая оценивается отношением числа потерянных вызовов к числу поступивших вызовов. Для этого к модели добавляется статистическая часть в виде двух переменных (счетчиков): a - счетчик потерянных вызовов, b - счетчик поступивших вызовов. Предполагается, что в начале моделирования $a = b = 0$. Структурная часть остается без изменений,

а динамическая часть расширяется добавлением к схеме действий двух акций по подсчету вызовов (рис.II). В конце моделирования вычисляется оценка вероятности потерь по формуле $\Pi = a/b$.

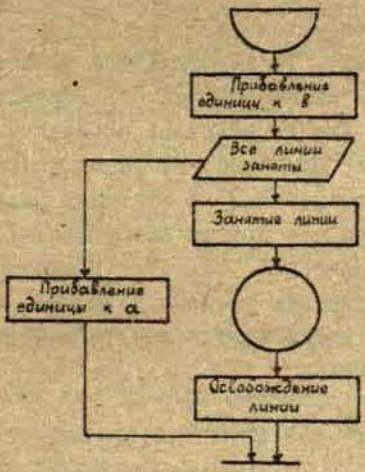


Рис.II. Схема действий расширенной модели полнодоступного пучка с потерями

9. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Для построения математической модели будем пользоваться элементами языка А4 [8] . Структурная часть СМО переходит в совокупность массивов, содержащих переменные, значения которых отображают состояние структурной части СМО. То же относится к статистической части в случае расширенной модели. Структурная и статистическая части математической модели задаются в виде таблиц массивов (ТМ), правила составления которой такие же, как в описании языка А4 [8] . Динамическая часть математической модели представляет собой алгоритм, изменяющий значения переменных структурной и статистической частей. Динамическая часть задается в виде схемы действий, причем действия описываются средствами языка А4.Схема действий мате-

матической модели строится почти так же, как блок-схема А4 - программы. Акции представляют собой последовательности вычислительных команд, альтернативы изображаются в виде усеченных команд сравнения. В клетках типа "задержка" пишется число или обозначение переменной, значение которой равно параметру задержки $\lambda = 1/\tau$ (τ - средняя длительность задержки). То же относится к клеткам типа "источник", где λ - интенсивность создаваемого источниками потока вызовов. Клетки действий остальных типов (генерация, активизация, ожидание) не заполняются. Информация об алгоритмах активизации, о функциях распределения длительности задержек и о других особенностях модели, не включенных в схему действий, задается в словесном (неформальном) виде в пояснениях к схеме действий. Формализация этой информации производится на этапе составления программы моделирования.

IO. ДИНАМИЧЕСКАЯ ИНФОРМАЦИЯ

В математических моделях СМО, за исключением самых простых случаев, возникает необходимость ввести переменные значения, которые связаны непосредственно с вызовами. Эти переменные составляют часть модели, называемую динамической информацией.

С каждым вызовом связано определенное фиксированное количество переменных динамической информации, а общее количество этих переменных пропорционально числу находящихся в СМО вызовов. Динамическая информация имеет активную и пассивную часть. Активная часть содержит столько переменных, сколько информации связано с одним вызовом; в ней всегда находится информация о том вызове, который активен. Пассивная часть содержит информацию о всех вызовах, которые в данный момент совершают пассивные действия. Перевод информации из активной части в пассивную и обратно производится алгоритмом выбора при смене активного вызова.

При составлении моделей следует учитывать, что для пользования доступна лишь активная часть динамической информации - активный вызов может пользоваться собственной информацией, но ему недоступна информация о других вызовах (исключение составляет действие активизации, когда активный вызов может получить информацию об ожидающих вызовах проверяемой очереди).

Активная часть динамической информации включается в таблицу массивов. Пассивная часть в математической модели прямо не указывается, но ее существование подразумевается. Пример модели с динамической информацией приведен в II.2.

II. ПРИМЕРЫ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Рассмотрим два примера математических моделей одной и той же СМО - рассмотренного в 7.I. полнодоступного пучка с потерями. Различаются модели только статистической частью - для каждого примера требуется своя статистическая информация. Эти примеры показывают, как требуемая информация влияет на выбор модели СМО.

II.I. Первая модель. Пусть имеется полнодоступный пучок из ν линий, на который поступает поток вызовов с интенсивностью λ. Средняя длительность обслуживания равна I. Требуется определить (точнее: получить статистические оценки) вероятности p_i ($i=0,1,\ldots,\nu$) того, что в момент поступления вызова занято i линий, т.е. распределение числа занятых линий. Отметим, что при этом получается и вероятность потерь $\pi = p_\nu$.

Для поставленной цели несущественно, какие именно линии заняты, нужно лишь общее количество занятых линий, поэтому состояние модели можно описать посредством одной переменной J, принимающей значение $0,1,\ldots,\nu$. Динамическая информация в этой модели не требуется.

Структурная и статистическая часть модели задается следующей ТМ:

ТМ для схемы рис.I2.

Идентификатор	Тип	Границы от	Границы до	Назначение	
V	J	-	-	число линий V	структурная часть
L	A	-	-	интенсивность потока	
J	J	-	-	число занятых линий	
W	A	-	-	счетчик поступивших вызовов	Статистическая часть
R	A	0	V	счетчики случаев поступления вызова в состоянии i, где i - число занятых линий	

Динамическая часть модели в виде схемы действий приведена на рис.I2.

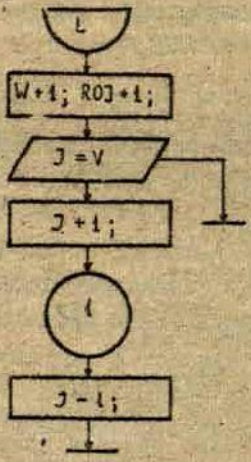


Рис.I2. Схема действий первой модели

Предполагается, что начальные значения переменных W, R_V, J равны нулю, а значения L и V задаются в качестве исходных данных. После окончания работы модели значения переменных массива R следует разделить на число поступивших вызовов, т.е. выполнить команду R_V/W. После этого значения R дают требуемую информацию - оценки вероятностей p_i.

II.2. Вторая модель

Рассматривается та же СМО, что в II.I., но требуется оценить вероятности q_i ($i=1,2,\ldots,V$) того, что в момент поступления вызова занята i-ая линия. Выбор занимаемой вызовом линии производится в порядке возрастания номеров (упорядоченный поиск), т.е. занимается та из свободных линий, которая имеет наименьший номер.

В этой модели уже не безразлично, какие именно линии заняты, поэтому для каждой линии вводится переменная, принимающая два значения 0 - свободно, I - занято. Появляется также потребность в динамической информации - вызов, находящийся на обслуживании, должен "запомнить" номер занимаемой линии, чтобы после обслуживания освободить именно эту линию. Активная часть динамической информации представлена переменной J.

К динамической части добавляются новые действия, связанные с поиском свободной линии. Модель задается в виде ТИ и схемы действий на рис.I3. После окончания моделирования значения переменных R следует разделить на значение W.

ТМ для схемы рис.I3.

Идентификатор	Тип	Границы от	Границы до	Назначение	
V	J	-	-	число линий	Структурная часть
L	A	-	-	интенсивность потока	Структурная часть
B	A	1	V	занятость линий (0 - свободно, I - занято)	Структурная часть
W	A	-	-	число поступивших вызовов	Статистическая часть
R	A	1	V	счетчики случаев занятости i - ой линии	Статистическая часть
J	J	-	-	номер занимаемой линии	Динамическая часть

Примечание: массив B имеет тип A для удобства сбора статистической информации.

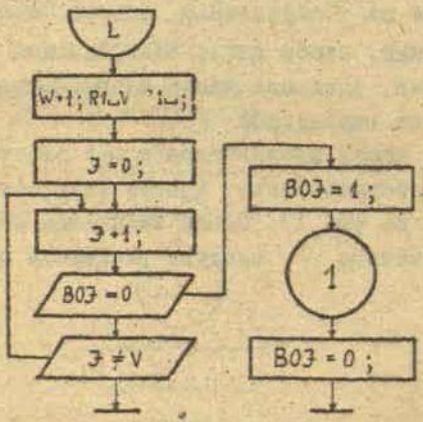


Рис.I3. Схема действий второй модели.

I2. ОСНОВЫ СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Математическая модель является средством описания работы СМО, но она не дает полного представления в статистическом моделировании. Во-первых, в модели не предусмотрено описание "начала" и "конца" моделирования, т.е. нет алгоритма для построения исходного состояния модели и алгоритма, обрабатывающего собранную в ходе работы модели информации. Во-вторых, схема действий дает алгоритм работы отдельного вызова, но не дает представления о функционировании всей модели в целом - не хватает алгоритма выбора. Часть этих пробелов восполняется при переходе от математической модели к программе моделирования; другая часть - применением универсальной программы моделирования, которая содержит в готовом виде те элементы статистического моделирования, которые являются общими для всех (или для многих) моделей СМО.

В настоящем разделе рассматриваются основные принципы построения законченого алгоритма статистического моделирования СМО - программы моделирования. Будем предполагать стационарность рассматриваемых моделей, т.е. независимость исследуемых характеристик СМО от времени.

I2.I. Общая схема моделирования

В целях получения более полной и надежной статистической информации предлагается следующая схема общего строения алгоритма моделирования. Во-первых, в начале работы модели производится приведение в стационарный режим - отрезок работы модели без сбора информации. Во-вторых, основное моделирование разделяется на равные по числу сделанных вызовов промежутки - серии. После каждой серии производится обработка и выдача результатов, в которых отражена работа модели за текущую и все предыдущие серии. Эта схема на языке А4 выражается в виде следующего блока:

```
PM.  *SA; *SB; W=0; W2=0;
     W1=Z; *MD; *SB; W=0; W1=0;   (I)
  1. W1+Z1; W2+1; *MD; *RZ;
     W2<Z2*1;
```

В записи блока применены следующие объекты.

Блоки:

SA - начальный алгоритм структурной части;
SB - начальный алгоритм статистической части;
MD - алгоритм собственного моделирования;
RZ - алгоритм обработки и выдачи результатов.

Переменные (все типа A):

W - число сделанных вызовов;
W1 - задаваемое для блока MD требуемое число вызова (при W>W1 следует выход из MD);
W2 - число сделанных серий;
Z - число вызовов для приведения в стационарный режим;
Z1 - число вызовов в серии;
Z2 - требуемое число серий.

Блок PM является стандартным (универсальным) основным блоком для всех программ моделирования. Теперь составление программ моделирования сводится к написанию блоков SA, SB, MD, RZ.

I2.2. Начальные алгоритмы

Блоки SA и SB предназначены для построения начального состояния структурной и статистической частей модели соответственно. В блоке SA, кроме засылки начальных значений переменных структурной части, программируются все подготовительные действия, предшествующие моделированию.

В блок SB засылаются начальные значения (как правило - нули) переменных статистической части. В (I) видно, что блок SB работает дважды - до и после приведения

в стационарный режим. Таким образом, приведение в стационарный режим достигается не отключением сбора информации (что трудно осуществимо), но стиранием накопленной информации).

В качестве примеров составим блоки SA и SB для моделей, рассмотренных в II.I и II.2.

Для первой модели:

SA.J=0; SB.R␣V=0; (2)

Для второй модели:

SA.B1␣V=0; SB.R1␣V=0; (3)

Начальное значение переменной W задается в блоке PM, поэтому в блоке SB это не делается.

I2.3. Блок моделирования и алгоритм выбора

Блок MD производит собственно моделирование - имитацию работы СМО в стационарном режиме. В MD объединены две части - алгоритм действий, который является отображением схемы действий, и алгоритм выбора, определяющий порядок активизации вызовов.

В блоке MD ведется подсчет вызовов - изменение переменной W. Выход из блока MD следует при достижении заданного числа вызовов W1, т.е. при выполнении неравенства W≥W1. Не обязательно через W обозначать число всех сделанных вызовов - W означает количество тех вызовов, по которым определяется длительность моделирования.

Алгоритм выбора зависит от выбранного способа моделирования. Имеются два основных способа - моделирование процессом и моделирование цепью.

Моделирование процессом является более универсальным способом и не накладывает никаких ограничений на моделируемые случайные величины длительностей задержек. Алгоритм выбора при этом работает следующим образом. Имеется некоторая переменная T, изображающая текущее время. Когда вызов приступает к действию типа "задержка", вычисляется время окончания задержки путем прибавления к T очередной

реализации случайной величины длительности задержки. Эти времена для всех вызовов, находящихся в состоянии задержки, располагаются в неубывающем порядке и алгоритм выбора на каждом шагу выбирает наименьшее из них. Выбранное время становится текущим временем, а соответствующий вызов активизируется.

Второй способ - моделирование цепью - применим лишь в том случае, если все длительности задержек имеют экспоненциальную функцию распределения.

$$F(t) = 1 - e^{-\lambda t},$$

где $F(t)$ - вероятность длительности меньше t, λ - параметр, обратный по величине средней длительности задержки. В этом случае вероятность окончания задержки в некоторый малый промежуток времени Δt равна $\lambda \Delta t + 0(\Delta t)$ и не зависит от уже прошедшей длительности задержки. Кроме того, требуется чтобы все вызовы, находящиеся на одной клетке типа "задержка" схемы действий, имели одно и то же λ. Пусть схема действий имеет N клеток типа "задержка", $\lambda_1, \lambda_2, \ldots, \lambda_N$ -параметры этих задержек, а $n_1, n_2, \ldots, n_N$ - число вызовов на каждой из этих клеток. Тогда алгоритм выбора с вероятностью

$$p_i = \frac{\lambda_i n_i}{\sum_{j=1}^{N} \lambda_j n_j} \quad (i = 1, 2, \ldots, N)$$

выбирает i - ую клетку, а в ней - равновероятно один из n_i вызовов. Выбранный вызов активизируется.

Второй способ моделирования в случае выполнения указанных требований предпочтительнее - алгоритм моделирования в этом случае проще, работает быстрее и требует меньше места для размещения информации по сравнению с алгоритмом моделирования процессом.

Кроме действий по активизации вызовов, алгоритм выбора управляет динамической информацией - переводит из активной части в пассивную и обратно, организует размещение информации в пассивной части.

12.4. Примеры блоков моделирования

Построим блоки MD для моделей, рассмотренных в II.I и II.2. Для первой модели предполагается моделирование цепью, для второй - моделирование процессом.

Пусть Q - идентификатор блока, реализующего построение случайной величины, равномерно распределенной в интервале (0,I), и одновременно обозначение переменной, в которой эта величина получается. Тогда блок MD имеет вид

```
MD. W>W1*; *Q; Q*(JA(J)+L)>L*2;                    (4)
I.  W+1; ROJ+1; J=V*0; J+1; *0;   2. J-1; *0;
```

Здесь алгоритм действий занимает часть блока начиная с метки I, остальная часть блока - алгоритм выбора, реализующий поступление вызова с вероятностью $\lambda/(\lambda+i)$ и окончание обслуживания с вероятностью $i/(\lambda+i)$, где λ, i - значения переменных L, J.

Построить алгоритм моделирования процессом для второй модели значительно сложнее. Здесь необходимо введение ряда новых массивов, которые перечислены в следующей дополнительной ТМ.

Идентификатор	Тип	Границы от	Границы до	Назначение
T	A	-	-	текущее время
U	A	-	-	время следующего вызова
J	J	-	-	число обслуживаемых вызовов
K	J	-	-	вспомогательная переменная
S	A	0	V	времена окончания обслуживаний
M	J	1	V	пассивная часть динамической информации (хранение значений J)
E	A	-	-	очередная реализация промежутка между вызовами
F	A	-	-	очередная реализация длительности обслуживания

Кроме того, вводятся два блока с идентификаторами E и F, реализующие построение значений случайных величин E и F. Теперь блок MD имеет вид:

```
MD. W≥W1*; U≤SOJ*1; J=MOJ; J-1; *8;
 1. T=U; *E; U+E; *5;
 2. *F; T+F; J+1; K=J;
 3. K-1; T<SOK*4; S1K=SOK; M1K=MOK; *3;
 4. S1K=T; M1K=J; *0;                          (5)
 5. W+1; R1␣V+B1␣; J=0;
 6. J+1; BOJ=0*7; J≠V*6; *0;
 7. BOJ=1; *2;
 8. BOJ=0; *0;
```

Здесь алгоритм действий начинается с метки 5, остальная часть блока - алгоритм выбора. Работа алгоритма выбора заключается в упорядочении времени окончания обслуживаний в массиве S (параллельно с S перемещается значение переменных массива M) и в сравнении наименьшего из этих времен поступления следующего вызова. Для правильной работы алгоритма необходимо, чтобы переменная с индексом 0 массива S имела значение, превышающее любое возможное значение времени окончания обслуживаний. Практически можно пользоваться, например, значением 10^{10}.

В блоке SA необходимо предусмотреть засылку начальных значений для J, S и U для чего составляется дополнительный блок SA.

```
SA. J=0; S=1 ▲ 11; U=0;                        (6)
```

12.5. Блок обработки результатов

Блок R2 работает после окончания каждой серии моделирования. Работа блока заключается в построении из значений статистической части модели той информации, которая предусмотрена в качестве результатов моделирования. При этом следует учитывать, что при работе R2 моделирование, в общем случае, лишь приостановлено, но не окончено, поэтому изменение статистической части модели

не допускается (напр., выполнение для модели из II.I команды R␣V/W ; в блоке RZ была бы ошибкой). Блок RZ выполняет также вывод (печать) результатов моделирования.

В качестве примеров напишем блоки RZ для рассматриваемых нами моделей из II.I и II.2. В обоих случаях необходимо введение нового массива P для результатов (тип A , индексы от 0 до V - для первой модели и от 1 до V для второй модели).

Для первой модели:

RZ. P␣V=R␣/W; *0*P␣V; (7)

Для второй модели:

RZ. P1␣V=R1␣/W; *0*P1␣V; (8)

Теперь построение программы моделирования для наших примеров, в основном, закончено. Программа для первой модели получается соединением в одну программу блоков (I), (2), (4) и (7); для второй модели - (I), (3), (5), (6) и (8). Для окончания программы остается добавить для первой модели блок G , для второй - блок E и F .

I2.6. Универсальная программа моделирования

В рассмотренных примерах можно проследить, что программа моделирования заключает в себе две части, которые назовем конкретной частью и универсальной частью.

Конкретная часть является описанием на алгоритмическом языке свойств моделируемой СМО. К конкретной части относятся блоки SA , SB , RZ и часть блока MD, а именно - алгоритм действий.

Универсальная часть описывает процесс статистического моделирования вообще, вне зависимости от моделируемой СМО. К универсальной части относится блок PM и часть блока MD - алгоритм выбора.

Универсальную часть можно формализовать до такой степени, что она полностью освобождается от объектов, связанных с конкретными моделями (в наших примерах это не сдела-

но до конца). В итоге получается универсальная программа моделирования (УПМ), которую можно использовать как готовую универсальную часть любой программы моделирования.

Кроме блока РМ и алгоритма выбора, в УПМ включается ряд стандартных программ для операций, обычно применяемых в программах моделирования - получение случайных величин с различными функциями распределения, статистическая обработка результатов по сериям и др.

В УПМ предусмотрены также средства для реализации отсутствующих в рассмотренных примерах действий типа "ожидание", "активизация" и "генерация".

Пример блока (5) показывает, что универсальная часть (алгоритм выбора) занимает больше половины блока собственно моделирования, причем эту часть программировать значительно сложнее, чем конкретную часть блока, которую можно просто "списывать" со схемы действий. Применение УПМ позволяет свести программу моделирования к ее конкретной части, которая в большинстве случаев составляет меньше половины всей программы.

УПМ облегчает труд программиста не только количественно, но и качественно. УПМ вместе с алгоритмическим языком и транслятором с него дает средство формального описания моделей СМО и аппарат реализации этого описания на ЭВМ. Если составлена математическая модель СМО, то при наличии УПМ разработка программы моделирования превращается в техническую работу.

13. ЗАКЛЮЧЕНИЕ

Рассмотренная в настоящей работе система понятий позволяет произвести постепенную формализацию описания СМО - от реальной СМО до программы моделирования. Последняя ступень этой формализации - составление программы моделирования - рассмотрена лишь для простейших примеров СМО. Фактическое программирование задач статистического моделирования СМО с применением специального математичес-

кого обеспечения - языка моделирования А4М - рассмотрено в [9] . Язык А4М является расширением языка А4 и имеет встроенную универсальную программу моделирования РМ-6.

Систему понятий можно использовать независимо от математического обеспечения как простое, наглядное и достаточно полное средство описания и моделирования СМО.

Литература

1. Саати Т.Л. Элементы теории массового обслуживания и ее приложения. М., "Советское радио", 1971, 520 с.
2. Дал О., Нигард К. Симула - язык для программирования с дискретными событиями. - В кн.: Алгоритмы и алгоритмические языки, вып.2. М., АН СССР, 1967, 72 с.
3. Марковиц Г., Хаусснер Б., Карр Г. Симскрипт. Алгоритмический язык для моделирования. М.,"Советское радио", 1966, с.152.
4. Herscovitch H., Schneider T. GPSS III. An expanded general purpose simulator. "IBM Syst.J"., v.4, No.3, 1965.
5. Бакаев А.А.,Костин Н.И.,Яровицкий Н.В. Автоматные модели экономических систем.К.,"Наукова думка",1970. 192 с.
6. Ионин Г.Л.,Седол Я.Я. Статистическое моделирование случайных процессов. - Международная конференция по теории вероятностей и математической статистики. Тезисы докладов, т.I. Вильнюс, 1973, с.281-284.
7. Ионин Г.Л., Седол Я.Я. Программирование и статистическое моделирование на БЭСМ-4, Рига, ЛГУ им.П.Стучки, 1976. 204 с.
8. Седол Я.Я. Алгоритмический язык А4. Настоящий сб., с.3-43.
9. Седол Я.Я. Язык статистического моделирования А4М. Настоящий сб., с.76-117.

ЯЗЫК СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ А4М

Седол Я.Я.

(ВЦ ЛГУ им.П.Стучки)

I. ВВЕДЕНИЕ

В настоящей работе рассматривается расширение языка А4 [I] - язык А4М для составления программ статистического моделирования систем массового обслуживания (СМО). Язык А4М основывается на системе понятий, изложенной в [2] и является ее логическим завершением. А4М дает средства для построения высшей ступени формализации описания СМО - программы моделирования. Язык А4М имеет встроенную (т.е. содержащуюся в математическом обеспечении ЭВМ) универсальную программу моделирования, упомянутую в [2] . Язык А4М предназначается для записи конкретной части программы моделирования, а универсальная часть добавляется автоматически.

По сравнению с другими известными языками моделирования (Симула, Симскрипт, GPSS) А4М имеет простое и наглядное строение, дает предельно короткие и рациональные программы. Математическое обеспечение для А4М реализовано в ВЦ ЛГУ им.П.Стучки на ЭВМ БЭСМ-4.

В настоящей работе используются термины и обозначения, введенные в [I] и [2] . В частности, формальные объекты языка А4М выделяются из текста посредством фигурных скобок.

2. СТРОЕНИЕ ПРОГРАММЫ

Язык А4М формально отличается от А4 лишь тем, что имеет расширенный набор встроенных подпрограмм (ВП). Обозначения ВП языка А4М строится из букв и цифр, причем обозначения специфических для А4М АП начинаются буквой.

Программа на языке А4М представляет собой конкретную

часть программы моделирования и называется исходной программой. В ходе трансляции конкретная программа дополняется универсальной частью и становится рабочей программой.

Рабочая программа имеет четырехступенчатое строение. Первая ступень, в которую входит основной блок исходной программы, вызывает моделирование обращением к ВП - РМ. Вторая ступень (в универсальной части) реализует общую схему моделирования, обращаясь к блокам третьей ступени (в конкретной части), в которых запрограммирована собственно модель. Четвертую ступень составляет набор ВП - подпрограммы моделирования - последством которых третья ступень обращается к универсальной части, для выполнения специфических элементов моделирования.

2.I. Обращение к РМ.

Основной блок исходной программы вызывает работу модели путем (прямого или косвенного) обращения к ВП - РМ. При выполнении этого обращения управление передается управляющему блоку универсальной части, который реализует общую схему моделирования по алгоритму, описанному в [2]. Обращение имеет вид $\{*PM*m,n;\}$, где $\{m\}$ и $\{n\}$ нумералы, означающие: m - число переменных динамической информации на один вызов; n - число очередей, предусмотренных в модели. Если m или n равны нулю, их можно опускать: например, $\{*PM*3;\}$ означает $m=3$, $n=0$, а $\{*PM*;\}$ - $m=n=0$. Программа может содержать несколько обращений к ВП - РМ, но обязательно с одинаковыми m, n. На основании значений m, n транслятор формирует универсальную часть программы: обработка динамической информации включается при $m\neq 0$, а работа с очередями - при $n\neq 0$.

2.2. Стандартные блоки

Основу модели составляют блоки исходной программы, называемые стандартными блоками. Они имеют следующие идентификаторы и назначение: SA - построение начального состоя-

ния модели; SB - задание начальных значений для статистической информации; MD - собственно моделирование (схема действий); RZ - обработка и вывод результатов; NF - управление потоком и сбор информации.

Назначение блоков SA, SB, RZ полностью соответствует описанию [2]. Блок MD представляет собой запись схемы действий, где специфические для моделирования действия (источник, задержка, генерация, ожидание, активизация) заменены обращениями к соответствующим ВП. Описание блока NF будет дано в 5.I. К стандартным блокам обращается управляющий блок универсальной части. Исходная программа может содержать и другие блоки, введенные программистом, к которым обращаются стандартные блоки.

3. ВСТРОЕННЫЕ ПЕРЕМЕННЫЕ

Язык A4U имеет четыре массива встроенных переменных для обмена информацией между конкретной и универсальной частью. Массивы встроенных переменных Q, W, T, QJ определяются автоматически, их определение в программе недопустимо. Перечень встроенных переменных приведен в таблице I, где обозначено: К - конкретная часть, У - универсальная часть.

Таблица I. Встроенные переменные

Имя	тип	Назначение	Автоматическое значение
Q	A	Обмен У → К	*
Q1	A	Число серий	5
Q2	A	Длина серии	2000
Q3	A	Длина предварительной серии	200
Q4	A	Параметр потока	I
Q5	A	Среднее время задержки	I
Q6	A	Обмен К → У	-

W	A	Счетчик вызовов	
W1	A	Значение W для конца серии	*
W2	A	Счетчик серий	*
T	A	Текущее время	*
T1	A	Длительность эксперимента	*
T2	A	Начало эксперимента	*
QJ	J	Число непроверенных вызовов в очереди	*

Указанные автоматические значения (предназначенные для отладки) засылаются при трансляции, их можно менять путем ввода исходных данных, а также в ходе работы программы. Звездочка означает, что значением переменной управляет универсальная част. Переменная W получает начальное значение 0 в универсальной части, но дальнейшее изменение программируется в конкретной части. Значение W1 указывает то значение W , при достижении которого следует конец серии. Можно вызвать "искусственное" окончание серии, например, с помощью команды {W=W1} .

В T2 хранится значение текущего времени T в момент окончания предварительной серии. Значение T1 по формуле T1=T-T2 подсчитывается в конце каждой серии, поэтому пользоваться правильным значением T1 можно только в блоке R2 .

Встроенные переменные размещаются перед остальными переменными в порядке, указанном в таблице I.

4. ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ

Первые n переменных, определенные в исходной программе, где число n задается при обращении к ВП - РМ (см.2.I.), составляют активную часть динамической информации и называются динамическими переменными. Динамичес-

кие переменные могут иметь любые обозначения и типы, но команды их определения обязательно пишутся перед другими командами определения: следовательно динамические переменные располагаются непосредственно после встроенных переменных. Значения динамических переменных автоматически сопряжены с действующими вызовами - при наступлении пассивного действия эти значения передаются на хранение и возвращаются, когда вызов вновь активизируется.

5. ПОДПРОГРАММЫ МОДЕЛИРОВАНИЯ

Язык А4М имеет следующий набор ВП для реализации основных элементов моделирования (табл.2). Список информации ВП моделирования либо пуст, либо имеет длину I.

Таблица 2. ВП моделирования

Обозначение	Назначение		Список информации
M	Источник		-
N	Задержка		-
P	Генерация		Метка нового вызова
F			-
F1	Равномерные случайные числа		-
F2			-
F3	Установка начала	Случайных чисел	-
F4	Изменение		-
F5			-
E	Экспоненциальные		-
E1	случайные		-
E2	числа		-
R	Ожидание		
S			
S1			
S2	Активизация		Номер очереди
S3			
S4			
S6			
T	Разгрузка		-

5.I. Источник

В А4И возможно образование в модели одного источника посредством обращения {*M*;} . Если в модели требуется несколько источников, то они, кроме одного, программируются в виде комбинации генерация-задержка. С источником, который образован с помощью ВП - M связана работа блока NF исходной программы. Обращение к NF происходит перед каждым поступлением вызова из источника. Если требуется не экспоненциальное распределение промежутков между вызовами, то в NF засылается в Q6 реализация очередного промежутка. Когда в NF значение Q6 не изменяется, реализуется автоматический пуассоновский поток с параметром, равным значению Q4 . Блок NF можно использовать в случае простейшего потока для сбора статистической информации о состоянии системы. Наличие в программе блока NF , а также источника, образуемого посредством ВП - M , не обязательно.

5.2. Задержка

Задержка программируется в виде обращения {*N*;} . Длительность задержки задается в Q6 . Если Q6 "не тронуто", то реализуется экспоненциально-распределенная длительность задержки, среднее значение которой равно значению Q5 .

Значение Q6 во всех случаях его использования имеет "одноразовое" действие - при работе ВП, использующих Q6 , его значение переходит в "нетронутое".

Пример. Операторы {Q6=2; *N*;} реализуют задержку с постоянной длительностью, равной 2 , а операторы {Q5=2; *N*;} - экспоненциально распределенную длительность со средним значением 2 .

5.3. Генерация

Для генерации используется ВП- P . Обращение имеет вид $\{*P*m;\}$, где $\{m\}$ - нумерал, указывающий метку, по которой передается управление для действий нового вызова. Действия "старого" вызова продолжаются на следующем за обращением операторе.

Пример. Требуется образовать в модели дополнительный источник, дающий поток с постоянными промежутками длины I/2. Решение: { 1.G6 = 5 Δ 0 ; *N* ; *P*1 ; }

5.4. Случайные числа

В А4М имеется набор ВП для получения равномерно и экспоненциально распределенных чисел. Три серии равномерно распределенных в интервале [0,1] случайных чисел. дают ВП- F и ВП- F1 соответственно. Каждое обращение переводит соответствующую серию на следующее значение и выдает это значение в G . Аналогично ВП- E , ВП- E1 и ВП- E2 дают (также в G) три серии экспоненциально распределенных случайных чисел со средним значением 1 . Для выработки экспоненциальных случайных чисел используются в качестве исходного материала соответствующие серии равномерных чисел: экспоненциальное случайное число e образуется от соответствующего равномерного числа f по формуле $e=-\ln f$.

Имеется вспомогательные ВП для управления сериями равномерных случайных чисел, а именно: F3 - установка исходных значений всех серий; F4 - циклическая перестановка серий; F5 - переключение всех серий на "обратные" (замена f на $1-f$). Обращение к ВП- F3 перед началом моделирования происходит автоматически из универсальной части. Использование ВП- F4 и ВП- F5 будет рассмотрено в 6.3.

Автоматическое экспоненциальное распределение для источника реализуется посредством ВП- E1 , а для задержки - посредством ВП- E2 . В конкретной части рекомендуется поль-

зоваться в основном "нулевой" серией, т.е. ВП- F и ВП- E .

Пример. Требуется построить случайное число X, принимающее значения $1, 2, \ldots, 10$ равновероятно, т.е. каждое с вероятностью $1/10$. Решение: {*F*; X=ENT(Q*10)+ +1;}.

5.5. Ожидание

В моделях, имеющих действие типа "ожидание", вводятся совокупности ожидающих вызовов, называемые очередями. Число очередей n задается в обращении к ВП- PM(см.2.I). Очереди нумеруются числами $1, 2, \ldots, n$. Очередь имеет линейное строение. Поступивший на ожидание вызов либо становится в конце очереди (становление по порядку), либо занимает равновероятно любое место в ней (случайное становление). Просмотр очереди при активизации ожидающих вызовов всегда производится с начала.

Обращение к ВП- R для становления в очередь имеет вид {*R*i} , где {i} - нумерал или имя типа J , указывающий номер очереди. Номер очереди можно опускать, если предыдущее (по порядку выполнения) действие типа "ожидание" или " активизация" производилась над той же очередью, а также в случае $n=1$.

Порядок становления в очередь определяет значение Q6- если это значение равно 1 , то реализуется случайное становление, в остальных случаях (в частности - при "нетронутом" Q6) - становление по порядку.

Пример. Случайное становление в четвертую очередь реализуется в виде {Q6=1; *R*4;}.

5.6. Активизация

Действия типа "активизация" строятся из следующих трех элементов: α - просмотр очереди, β - исключение из очереди, γ - закрытие просмотра.

Имеются два состояния просмотра очереди - открытое и закрытое. В начале моделирования состояние автоматически

закрытое. Действие α, выполненное при закрытом состоянии, переводит состояние в открытое и ставит "указатель просмотра" на первый вызов очереди. При открытом состоянии действие α переводит "указатель" на следующий вызов очереди. Открытие просмотра сопровождается передачей на хранение динамической информации "просматривающего" вызова, которая возвращается после закрытия просмотра. Во время просмотра динамические переменные принимают информацию того вызова, на котором стоит "указатель". В αJ при действии α выдается число еще не просмотренных вызовов очереди (включая просматриваемый). Если таких вызовов нет (очередь исчерпана), то состояние просмотра автоматически переходит в закрытое, и в αJ засылается нуль.

Действие β выпускает из очереди и активизирует тот вызов, на котором стоит "указатель просмотра". Выполнение β при закрытом состоянии просмотра не допускается.

Действие γ переводит просмотр в закрытое состояние. Закрытие просмотра посредством γ допускается только непосредственно после действия β.

Для выполнения рассмотренных действий имеется группа ВП, реализующих различные их комбинации согласно табл.3.

Таблица 3. ВП активизации

ВП	действия
S	$\alpha\beta\gamma$
S1	α
S2	β
S3	$\alpha\beta$
S4	γ
S6	$\beta\gamma$

В случаях ВП- S , и ВП- S3 , если действие α исчерпывает очередь (т.е. очередь пуста), то следующие действия не производятся.

Номер очереди при обращении к ВП активизации указывает-

ся по тем же правилам, что для ВП- R (см.5.5).

Пример. Требуется активизировать последний вызов J -ой очереди (если такой имеется), т.е. реализовать принцип "последний пришел - первым обслуживается". Решение: { 1. *S1*J; @J>1*1; @J=0*2; *S6*; 2.}.

5.7. Разгрузка

Действие "разгрузка" предназначено для контроля правильности работы программы. При обращении вида {*T*;} выключается источник и модель работает до тех пор, пока число задержанных вызовов не станет равным нулю. После этого в правильно составленной модели все элементы структурной части должны быть в состоянии "свободно". Применение разгрузки в моделях с источниками, построенными на основании генераций, не допустимо.

Пример применения разгрузки рассмотрен в 8.4.

6. РЕАЛИЗАЦИЯ НА БЭСМ-4

Транслятор с языка A4M на БЭСМ-4 объединен с транслятором A4TI, поэтому комплектация, запуск и трансляция A4M-программ ничем не отличается от тех же действий для A4-программ. Признак по которому транслятор отличает A4M-программу от A4-программы - наличие в программе обращения к ВП- PM .

Транслятор содержит универсальную программу моделирования PM6 , из которой строится универсальная часть рабочей программы PM6 производит моделирование на основе процесса с реальными временами. PM6 содержит следующие части, которые при надобности автоматически включаются в программу (табл.4).

Таблица 4. Части PM6 .

Обозначение	Назначение	Номер	Условие включения в программу
PM	Управляющая часть	1715	Обращение в ВП- PM
M	Процесс моделирования	0150	Обращение к ВП- PM
E	Экспоненциальные } случайные	0050	Обращение к ВП- PM
F	равномерные } числа	0060	Обращение к ВП- PM
P	Генерация	0170	Обращение к ВП- P
L	Обработка динамической информации	0140	$m \neq 0$
R	Работа с очередями	0200	$n \neq 0$

Номера частей печатаются в таблице блоков (ТБ), по которой можно контролировать строение рабочей программы.

Максимальное число вызовов, одновременно имеющихся в модели, ограничивается числом $N = \left[\frac{4090 - 3n}{m+1}\right]$

Превышение этого предела (переполнение модели) вызывает останов в 7741. Печатать информацию рекомендуется только в блоке RZ . В целях отладки допускается печать из блока MD в начальной стадии моделирования или в моделях с небольшим числом вызовов в системе. Печать из MD при наличии большого количества вызовов в системе (порядка $\geq$ 100) может привести к неверной работе модели.

Использование ВП случайных чисел допускается в программах, не имеющих обращения к ВП- PM . В этом случае встроенных переменных нет, а результат выдается в первой из определяемых программой переменных. Перед началом работы обязательно обращение к ВП- F3 .

7. ПРИМЕР ПРОГРАММЫ

Рассмотрим составление программы моделирования на языке A4M для неполнодоступной схемы (НС) с ожиданием. Постановка задачи почти полностью совпадает с примером, рассмотренным в [3] .

7.1. Постановка задачи

Рассматривается НС с параметрами q (число групп), σ (число линий) и d (доступность). На НС поступает простейший поток вызовов с параметром Λ. Поступивший вызов выбирает равновероятно одну из q групп и ищет в порядке слева на право первую свободную доступную линию. Если линия найдена, то она занимается вызовом на время обслуживания средней длительностью t с экспоненциальным распределением. Если все доступные линии заняты, то вызов становится на ожидание, сохраняя выбранный номер группы. Выход из очереди при освобождении линии предполагается в двух вариантах: а) в порядке поступления, б) случайно.

Требуется определить следующие характеристики: вероятность становления на ожидание p, среднюю длину очереди n, среднее число занятых линий z, распределение числа занятых линий, т.е. вероятность μ_i $(i = 1,2,\ldots,\nu)$ того, что занято ровно i линий.

7.2. Функциональная модель

В первую очередь составляется функциональная модель в виде схемы действий (рис.I).

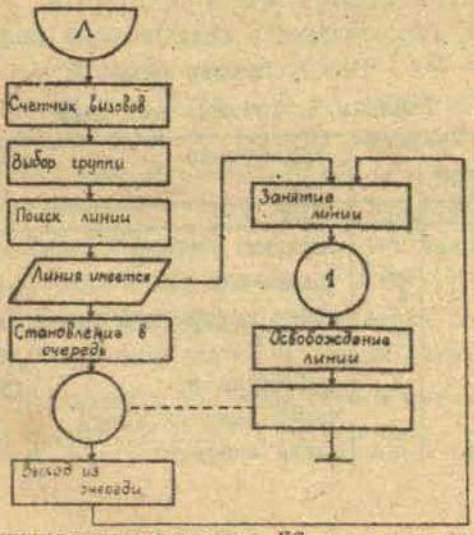


Рис. I. Функциональная модель НС с ожиданием

7.3. Структурная часть

Составление математической модели начинается построением структурной части в виде таблицы массивов (ТМ).

Исходные данные состоят из нагрузки A (вводится прямо в $Q4$, вариант очереди Z (0 - в порядке поступления, 1 - случайно) и записи строения НС в виде массива S из $g \times d+3$ чисел. Первые три числа в S представляют собой параметры g, v, d следующие - номера линий по точкам контактного поля. Напр. НС, изображенная на рис.2 записывается в виде $\{S: 4,6,3,1,3,5,1,4,6,2,3,6,2,4,5;\}$.

Динамическая информация занимает одну переменную и указывает номер группы (для ожидающих) или номер линии (для обслуживающихся), занятой вызовом. Для удобства программирования поиска линии вместо номера группы j используется число $(j-1) \times d$ указывающее начало группы в записи НС. Состояние НС записывается в массиве A, где каждой линии отводится одна переменная (0 - свободно, 1 - занято). Назначение остальных массивов определено в ТМ (табл.5).

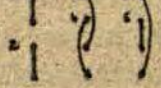


Рис.2. Пример НС.

Таблица 5. Таблица массивов

Идентификатор	Тип	Границы от	Границы до	Назначение	Замечания
1	2	3	4	5	6
Z	A	-	-	вариант очереди	Исходные
S	J	0	$g \times d+2$	запись НС	данные
E	J	-	-	номер группы или линии	Динамическая переменная
A	J	1	v,	состояние НС	Структурная
L	J	-	-	число занятых линий	часть
U	A	-	-	длина очереди	модели

1	2	3	4	5	6
P	A	0	v+3	сбор информации для p_i, n_i, k_i и μ_i $(i=1,\dots,v)$	Статистическая
R	A	0	v+3	массив результатов (p_i, n_i, k_i, μ_i)	информация
M	J	-	-	$=v+3$	
DM	J	-	-	$=d-1$	
D	A	-	-	$=d$	Вспомогатель-
G	A	-	-	$=g$	ные
K	J	-	-	номер найденной линии	переменные
J	J	-	-		
J	J	-	-		

7.4. Программа

На основе схемы действий (рис.I) и ТН (табл.5) будем писать программу моделирования, пропуская составление схемы действий для математической модели.

Предположим, что программы НС не превышают пределы $v=50$, $gd=100$. Теперь можем написать необходимые определения: {:J:E,S⌣10I, A1⌣50; :A: P⌣53, R⌣53;}. Остальные массивы определятся автоматически. Назовем нашу программу NPG и напишем основной блок: {NPG.*PM*1,1;}. Далее пишутся блоки SA и SB. В SA образуются значения вспомогательных переменных и строится исходное состояние структурной части: {SA. M=51+3; DM=52-1; D=JA(52); G=JA(5); A1⌣51=0; L=0; U=0;}. В SB числится массив P {SB. P⌣M=0;} Блок NF будем использовать для сбора информации о длине очереди и числе занятых линий {NF. P1+U; P3L+1;}. Теперь приступим к разработке блока MD. Оформим в виде отдельного блока M поиск линии в заданной группе. Блок M выдает в K номер первой свободной линии или нуль, если свободных линий в группе нет:

{M. J=E:DM; J=E,J; K=533J; A0K=0*;; K=0;}.

Далее будем писать по частям блок MD :

а) Источник и подсчет вызовов: { MD. * M * ; W + 1 ; }

б) Выбор группы: { * F * ; E = AJ (ENT (Q * G) * D) ; }

в) Поиск линии и проверка ее наличия: { * M ; K ≠ 0 * 1 ; }

г) Становление в очередь (с подсчетом вызовов), ожидание и выход из очереди: { P + 1 ; U + 1 ; 86 = Z ; * R * ; U - 1 ; }

д) Занятие линии, обслуживание и освобождение линии: { 1.E = K ; AOE = 1 ; L + 1 ; * N * ; AOE = 0 ; L - 1 ; }

е) Проверка очереди: { 2. * 51 * ; BJ = 0 * ; * M ; K = 0 * 2 ; * 56 * ; }

Наглядное изображение алгоритма проверки очереди показано на рис.3.

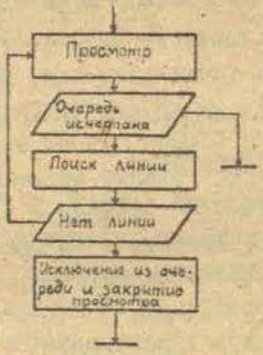


Рис.3. Схема проверки очереди

Случай "нет линии" означает, что освободившаяся линия не доступна группе, в которой ожидает просматриваемый вызов. Заключает программу блок R2 . Работа блока заключается в построении и печати массива результатов R . Все элементы R , за исключением R2 , образуются из соответствующих P делением на число вызовов W . Значение R2 – среднее число занятых линий Z – подсчитывается по формуле

$$Z = \sum_{i=1}^{V} i \cdot \mu_i$$

Кроме R, в R2 печатаются значения W и T, T1 { R2 R_M = P_/ J ; J = 1,51 ; R2 + R3J * JA(J) ; ; * 0 * W, T_1, R_2, R3_M ; }.

Теперь программа окончена. Для контрольного запуска ее составляется набор исходных данных, например, запись НС из 7.3 и { J4 : 4, Z : 0 } , т.е. A = 4 , вариант 0 . Лис-

тинг программы с исходными данными приведен в приложении I.

8. КОНТРОЛЬ ПРАВИЛЬНОСТИ

Когда программа составлена и отлажена до такой степени, что она работает и дает результаты, необходимо убедиться в правильности результатов. В настоящем разделе рассматриваются некоторые способы контроля программ моделирования на примере программы предыдущего раздела.

8.I. Сравнение с точными данными

Моделирование всегда дает приближенные значения характеристик модели, причем моделируются, как правило, системы, для которых получить точные результаты трудно или невозможно. Однако часто путем изменения исходных данных можно построить такой частный случай модели, который имеет способ получения точных характеристик.

В нашем примере можно воспользоваться тем, что полнодоступная схема (ПС) является частным случаем НС, а ПС с ожиданием имеет точные формулы для вероятности ожидания p и средней длины очереди n, а именно:

$$p = \frac{\frac{\lambda^{v}}{(v-1)!\,(v-\lambda)}}{\frac{\lambda^{v+1}}{v!\,(v-\lambda)} + \sum_{i=0}^{v} \frac{\lambda^{i}}{i!}}, \qquad n = \frac{\lambda}{v-\lambda}\, p,$$

(v – число линий, λ – параметр потока).

Возьмем пример $v=3$, $\lambda=2$. Исходные данные для программы имеют вид $\{0,4;2;\ 2;0;\ 5;1,3,3,1,2,3;\}$. Моделирование (5 автоматических серий по 2000 вызовов) дает $p=0,441$; $n=0,984$, а точный расчет по формулам $p=0,(4)$; $n=0,(8)$. Сравнение показывает, что программа моделирования, по крайней мере для p, дает достоверный результат.

Недостаток метода заключается в том, что программа проверяется на упрощенном примере и проверка может оказать-

ся неполной. В нашем примере не проверяется ветвь, соответствующая случаю "нет линии" (рис. 3) и ошибка, например, {K=0*;} вместо {K=0*2;} в предпоследней команде блока MD не обнаружилась бы.

Модификацией метода можно считать использование соотношений, вытекающих из теоретических соображений. В нашей модели такое соотношение $z=\Lambda$, следующее из того, что вся поступающая нагрузка обслуживается. Запуск примера, показанного в приложении I, дает $z=3,9976$, что близко к заданной $\Lambda=4$.

8.2. Метод перегрузки

Если в модели имеется возможность менять поступающую нагрузку Λ, то представляет интерес моделирование при предельно больших Λ. Обычно значения характеристик системы при больших Λ легко определяются теоретически.

В нашем примере значение Λ ограничено соотношением $\Lambda<\nu$ (при $\Lambda>\nu$ модель не стационарна). Если положить $\Lambda=\nu$, то теоретически длина очереди бесконечно возрастает, но это происходит очень медленно и практически модель находится в почти равновесном состоянии. Следует ожидать, что при $\Lambda=\nu$ характеристики системы близки к $p=1$, $z=\nu$, $\mu_0=\mu_1=\dots=\mu_{n-1}=0$, а значение n с течением времени медленно возрастает. Запуск примера с данными {0.4:6;2:0;} и НС из рис. 2 дает следующие результаты: $p=0,9864$; $z=5,9635$; $n=25,36$; 80,05; 83,98; 77,24; 76,56 (по сериям), что свидетельствует о правильности программы.

В случаях систем с потерями метод перегрузки осуществляется заданием очень больших значений Λ (напр. $\Lambda=10^{10}$).

8.3. Изменение случайных чисел

Рассмотренные выше методы контроля заключаются в сравнении результатов моделирования с предполагаемыми. Такое сравнение имеет субъективный характер - моделирование

всегда дает приближенные результаты, которые более или менее отличаются от точных. Даже значительные расхождения не всегда вызываются ошибками в программе - некоторые модели имеют большой разброс (дисперсию) результатов, и отклонения могут быть чисто случайными. Для отделения систематических ошибок от случайных применяется метод изменения источников случайных чисел. Если модель при всевозможных изменениях случайных чисел дает отклонение в одну сторону, то следует искать ошибку в программе (систематическая ошибка), в противном случае отклонения имеют случайный характер.

Для изменения случайных чисел язык АЗМ имеет ВП- *F4* и ВП- *F5* . Первая из них дает циклическую перестановку трех источников (серий) равномерных случайных чисел, а вторая делает обращение всех серий - вместо напр., серии $f_1, f_2, \ldots$ выдается $1-f_1, 1-f_2, \ldots$. Каждое обращение к ВП- *F4* или ВП- *F5* сопровождается автоматической установкой начала всех серий. Всего можно построить 6 вариантов источников случайных чисел.

Запрограммируем в нашем примере моделирование во всех шести вариантах. Для этого основной блок программы заменяется на следующий: {NPG3. A3=1,3; B3=1,2; *PM=1,1; *F5*; *F4*;}, где A3, B3 - вспомогательные переменные. Запуск программы NPG3 на примере из 8.I. дает следующие результаты (таблица 6), что свидетельствует о случайном характере отклонений:

Таблица 6. Результаты моделирования

Вариант	p	π	$\hat{\pi}$
I	0,4406	0,9840	I,9922
2	0,450I	0,9006	2,02I8
3	0,4703	I,0835	2,0450
4	0,424I	0,9467	I,9656
5	0,4302	0,873I	I,9602
6	0,4475	I,04I7	2,0083
точные	0,44444	0,88889	2,0000

8.4. Применение разгрузки

Язык А4М имеет средство, контролирующее программу не по результатам, а непосредственно. Это действие разгрузки, реализуемое посредством ВП-Т. Разгрузка "очищает" систему от вызовов и для правильной программы в результате получается первоначальное полностью свободное состояние структурной части.

В реализации на БЭСМ-4 разгрузку можно делать только в самом конце моделирования, ибо после разгрузки модель работать уже не может. После разгрузки ставится печать структурной части модели.

В нашем примере разгрузку можно добавить в конце программы в виде дополнительного блока, одноименного с основным блоком: { NPG.*T*;*O* A4⌐S4,L,U;} . Если программа правильна, то все печатаемые числа равны нулю.

9. ЗАКЛЮЧЕНИЕ

Сравнение рассмотренного в 7. примера программы с программой той же задачи на языке АЗ [3] показывает, что язык А4М дает программу значительно меньшего объема, более наглядную и совершенную. С другой стороны, язык А4М (а также А4) в некоторых деталях уступает рассмотренной в [3] системе программирования: в А4М нет средств динамического распределения памяти, нет автоматического вычисления доверительных интервалов и автоматической печати результатов моделирования на АЦПУ, не предусмотрено моделирование цепью. Эти и другие недостатки предполагается устранить в ходе дальнейшего совершенствования языков А4, А4М и системы математического обеспечения. В перспективе намечается создание на основе принципов А4 и А4М языка программирования высокого уровня, по возможностям не уступающего другим языкам подобного назначения, но имеющего более простое и наглядное строение.

Литература

I. Седол Я.Я. Алгоритмический язык А4. Наст.об., с.3 - 43.

2. Ионин Г.Л., Седол Я.Я. Описание и моделирование систем массового обслуживания. Наст.об.,с.44 - 75.

3. Ионин Г.Л., Седол Я.Я. Программирование и статистическое моделирование на БЭСМ-4, Рига, ЛГУ им.П.Стучки, 1976. 204 с.

Приложение I

Программа

```
1   !!: E, S=102, A1=50;   !A: P=53, R=53;
    NPG.*PM*1.1;
2   SA.M=S1+3; DM=S2-1; D=!A(S2); G=!A(S);
      A1=S1=0; L=0; U=0;
3   S8.P=M=0;
    NF.P1*U; P3L*1;
4   M.J=E+DM; I=E.J; K=S3;; A0K=0*;; K=0;

5   ND.*M*; N*1; *F*; E=A1(ENT(R*G)*D);
      *M; K*0=1; P*1; U*1; Q6=7; *R*; U=1;
6     1.E=K; A0E=1; L*1; *N*; A0E=0; L=1;
      2.*S1*; Q1=0*; *M; K=0*2; *S6*;
7   RZ.RL=P=/W; I=1,S1; R2*R3;*!A(I);;
      *0* N; T=1. R=2. R3=M;
```

```
1   Q4;4;  Z;0;

2   S; 4. 0. 3.   1. 3. 5.
                  1. 4. 6.
3                 2. 3. 6.
                  2. 4. 5;
```

СТАТИСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ МНОГОКАСКАДНЫХ СХЕМ

Ионин Г.Л.

(ВЦ ЛГУ им.П.Стучки)

Многокаскадные (многозвенные) схемы имеют широкое применение в координатных автоматических телефонных станциях (АТСК), в квазиэлектронных и электронных АТС [1,2,3] . Определение вероятностных характеристик многокаскадных схем при проектировании АТС, особенно с учетом работы регистров, маркеров, цифровых комплектов и других устройств , проводится на основе статистического моделирования [4 + 13] . Составление и отладка программ моделирования многокаскадных схем требует много времени даже с применением алгоритмических языков моделирования как СИМСКРИПТ [14] , СИМУЛА [15] GPSS [16] . Статистическое моделирование многокаскадных схем трудно поддается унификации, хотя такие попытки предпринимаются [17] . В данной работе предлагается построение массивов, отражающих структуру и состояние многокаскадной схемы и алгоритм поиска занятия и освобождения линий в схеме. Алгоритм предполагается реализовать на ЭВМ в качестве блока конкретной части в системе программирования статистического моделирования, описанной в [18, 19] . Это позволит в значительной мере унифицировать статистическое моделирование многокаскадных схем. Имеются основания считать, что предложенный алгоритм достаточно эффективен.

I. ПОСТРОЕНИЕ МНОГОКАСКАДНЫХ СХЕМ

Многокаскадные схемы строятся из коммутаторов. Коммутатор представляет устройство из n входов и m выходов (рис.I). Любой вход коммутатора может быть соединен с любым выходом коммутатора. Число соединений в коммутаторе,

Рис. I. Коммутатор

которые занимают вход и выход коммутатора, не превышает $\min(n,m)$. Схема из s каскадов строится из α_1 коммутаторов первого каскада, α_2 коммутаторов второго каскада, ..., α_s коммутаторов s -го каскада.

Как правило, все коммутаторы i -го каскада ($i = 1, 2, \ldots, s$) имеют одинаковые параметры n_i (число входов коммутатора), m_i (число выходов коммутатора). Один вход коммутатора $(i+1)$ -го каскада соединен с одним выходом коммутатора i -го каскада ($i = 1, 2, \ldots, s-1$). Соединение задается соответствием (α, β) для всех выходов коммутаторов i -го каскада, где α - номер коммутатора для выхода i -го каскада и β - номер коммутатора для входа $(i+1)$ -го каскада. Совокупность соединений входов коммутаторов $(i+1)$ -го каскада с выходами коммутаторов i -го каскада, задаваемых соответствием (α, β) для $\alpha = 1, 2, \ldots, \alpha_i$, образует промежуточные линии (ПЛ) между i и $(i+1)$ каскадами. Примеры 6-каскадных схем приведены на рис.2 и рис.3. (6-каскадная схема рис.3 заимствована из работы [8]). Соответствие $(\alpha, \beta)_i$ ($i = 1, 2, \ldots, s-1$) для ПЛ схем рис.2 и рис.3 приведено в табл.I и 2 соответственно.

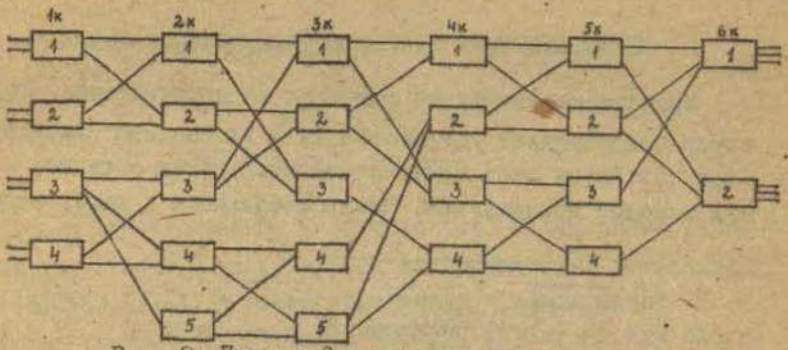


Рис. 2. Пример 6-каскадной схемы.
($a_1=4$, $a_2=5$, $a_3=5$, $a_4=4$, $a_5=4$, $a_6=2$)

Таблица I. Соответствие (α, β) для схемы рис.2

$i=1$, число ПЛ=9

α	β
I	I
I	2
2	I
2	2
3	3
3	4
3	5
4	3
4	4

$i=2$, число ПЛ=10

α	β
I	I
I	3
2	2
2	3
3	I
3	2
4	4
4	5
5	4
5	5

$i=3$, число ПЛ=8

α	β
I	I
I	3
2	I
2	3
3	4
4	2
5	2
5	4

$i=4$, число ПЛ=8

α	β
I	I
I	2
2	I
2	2
3	3
3	4
4	3
4	4

$i=5$, число ПЛ=6

α	β
I	I
I	2
2	I
2	2
3	I
4	2

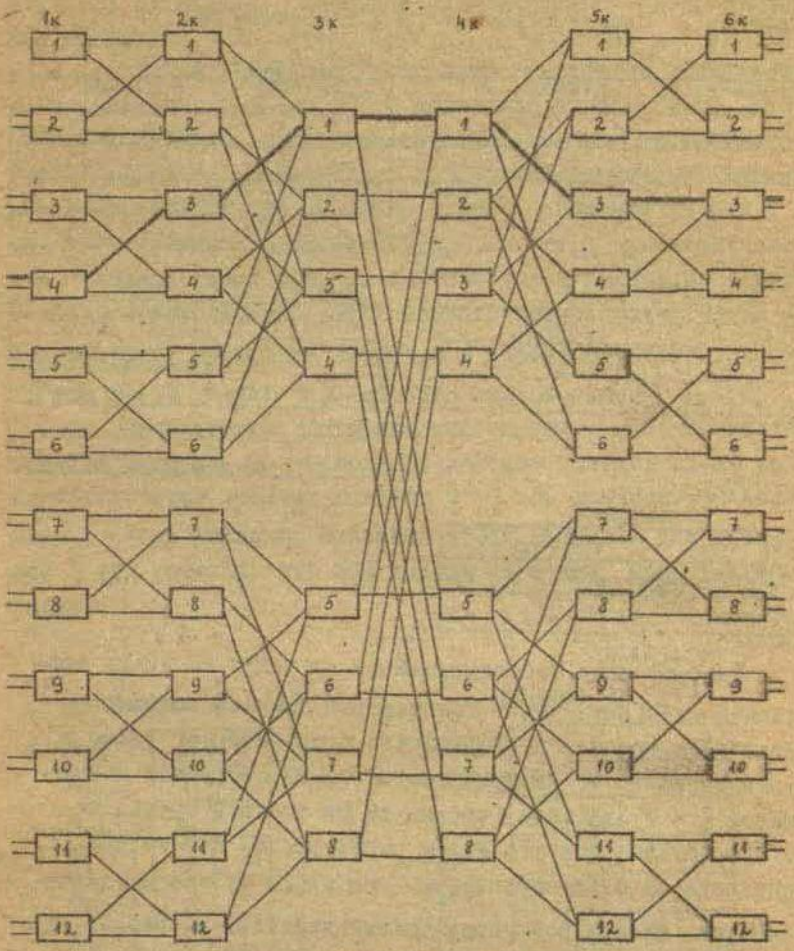


Рис.3. Пример регулярной 6-каскадной схемы ($a_1=12$, $a_2=12$, $a_3=8$, $a_4=8$, $a_5=12$, $a_6=12$, $n_1=2$, $m_1=2$, $n_2=m_2=2$, $n_3=3$, $m_3=2$, $n_4=2$, $m_4=3$, $n_5=2$, $m_5=2$, $n_6=2$, $m_6=2$)

Таблица 2. Соответствие (α, β) для схемы рис. 3

$i=1$, число ПЛ=24

Значения α	Значения β
α – нечетное	$\beta_1=\alpha$; $\beta_2=\alpha+1$
α – четное	$\beta_1=\alpha-1$; $\beta_2=\alpha$

$i=2$, число ПЛ=24

Значения α	Значения β
$\alpha=1,3,5$	$\beta_1=1$; $\beta_2=3$
$\alpha=2,4,6$	$\beta_1=2$; $\beta_2=4$
$\alpha=7,9,11$	$\beta_1=5$; $\beta_2=7$
$\alpha=8,10,12$	$\beta_1=6$; $\beta_2=8$

$i=3$; число ПЛ=16

Значения	Значения
$\alpha \leq 4$	$\beta_1=\alpha$; $\beta_2=\alpha+4$
$\alpha > 4$	$\beta_1=\alpha-4$; $\beta_2=\alpha$

$i=4$; число ПЛ=24

Значения	Значения
$\alpha=1,3$	$\beta_1=1$; $\beta_2=3$; $\beta_3=5$
$\alpha=2,4$	$\beta_1=2$; $\beta_2=4$; $\beta_3=6$
$\alpha=5,7$	$\beta_1=7$; $\beta_2=9$; $\beta_3=11$
$\alpha=6,8$	$\beta_1=8$; $\beta_2=10$; $\beta_3=12$

$i=5$, число ПЛ=24

Значения	Значения
α – нечетное	$\beta_1=\alpha$; $\beta_2=\alpha+1$
α – четное	$\beta_1=\alpha-1$; $\beta_2=\alpha$

Рассмотрение соответствий (α, β) для ПЛ между каскадами показывает, что в большинстве случаев выделяются группы коммутаторов i-го каскада (обозначим их число в группе через b_i), которые соединены ПЛ с группой коммутаторов $(i+1)$ каскада (обозначим их число в группе через c_i). Будем называть схемы регулярными, для которых соединения ПЛ между группами i-го и $(i+1)$-го каскадов идентичны. Пример регулярной схемы приведен на рис.3, значения b и c для которой принимают значения: $b_1=2$, $c_1=2$, $b_2=6$, $c_2=4$, $b_3=8$, $c_3=8$, $b_4=4$, $c_4=6$, $b_5=2$, $c_5=2$. Схема рис.2 не является регулярной.

2. ПРЕДСТАВЛЕНИЕ СТРУКТУРЫ И ПОИСК СВОБОДНОЙ ЛИНИИ В МНОГОКАСКАДНОЙ СХЕМЕ

Будем рассматривать регулярные схемы, так как принципиальной разницы нет, а описание является более простым в связи с одинаковыми значениями b_i, c_i для групп ПЛ между i

i ($i+1$) каскадами. Состояние многокаскадной схемы представим массивами $A(0)$, $A(1)$, $A(2)$, ..., $A(s)$, где массив $A(i)$ ($i = 1, 2, \ldots, s$) отражает состояние выходов из коммутаторов i-го каскада, массив $A(0)$ - состояние входов схемы. Массив $A(i)$ ($i = 0, 1, 2, \ldots, s$) состоит из α_i ячеек (переменных). В ячейке массива $A(i)$ $i = 1, 2, \ldots, s-1$ используется c_i двоичных разрядов, k-ый ($k = 1, 2, \ldots, c_i$) двоичный разряд в ячейке $A_j(i)$ ($j = 1, 2, \ldots, \alpha_i$; $i = 1, 2, \ldots, s-1$.) принимает значение "единица" если соответствующий выход свободен, и принимает значение "нуль", если соответствующий выход занят или отсутствует. Нумерация разрядов в ячейке производится справа налево. В ячейке $A_j(0)$ массива входов схемы ($j = 1, 2, \ldots, \alpha_0 = \alpha_1$) записывается число занятых входов j-го коммутатора первого каскада. В ячейке $A_j(s)$ массива выходов схемы записывается число занятых выходов в j-ом коммутаторе последнего (s-го) каскада. Значения ячеек массивов для схемы рис.3 при условии, что все выходы свободны, представлены в табл.3.

Таблица 3

Массив A (1)

$A_1(1)$	1	1
$A_2(1)$	1	1
$A_3(1)$	1	1
$A_4(1)$	1	1
$A_5(1)$	1	1
$A_6(1)$	1	1
$A_7(1)$	1	1
$A_8(1)$	1	1
$A_9(1)$	1	1
$A_{10}(1)$	1	1
$A_{11}(1)$	1	1
$A_{12}(1)$	1	1

$c_1 = 2$

Массив A (2)

$A_1(2)$	0	1	0	1
$A_2(2)$	1	0	1	0
$A_3(2)$	0	1	0	1
$A_4(2)$	1	0	1	0
$A_5(2)$	0	1	0	1
$A_6(2)$	1	0	1	0
$A_7(2)$	0	1	0	1
$A_8(2)$	1	0	1	0
$A_9(2)$	0	1	0	1
$A_{10}(2)$	1	0	1	0
$A_{11}(2)$	0	1	0	1
$A_{12}(2)$	1	0	1	0

$c_2 = 4$

Массив А (3)

$A_1(3)$	0	0	0	I	0	0	0	I
$A_2(3)$	0	0	I	0	0	0	I	0
$A_3(3)$	0	I	0	0	0	I	0	0
$A_4(3)$	I	0	0	0	I	0	0	0
$A_5(3)$	0	0	0	I	0	0	0	I
$A_6(3)$	0	0	I	0	0	0	I	0
$A_7(3)$	0	I	0	0	0	I	0	0
$A_8(3)$	I	0	0	0	I	0	0	0

$b_3(8)$

$c_3 = 8$

Массив А (4)

$A_1(4)$	0	I	0	I	0	I
$A_2(4)$	I	0	I	0	I	0
$A_3(4)$	0	I	0	I	0	I
$A_4(4)$	I	0	I	0	I	0
$A_5(4)$	0	I	0	I	0	I
$A_6(4)$	I	0	I	0	I	0
$A_7(4)$	0	I	0	I	0	I
$A_8(4)$	I	0	I	0	I	0

$b_4=4$, $b_4=4$

$c_4 = 6$

Массив А (5)

$A_1(5)$	I	I
$A_2(5)$	I	I
$A_3(5)$	I	I
$A_4(5)$	I	I
$A_5(5)$	I	I
$A_6(5)$	I	I
$A_7(5)$	I	I
$A_8(5)$	I	I
$A_9(5)$	I	I
$A_{10}(5)$	I	I
$A_{11}(5)$	I	I
$A_{12}(5)$	I	I

$b_5=2$, $b_5=2$, $b_5=2$, $b_5=2$, $b_5=2$, $b_5=2$

$c_5=2$

Массив А (6)

$A_1(6)$	0
$A_2(6)$	0
$A_3(6)$	0
$A_4(6)$	0
$A_5(6)$	0
$A_6(6)$	0
$A_7(6)$	0
$A_8(6)$	0
$A_9(6)$	0
$A_{10}(6)$	0
$A_{11}(6)$	0
$A_{12}(6)$	0

Массив А (0)

$A_1(0)$	0
$A_2(0)$	0
$A_3(0)$	0
$A_4(0)$	0
$A_5(0)$	0
$A_6(0)$	0
$A_7(0)$	0
$A_8(0)$	0
$A_9(0)$	0
$A_{10}(0)$	0
$A_{11}(0)$	0
$A_{12}(0)$	0

Для эффективности поиска свободной линии в схеме создаются массивы $B(1)$, $B(2)$, ..., $B(S-1)$ и F . Массив $B(i)$ ($i = I, 2, \ldots, S-I$) состоит из $\frac{a_i}{b_i} \cdot \frac{a_{i+1}}{c_i}$ ячеек, в ячейке используется c_i разрядов. В ячейке $B_j(i)$ ($j = I, 2, \ldots, \frac{a_i}{b_i}$) κ -ый разряд соответствует κ - ому ($\kappa = I, 2, \ldots, c_i$) коммутатору в j -ой группе ($i + I$) - го каскада, т.е. коммутатору с номером $(jc_i + \kappa)$ при сплошной нумерации коммутаторов в каскаде ($i + I$) (как на рис.3). Нумерация разрядов в ячейке производится справа налево.

В процессе поиска свободной линии в ячейке $B_j(i)$ присваиваются нули разрядам, которые соответствуют коммутаторам через которые не может быть установлено соединение входа с выходом схемы. В начале поиска, как правило, разрядам в ячейках массива $B(i)$ ($i = 1, 2, \ldots, S-1$) присваиваются "единицы". Значения массивов $B(i)$ в начальный момент для схемы рис. 3 представлены в табл. 4. Массив F состоит из S ячеек $F_1, F_2, \ldots, F_S$. В ячейке F_l ($l = 1, 2, \ldots, S$) содержится число коммутаторов на l-ом каскаде, через которые может быть осуществлено соединение заданных входов схемы с заданными выходами схемы. По определению, максимальное значение, которое может принимать F_l, равно a_l ($l = 1, 2, \ldots, S$).

Таблица 4

Массив B (1)

$B_1(1)$	1	1	
$B_2(1)$	1	1	
$B_3(1)$	1	1	
$B_4(1)$	1	1	
$B_5(1)$	1	1	
$B_6(1)$	1	1	

$\frac{a_1}{b_1} = \frac{a_1}{c_1} = 6$, $c_1 = 2$

Массив B (2)

$B_1(2)$	1	1	1	1	$\frac{a_2}{b_2} = \frac{a_2}{c_2} = 2$
$B_2(2)$	1	1	1	1	

$c_2 = 4$

Массив B (3)

$B_1(3)$	1	1	1	1	1	1	1	1

$\frac{a_3}{b_3} = 1$, $c_3 = 8$

Массив B (4)

$B_1(4)$	1	1	1	1	1	1	$\frac{a_4}{b_4} = 2$
$B_2(4)$	1	1	1	1	1	1	

$c_4 = 6$

Массив B (5)

$B_1(5)$	1	1	
$B_2(5)$	1	1	
$B_3(5)$	1	1	
$B_4(5)$	1	1	
$B_5(5)$	1	1	
$B_6(5)$	1	1	

$\frac{a_5}{b_5} = 6$, $c_5 = 2$

Рассмотрим теперь наиболее распространенный случай поиска от заданного свободного входа к заданному свободному выходу. Пусть вход принадлежит k_1 коммутатору I каскада,

а выход принадлежит κ_s коммутатору последнего S-го каскада $1 \le \kappa_s \le a_s$; $1 \le \kappa_s \le a_s$. Для осуществления поиска свободной линии предварительно присваиваем начальные значения ячейкам массива B и F в порядке $B(s-1)$, $B(s-2)$, ..., $B(1)$ и F . Номер группы ПЛ между $(s-1)$ и s каскадами, к которому принадлежит коммутатор κ_s равен j_{s-1} , где

$$j_{s-1} = \left[\frac{\kappa_s - 1}{c_{s-1}}\right] + 1 , \quad \left(1 \le j_{s-1} \le \frac{a_s}{c_{s-1}}\right)$$

Номер коммутатора κ_s в данной группе равен i_s , где $i_s = \kappa_s - (j_{s-1} - 1)c_{s-1}$; $(1 \le i_s \le c_{s-1})$. Теперь в массиве $B(s-1)$ в ячейке $B_{j_{s-1}}(s-1)$ засылаем "единицу" в i_s разряд, а остальные разряды ячейки принимают значение "нуль". Остальные ячейки массива $B(s-1)$ не используются. Так как доступные выходу из коммутатора κ_s коммутаторы $(s-1)$ каскада принадлежат j_{s-1} группе ПЛ между $(s-1)$ и s каскадами, то их номера принимают значения κ_{s-1} из множества $\{(j_{s-1} - 1)b_{s-1} + 1, (j_{s-1} - 1)b_{s-1} + 2, \ldots, j_{s-1} \cdot b_{s-1}\}$. Причем $\min \kappa_{s-1} = \kappa'_{s-1} = (j_{s-1} - 1)b_{s-1} + 1$; $\max \kappa_{s-1} = \kappa''_{s-1} = j_{s-1} \cdot b_{s-1}$. Определяем, к каким группам ПЛ между $(s-2)$ и $(s-1)$ каскадами принадлежат коммутаторы κ'_{s-1} и κ''_{s-1} .

$$j'_{s-2} = \left[\frac{\kappa'_{s-1} - 1}{c_{s-2}}\right] + 1 ; \left(1 \le j'_{s-2} \le \frac{a_{s-1}}{c_{s-2}}\right);$$

$$j''_{s-2} = \left[\frac{\kappa''_{s-1} - 1}{c_{s-2}}\right] + 1 ; \left(1 \le j''_{s-2} \le \frac{a_{s-1}}{c_{s-2}}\right);$$

Номера коммутаторов κ'_{s-1} и κ''_{s-1} в группах соответственно равны $i'_{s-1} = \kappa'_{s-1} - (j'_{s-2} - 1)c_{s-2}$; $(1 \le i'_{s-1} \le c_{s-2})$; $i''_{s-1} = \kappa''_{s-1} - (j''_{s-2} - 1)c_{s-2}$; $(1 \le i''_{s-1} \le c_{s-2})$. Если коммутаторы κ'_{s-1} и κ''_{s-1} принадлежат к одной группе, т.е. $j'_{s-2} = j''_{s-2} = j_{s-2}$, то в ячейке $B_{j_{s-2}}(s-2)$ массива $B(s-2)$ засылаем "единицу" в разряды, начиная с i'_{s-1} по i''_{s-1} включая, остальные разряды ячейки принимают значение "нуль". Другие ячейки массива $B(s-2)$ не используются.

В случае, если коммутаторы κ'_{s-1} и κ''_{s-1} принадлежат к разным группам, т.е. $j'_{s-2} \ne j''_{s-2}$, то засылаем "единицу" в разряды начиная с i'_{s-1} по c_{s-2} ячейки $B_{j'_{s-2}}(s-2)$, с I по c_{s-2} в ячейки $B_{j'_{s-2}+1}(s-2), B_{j'_{s-2}+2}(s-2), \ldots, B_{j''_{s-2}-1}(s-2)$ и с I по i''_{s-1} в ячейку $B_{j''_{s-2}}(s-2)$. Остальные разряды рас-

смотренных ячеек принимают значение "нуль". Остальные ячейки массива $B(s-2)$ не используются.

Мы определили, что доступные выходам из коммутатора κ_{i_0} коммутаторы $(s-2)$ каскада принадлежат группам $j'_{s-2}, j'_{s-2}+1, \dots, j''_{s-2}$ ПЛ между $(s-2)$ и $(s-1)$ каскадами и их номера принимают значения $\kappa_{i_{s-2}}$ из множества

$$\{(j'_{s-2}-1)b_{s-2}+1, (j'_{s-2}-1)b_{s-2}+2, \dots, j''_{s-2}b_{s-2}\}$$

Очевидно, что $\min \kappa_{i_{s-2}} = \kappa'_{i_{s-2}} = (j'_{s-2}-1)b_{s-2}+1$; $\max \kappa_{i_{s-2}} = \kappa''_{i_{s-2}} = j''_{s-2}b_{s-2}$. Рассмотрение заканчивается, если $\kappa'_{i_{s-2}}=1$, а $\kappa''_{i_{s-2}}=a_{s-2}$. Если указанные равенства не выполняются, то продолжаем аналогичное рассмотрение с коммутаторами $(s-3)$ каскада и т.д. до тех пор, пока у каскада f $\kappa'_{i_f}=1$, а $\kappa''_{i_f}=a_f (f=1,\dots,s-1)$. Остальные ячейки массивов $B(1)$, $B(2)$, ..., $B(f-1)$ во всех разрядах принимают значения "единиц". Присваиваем начальные значения ячейкам массива F. В случае заданного коммутатора κ_1 и κ_4, тем самым, задано, что $F_1=1$, $F_s=1$. Начальные значения F_l $(l=2,3,\dots,s-1)$ вычисляются по формуле $F_l=\kappa''_l-\kappa'_l+1$. Значения κ'_l и κ''_l $(l=f+1,f+2,\dots,s-1)$ определены при присвоении начальных значений массивам B. Значения κ'_l и κ''_l $(l=2,3,\dots,t-1)$ вычисляются рекуррентно по формулам $\kappa'_l = c_{l-1}(j'_{l-1}-1)+1$; $\kappa''_l = c_{l-1}j''_{l-1}$;

$$j'_{l-1} = \left[\frac{\kappa'_{l-1}-1}{b_{l-1}}\right]+1; \quad j''_{l-1} = \left[\frac{\kappa''_{l-1}-1}{b_{l-1}}\right]+1;$$

с начальным значением $\kappa'_1=\kappa''_1=\kappa_1$. Значение t определяется из рекуррентного соотношения при выполнении $\kappa'_t=1; \kappa''_t=a_t$. Величины F_l $(l=t,t+1,\dots,f)$, если они имеются, принимают максимальные значения $F_l=a_l$ $(l=t,t+1,\dots,f)$. После предварительного присваивания начальных значений ячейкам массивов $B(s-1), B(s-2),\dots, B(f)$ и пересылки для использования полученных значений $j_1, j_{s-1}, j'_{s-2}, j''_{s-2}, \dots, j'_f, j''_f$ переходим к поиску свободной линии между коммутатором κ_1 первого каскада и коммутатором κ_4 последнего каскада.

По номеру коммутатора κ_1 первого каскада определяем

номер группы ПЛ между первым и вторым каскадами j_1, где

$$j_1 = \left[\frac{\kappa_1 - 1}{b_1}\right] + 1.$$

Затем производим логическое поразрядное умножение ячеек $A_{\kappa_1}(1)$ и $B_{j_1}(1)$ с засылкой результата в $R(1)$. В ячейке $R(1)$ определяем номер разряда i_1, который принимает значение "единица", (если таких разрядов несколько, то задается алгорити выбора, например, с наименьшим номером). Если разряда со значением "единица" не имеется, то вызов не может получить соединения. Номер группы j_1 и номер коммутатора в группе второго каскада i_1 определят выбор коммутатора второго каскада с номером $\kappa_1 = (j_1 - 1)c_1 + i_1$. По номеру коммутатора κ_1 второго каскада определяем номер группы ПЛ между вторым и третьим каскадами j_2, где

$$j_2 = \left[\frac{\kappa_2 - 1}{b_2}\right] + 1.$$

Затем производим логическое поразрядное умножение ячеек $A_{\kappa_2}(2)$ и $B_{j_2}(2)$ с засылкой результата в $R(2)$. В ячейке $R(2)$ определяем номер разряда i_2, который принимает значение "единица" (задается алгорити выбора, если таких несколько). Если разряда со значением "единица" не имеется, то в разряд i_2 ячейки $B_{j_2}(2)$ высылаем значение "нуль" и от ячейки F_2 вычитаем единицу. Если $F_2 > 0$, то возобновляем поиск ПЛ между первым и вторым каскадами (нового i_1). Если $F_2 = 0$, то вызов не может получить соединения. Номер группы j_2 и номер коммутатора в группе третьего каскада i_3 определят выбор коммутатора третьего каскада с номером $\kappa_3 = (j_2 - 1)c_2 + i_3$.

Дальнейшие действия совершенно аналогичны между коммутаторами любого каскада, поэтому напишем для ℓ-го поиска ПЛ между ℓ и $(\ell+1)$ каскадами. Пусть $\kappa_\ell = (j_{\ell-1} - 1)c_{\ell-1} + i_\ell$. По номеру коммутатора κ_ℓ ℓ-го каскада определяем номер группы ПЛ между ℓ и $(\ell+1)$ каскадами j_ℓ, где

$$j_\ell = \left[\frac{\kappa_\ell - 1}{b_\ell}\right] + 1.$$

затем производим логическое поразрядное умножение ячеек $A_{\kappa_\ell}(\ell)$ и $B_{j_\ell}(\ell)$ с засылкой результата в $R(\ell)$. В ячейке $R(\ell)$ определяем номер разряда $i_{\ell-1}$, который принимает значение "единица" (задается алгоритм выбора, если таких несколько). Если разряда со значением единица не имеется, то в разряд i_ℓ ячейки $B_{j_{\ell-1}}(\ell-1)$ засылаем значение "нуль" и от ячейки F_ℓ вычитаем единицу. Если $F_\ell > 0$, то возобновляем поиски соединения ПЛ между $(\ell-1)$ и ℓ каскадами (нового i_ℓ). Если $F_\ell = 0$, то вызов не может получить соединения.

Если соединение между коммутаторами κ_1 первого каскада и κ_s s-го каскада возможно, то в результате описанного алгоритма найдем соединение, определяемое числами: $\kappa_1, i_2, \kappa_2, i_3, \ldots, \kappa_{s-1}, i_s, \kappa_s$. Для установления соединения (занятия линии) необходимо заслать "нуль": в разряд i_2 ячейки $A_{\kappa_1}(1)$, в разряд i_3 ячейки $A_{\kappa_2}(2)$, ..., в разряд i_s ячейки $A_{\kappa_{s-1}}(s-1)$. Кроме того, отметим, что число занятых входов в коммутаторе κ_1 первого каскада и выходов в коммутаторе κ_s -го каскада увеличилось на единицу. Это производится прибавлением единицы к ячейкам $A_{\kappa_1}(0)$ и $A_{\kappa_s}(s)$. Значения $\kappa_1, i_2, \kappa_2, i_3, \ldots, \kappa_{s-1}, i_s, \kappa_s$ запоминаются вызовом и используются при освобождении линии. Освобождение линии производится засылкой "единицы" в разряд i_ℓ ячейки $A_{\kappa_{\ell-1}}(\ell-1)$ для всех $\ell = 2, 3, \ldots, s$, и вычитанием "единицы" у ячеек $A_{\kappa_1}(0)$ и $A_{\kappa_s}(s)$. Заметим, что можно запомнить только значение $\kappa_1, \kappa_2, \ldots \kappa_s$, а значение $i_2, i_3, \ldots i_s$ вычислить по формуле

$$i_\ell = \kappa_\ell - (j_{\ell-1} - 1)c_{\ell-1}, \quad \text{где } j_{\ell-1} = \left[\frac{\kappa_\ell - 1}{c_{\ell-1}}\right] + 1, \ (\ell = 2, \ldots, s).$$

Для иллюстрации рассмотрим осуществление поиска свободной линии от четвертого коммутатора первого каскада ($\kappa_1 = 4$) к третьему коммутатору шестого каскада ($\kappa_s = 3$) в полностью свободном состоянии схемы рис.3 (отображается массивами табл.3). Присвоим начальные значения ячейкам массивов B. Вычисляем $j_2 = \left[\frac{3-1}{2}\right] + 1 = 2$; $i_2 = 3 - (2-1)2 = 3 - 2 = 1$.

Следовательно, $B_2(5)$ [0|1] .

Вычисляем последовательно далее

$\kappa_5' = (2-1)2+1 = 3$; $\kappa_5'' = 2 \cdot 2 = 4$; $j_4' = \left[\frac{3-1}{6}\right]+1 = 1$;

$j_4'' = \left[\frac{4-1}{6}\right]+1 = 1$; $i_5' = 3-(1-1)6 = 3$; $i_5'' = 4-(1-1)6 = 4$;

$B_1(4)$ [0|0|1|1|0|0] $\kappa_4' = (1-1)4+1 = 1$; $\kappa_4'' = 1 \cdot 4 = 4$;

$j_3' = \left[\frac{1-1}{8}\right]+1 = 1$; $j_3'' = \left[\frac{4-1}{8}\right]+1 = 1$; $i_4' = 1-(1-1)8 = 1$; $i_4'' = 4-(1-1)8 = 4$;

$B_1(3)$ [0|0|0|0|1|1|1|1] ; $\kappa_3' = (1-1)2+1 = 1$; $\kappa_3'' = 1 \cdot 8 = 8$.

Рассмотрение заканчивается, так как $\kappa_3' = 1, \kappa_3'' = 8 = a_3$ и, следовательно, $j = 3$ и остальные ячейки массивов $B(1)$, $B(2)$ принимают соответственно значения:

$B(1)$ [1|1] ; $B(2)$ [1|1|1|1] ;

Определяем значения ячеек F_1 , F_2 , F_3 , F_4 , F_5 , F_6 .

Так как коммутаторы $\kappa_1 = 4$, $\kappa_6 = 3$ заданы, то $F_1 = 1$, $F_6 = 1$.

Значения F_4 , F_5 вычисляются с применением полученных чисел $j = 3$, $\kappa_5' = 3$, $\kappa_5'' = 4$, $\kappa_4' = 1$, $\kappa_4'' = 4$. Получаем $F_5 = 4-3+1 = 2$, $F_4 = 4-1+1 = 4$. Значения F_2 и F_3 определяются по рекуррентным формулам с использованием значения $\kappa_1' = \kappa_1'' = \kappa_1 = 4$;

$j_2' = \left[\frac{4-1}{2}\right]+1 = 2$; $j_2'' = 2$; $\kappa_2' = 2 \cdot 1+1 = 3$; $\kappa_2'' = 2 \cdot 2 = 4$; $j_2' = \left[\frac{3-1}{6}\right]+1 = 1$;

$j_2'' = \left[\frac{4-1}{6}\right]+1 = 1$; $\kappa_3' = 4 \cdot 0+1 = 1$; $\kappa_3'' = 4 \cdot 1 = 4$; $j_3' = \left[\frac{1-1}{8}\right]+1 = 1$;

$j_3'' = \left[\frac{4-1}{8}\right]+1 = 1$; $\kappa_4' = 8(1-1)+1 = 1$; $\kappa_4'' = 8 \cdot 1 = 8$;

Получили, что $t = 4$, так как $\kappa_4' = 1$ и $\kappa_4'' = a_4 = 8$. Следовательно, $F_2 = 4-3+1 = 2$; $F_3 = 4-1+1 = 4$.

Теперь переходим к поиску свободной линии между коммутаторами $\kappa_1 = 4$ и $\kappa_6 = 3$. Вычисляем

$i_1 = \left[\frac{4-1}{2}\right]+1 = 2$; $R(1) = A_4(1) \wedge B_2(1)$; $R(1)$ [1|1] .

Из ячейки $R(1)$ выбираем первый разряд со значением "единица", получаем $i_2 = 1$. Далее вычисляем $\kappa_2 = (2-1)2+1 = 3$;

$i_2 = \left[\frac{3-1}{6}\right]+1 = 1$; $R(2) = A_3(2) \wedge B_1(2)$; $R(2)$ [0|1|0|1] .

Из ячейки $R(2)$ выбираем первый разряд со значением "единица", поэтому $i_3 = 1$. Далее вычисляем $\kappa_3 = (1-1)4+1 = 1$;

$i_3 = \left[\frac{1-1}{8}\right]+1 = 1$; $R(3) = A_1(3) \wedge B_1(3)$; $R(3)$ [0|0|0|0|0|0|1] .

Из ячейки $R(3)$ получаем $i_4 = 1$. Далее вычисляем

$k_4 = (1-1)8+1 = 1$; $j_4 = \left[\frac{1-1}{4}\right]+1 = 1$;

$R(4) = A_1(4) \wedge B_1(4)$; $R(4)$ [0|0|0|1|0|0].

Из ячейки $R(4)$ получаем $i_5 = 3$. Далее вычисляем

$k_5 = (1-1)6+3 = 3$; $j_5 = \left[\frac{3-1}{2}\right]+1 = 2$; $R(5) = A_2(5) \wedge B_2(5)$;

$R(5)$ [0|1],

Из ячейки $R(5)$ получаем $i_6 = 1$. Для проверки вычислим k_6 .

$k_6 = (2-1)2+1 = 3$. Найденное соединение определяется следующими значениями: $k_1 = 4$, $i_2 = 1$, $k_2 = 3$, $i_3 = 1$, $k_3 = 1$, $i_4 = 1$, $k_4 = 1$, $i_5 = 3$, $k_5 = 3$; $i_6 = 1$, $k_6 = 3$. В результате занятия найденной линии (соединения) изменяемые ячейки из таблицы 3 принимают следующие значения

$A_4(1)$ [1|0]

$A_3(2)$ [0|1|0|0]

$A_1(3)$ [0|0|0|1|0|0|0|0]

$A_1(4)$ [0|1|0|0|0|1]

$A_3(5)$ [1|0]

$A_4(0)$ [1]

$A_3(6)$ [1]

На рис. 3 данное соединение отмечено жирными линиями.

Заметим, что при данном рассмотрении мы не различаем, какие входы в коммутатор первого каскада и какие выходы из коммутатора последнего каскада заняты (такое представление называется числовым). В случаях, когда необходимо такое различие, изменим содержание массивов $A(0)$ и $A(5)$, которое назовем поразрядным. Массив $A(0)$ для поразрядного представления состоит из $a_0 = a_4$ ячеек. В ячейке используется n_4 двоичных разрядов, κ -ый двоичный разряд ($\kappa = 1,2,\dots,n_4$) в ячейке $A_j(0)$ ($j = 1,2,\dots,a_0$) принимает

значение "нуль", если соответствующий вход j -го коммутатора занят или отсутствует, а значение "единица" - если свободен. Массив $A(s)$ состоит из a_s ячеек. В ячейке используется m_s двоичных разрядов, κ -ый двоичный разряд $(\kappa = 1, 2, \ldots, m_s)$ в ячейке $A_j(s)$ $(j = 1, 2, \ldots, a_s)$ принимает значение "нуль" если соответствующий выход j -го коммутатора занят или отсутствует, а значение "единица" - если свободен. Для схемы рис.3 в свободном состоянии в таком варианте массивы $A(0)$ и $A(6)$ имеют вид:

Массив $A(0)$

$A_1(0)$	1	1
$A_2(0)$	1	1
$A_3(0)$	1	1
$A_4(0)$	1	1
$A_5(0)$	1	1
$A_6(0)$	1	1
$A_7(0)$	1	1
$A_8(0)$	1	1
$A_9(0)$	1	1
$A_{10}(0)$	1	1
$A_{11}(0)$	1	1
$A_{12}(0)$	1	1

$m_1 = 2$

Массив $A(6)$

$A_1(6)$	1	1
$A_2(6)$	1	1
$A_3(6)$	1	1
$A_4(6)$	1	1
$A_5(6)$	1	1
$A_6(6)$	1	1
$A_7(6)$	1	1
$A_8(6)$	1	1
$A_9(6)$	1	1
$A_{10}(6)$	1	1
$A_{11}(6)$	1	1
$A_{12}(6)$	1	1

$m_6 = 2$

В результате занятия i_1 входа $(i_1 = 1, 2, \ldots, n_1)$ в κ_1 коммутаторе первого каскада и i_{s+1} выхода $(i_{s+1} = 1, 2, \ldots, m_s)$ в κ_s коммутаторе s -го каскада (значения i_1 и i_{s+1} необходимо задавать в этом случае) присваивается "нуль" i_1 разряду ячейки $A_{\kappa_1}(0)$ и i_{s+1} разряду ячейки $A_{\kappa_s}(s)$. Для схемы рис.3 для рассмотренного уже соединения при $i_1 = 1$, $i_2 = 1$ изменяемые ячейки в массивах $A(0)$ и $A(6)$ принимают значения

$A_4(0)$ | 1 | 0 | , $A_1(6)$ | 1 | 0 |

Выбор представления для массива входов схемы $A(0)$ и массива выходов схемы $A(s)$ зависит от конкретных свойств и статистических результатов моделируемой СМО.

3. УПРОЩЕННЫЙ СПОСОБ МОДЕЛИРОВАНИЯ

Наряду с информацией о занятых ЦЛ в многокаскадной схеме при статистическом моделировании для каждого вызова в системе хранится информация о занятой им линии. При использовании описания предыдущего раздела информация о вызове включает значения $\kappa_1, \kappa_2, \kappa_3, \ldots, \kappa_5$ при числовом представлении массивов входов и выходов схемы и значения $i_1, i_{2-4}, \kappa_1, \kappa_2, \ldots, \kappa_5$ при поразрядном представлении массивов входов и выходов схемы. Учитывая, что число вызовов в схеме достаточно велико, информация о вызовах занимает значительную часть памяти ЭВМ, что во многих случаях ограничивает возможность статистического моделирования проектируемых схем АТС. Имеется возможность не хранить информацию о вызове, но при этом результаты статистического моделирования будут менее точными. При таком подходе вызов, который должен покинуть систему, не знает занимаемую им линию (значения $\kappa_1, \kappa_2, \ldots, \kappa_5$ при числовом представлении массивов $A(0), A(5)$) и тем самым неизвестно, как осуществить освобождение линии (осуществить соответствующие изменения в массивах $A(i)$ $i=0,1,\ldots,5$). Предлагается следующий алгоритм определения освобождаемой вызовом линии (значений $\kappa_1, \kappa_2, \ldots, \kappa_5$ при числовом представлении массивов $A(0), A(5)$). Выбирается значение номера коммутатора κ_1 первого каскада с вероятностью

$$p_{\kappa_1} = \frac{A_{\kappa_1}(0)}{\sum_{j=1}^{a_1} A_j(0)}$$

Затем из ячейки $A_{\kappa_1}(1)$ устанавливаем, какие выходы в коммутаторе κ_1 первого каскада заняты. Число занятых выходов в κ_i коммутаторе i-го каскада ($i=1,2,\ldots,5$) обозначим через δ_i. Опишем определение значения номера κ_{i+1} коммутатора $(i+1)$ каскада, через который будет проходить освобождаемая линия, если уже известно, что линия проходит через

коммутатор ℓ -го каскада с номером κ_ℓ $(\ell = 1,2,\ldots,s-1)$. Для этого из ячейки $A_{\kappa_\ell}(\ell)$ массива $A(\ell)$ $(\ell = 1,2,\ldots,s-1)$ определяем номера и число занятых выходов δ_ℓ в коммутаторе κ_ℓ (для $\ell = I$ число $\delta_1 = A_{\kappa_1}(0)$). Для освобождаемой линии выбираем равновероятно один из δ_ℓ занятых выходов данного коммутатора κ_ℓ ℓ -го каскада $(\ell = 1,2,\ldots,s-1)$. Пусть данный выход соответствует в ячейке $A_{\kappa_\ell}(\ell)$ двоичному разряду с номером $i_{\ell+1}$, что в свою очередь соответствует коммутатору $(\ell+1)$ -го каскада с номером $\kappa_{\ell+1}$, где

$$\kappa_{\ell+1} = i_{\ell+1} + (j_\ell - 1)c_\ell; \quad j_\ell = \left[\frac{\kappa_\ell - 1}{b_\ell}\right] + 1.$$

Рекуррентным применением описанной процедуры определяем значения $i_1, \kappa_1, i_2, \kappa_2, \ldots, i_s, \kappa_s$. Теперь по найденным значениям $\kappa_1, i_1, \kappa_2, i_2, \ldots, \kappa_{s-1}, i_s, \kappa_s$ осуществляем освобождение линии в соответствии с описанным ранее алгоритмом и вызов покидает систему. Отличие результатов статистического моделирования происходит по причине несовпадения чисел $\kappa_1, \kappa_2, \ldots, \kappa_s$ занимаемой и освобождаемой линии одного вызова. Однако это несовпадение несущественно, так как равновероятный выбор освобождаемого выхода в коммутаторе обеспечивает хорошее перемешивание освобождаемых линий. Незначительное отличие результатов статистического моделирования по упрощенному способу по сравнению с обычным способом хранения информации занятой линии подтверждается результатами моделирования двухкаскадной схемы. Оценки потерь, полученные обоими способами моделирования при поступлении простейшего потока и экспоненциального закона обслуживания для двухкаскадной схемы рис.4 с потерями при наличии двух направлений приведены в таблице 5. В таблице 5 использованы обозначения: Λ - нагрузка на схему, p_i - вероятность поступления вызова на i - ое направление $(i = I, 2)$, π_i - вероятность потерь в i - ом направлении $(i = I, 2)$, π - вероятность потерь в схеме, W - общее число поступивших вызовов при статистическом моделировании.

Таблица 5

Вероятность потерь двухкаскадной схемы

Поиск линии	λ	p_1	p_2	w	При полной информации			При неполной информации		
					π	π_1	π_2	π	π_1	π_2
Слу-	5	0,3	0,7	10^4	0,I26	0,082	0,I45	0,I28	0,092	0,I44
чай-	I0	0,3	0,7	10^4	0,364	0,273	0,403	0,365	0,27I	0,405
ный	5	0,5	0,5	10^4	0,II3	0,I20	0,I06	0,II6	0,I2I	0,III
	I0	0,5	0,5	10^4	0,348	0,360	0,337	0,348	0,350	0,346
Упо-	5	0,3	0,7	10^4	0,I28	0,08I	0,I48	0,I27	0,084	0,I46
рядо-	I0	0,3	0,7	10^4	0,364	0,269	0,405	0,370	0,262	0,4I7
чен-	5	0,5	0,5	10^4	0,II3	0,II7	0,I08	0,II3	0,III	0,II4
ный	I0	0,5	0,5	10^4	0,35I	0,348	0,355	0,35I	0,354	0,348

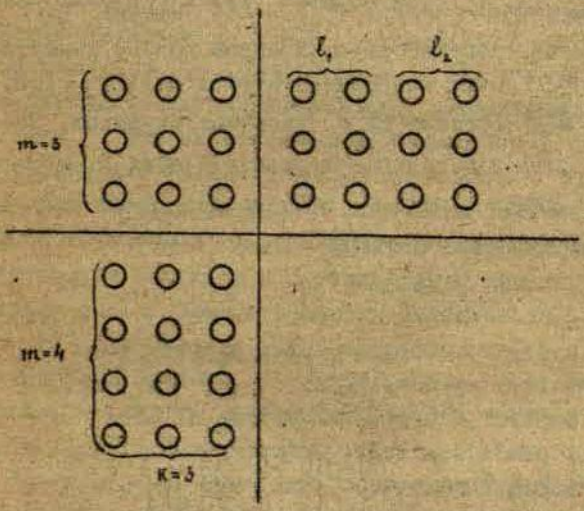


Рис.4. Пример 2-каскадной схемы

Описанный алгоритм определения значений $\kappa_1, \kappa_2, \ldots, \kappa_s$ для освобождаемой вызовом линии с применением массивов $A(i)$ ($i = 0, I, \ldots, s$) возможен для полных регулярных многокаскадных схем, т.е. схем, у которых в состоянии, когда все линии свободны, ячейки массива $A(i)$ представляют последовательность c_i "единиц" ($i = I, 2, \ldots, s-I$). Регулярная схема рис. 3 не является полной, так как указанному требованию не удовлетворяют массивы A(2), A(3), A(4). В произвольной регулярной схеме в ячейке массива $A(i)$ ($i = I, 2, \ldots, s-I$) разряд принимает значение "нуль" не только когда соответствующий выход коммутатора занят, но также когда выход отсутствует. Для определения значений $\kappa_1, \kappa_2, \ldots, \kappa_s$ освобождаемой линии в любой регулярной схеме необходимо дополнить массивами $D(i)$ ($i = I, 2, \ldots, s-I$). Массив $D(i)$ состоит из b_i ячеек, соответствующих одной группе коммутаторов ПЛ между i и $i+I$ каскадами. Ячейка $D_j(i)$ ($j = I, 2, \ldots, b_i$) массива $B(i)$ состоит из c_i двоичных разрядов. Содержание ячейки $D_j(i)$ идентично содержанию ячейки $A_j(i)$, когда в схеме все линии свободны. Массивы $D(i)$ ($i = I, 2, \ldots, s-I$) для схемы рис. 3 представлены в таблице 6.

Таблица 6

Массив D (I)

$D_1(1)$	I	I
$D_2(1)$	I	I

$b_1 = 2$

$c_1 = 2$

Массив D (2)

$D_1(2)$	0	I	0	I
$D_2(2)$	I	0	I	0
$D_3(2)$	0	I	0	I
$D_4(2)$	I	0	I	0
$D_5(2)$	0	I	0	I
$D_6(2)$	I	0	I	0

$b_2 = 6$

$c_2 = 4$

Массив D (3)

$D_1(3)$	0	0	0	I	0	0	0	I
$D_2(3)$	0	0	I	0	0	0	I	0
$D_3(3)$	0	I	0	0	0	I	0	0
$D_4(3)$	I	0	0	0	I	0	0	0
$D_5(3)$	0	0	0	I	0	0	0	I
$D_6(3)$	0	0	I	0	0	0	I	0
$D_7(3)$	0	I	0	0	0	I	0	0
$D_8(3)$	I	0	0	0	I	0	0	0

$\delta_3 = 8$

$C_3 = 8$

Массив D (4)

$D_1(4)$	0	I	0	I	0	I
$D_2(4)$	I	0	I	0	I	0
$D_3(4)$	0	I	0	I	0	I
$D_4(4)$	I	0	I	0	I	0

$\delta_4 = 4$

$C_4 = 6$

Массив D (5)

$D_1(5)$	I	I
$D_2(5)$	I	I

$\delta_5 = 2$

$C_5 = 2$

Изменение описанного алгоритма поиска освобождаемой линии заключается только в том, что для определения числа занятых выходов δ_ℓ в κ_ℓ коммутаторе ℓ-го каскада и определения значения $i_{\ell+1}$ и $\kappa_{\ell+1}$ ($\ell = I, 2, \ldots, s-I$) используется не ячейка $A_{\kappa_\ell}(\ell)$, а ячейка R, где $R = A_{\kappa_\ell}(\ell) \bmod_{\ell} D_\ell(\ell)$. В ячейке R двоичные разряды, соответствующие занятым выходам, принимают значение "единица", а свободные и отсутствующие выходы принимают значения и "нуль". Равновероятно выберем разряд со значением "единица", номер его обозначим через $i_{\ell+1}$. Далее реализация алгоритма соответствует описанию в случае полной регулярной схемы. Возможно также уменьшить объемы массивов $D(i)$ ($i = I, 2, \ldots, s-I$). Для массива $D(i)$ достаточно взять первые d_i ячеек, которые в массиве $D(i)$ в дальнейшем повторяются целые $\frac{\delta_i}{d_i}$ раз ($i = I, 2, \ldots, s-I$). Для рис.3 в таблице 6 для массивов $D(1)$, $D(2)$, $D(3)$, $D(4)$, $D(5)$ соответственно $d_1 = I$, $d_2 = 2$, $d_3 = 4$, $d_4 = 2$, $d_5 = I$. При таком определении массивов $D(i)$ ($i = I, 2, \ldots, s-I$) ячейка R определяется

$R = A_{\kappa_\ell}(\ell) \bmod_{\ell} D_\ell(\ell)$, где

$z = i_\ell - \left[\frac{i_\ell - 1}{d_\ell}\right] d_\ell$; ($\ell = I, 2, \ldots, s-I$).

Отметим, что применение упрощенного способа моделирования требует меньше памяти ЭВМ для массивов, что дает возможность моделировать схемы с большими параметрами, но

увеличивается (хотя и незначительно) время моделирования, так как при освобождении линии требуется определить значения $k_1, k_2, \ldots, k_s$. Упрощенный способ может быть применен и для моделирования других классов схем.

Литература

I. Лившиц Б.С., Фидлин Я.В., Харкевич А.Д. Теория телефонных и телеграфных сообщений. М., "Связь", 1971, 304 с.

2. Иванова О.Н. Электронная коммутация. М., "Связь", 1971. 296 с.

3. Штермер Х., Белендорф Э., Бининда Н., Бретшнайдер Г., Хоффман Э., Зухландт Г. Теория телетрафика. Перевод с немецкого под ред. Башарина Г.П. М., "Связь", 1971, 320 с.

4. Бусленко Н.П., Голенко Д.И., Соболь И.М., Срагович В.Г., Шрейдер Ю.А. Метод статистических испытаний (метод Монте-Карло). М., Физматгиз, 1962, 332 с.

5. Ермаков С.М. Метод Монте-Карло и смежные вопросы. М., "Наука", 1975, 472 с.

6. Соболь И.М. Численные методы Монте-Карло. М., "Наука", 1973, 312 с.

7. Ионин Г.Л., Розитис Т.Я., Седол Я.Я., Шнепс-Шнеппе М.А. Статистическое моделирование телефонных систем на электронных вычислительных машинах. Латвийский Республиканский институт научно-технической информации и пропаганды, Рига, 1968, 100 с.

8. Шнепс М.А. Численные методы теории телетрафика. М., "Связь", 1974, 232 с.

9. Башарин Г.П., Харкевич А.Д., Шнепс М.А. Массовое обслуживание в телефонии. М., "Наука", 1968, 246 с.

10. Башарин Г.П., Швальб В.П. О моделировании действия коммутационных систем методом Монте-Карло на ЭЦВМ. - "Известия АН СССР." Энергетика и автоматика", 1962, № 3, с.143-153.

11. Вдовин А.А., Швальб В.П. Исследование действия коммутационных схем в режиме группового искания на ЭЦВМ. - "Проблемы передачи информации", вып.II, М., АН СССР, 1962, с.77-85.

I2. Мельникова К.П.Моделирование коммутационной системы, на которую поступает поток с простым последействием.- Сб.трудов ЦНИИС ЛФ, I966, № I7, с.32-43.

I3. Bazlen D., Kampe G., Lotze A. On the influence of hunting mode and link wiring on the loss of link systems. 7th ITC, Stockholm, 1973, 232/1 - 232/12.

I4. Марковиц Г.,Хаусснер Б.,Карр Г. СИМСКРИПТ. Алгоритмический язык для моделирования. М.,"Советское радио", I966, I52 с.

I5. Дал О.,Нигард К. Симула - язык для программирования с дискретными событиями. - В кн.: Алгоритмы и алгоритмические языки,вып.2,М., АН СССР,1967,72 с.

I6. Herscovitch H., Schneider T. GPSS III. An expanded general purpose simulator."IBM Syst.J.",1965,No.3,v.4.

I7. Архипов И.М. Проектирование коммутационных систем с использованием статистического моделирования. Автореферат на соискание ученой.степени кандидата технических наук. М., ИПИ АН СССР, I975, 26 с.

I8. Ионин Г.Л., Седол Я.Я. Программирование и статистическое моделирование на БЭСМ-4. Рига,ЛГУ им.П.Стучки, I976. 204 с.

I9. Ионин Г.Л., Седол Я.Я. Описание и моделирование систем массового обслуживания . Наст.сб., с.44-75.

ОЦЕНКА ПРОПУСКНОЙ СПОСОБНОСТИ ЗВЕНЬЕВЫХ СХЕМ КОММУТАЦИИ

Буйке Б.А.(ВЦ ЛГУ им.П.Стучки)
Осокина Н.Н. (РПИ)

I. ВВЕДЕНИЕ

Для современных сложных систем коммутации определение вероятностных характеристик аналитическими методами является трудоемкой и в большинстве случаев чрезвычайно сложной задачей. Даже используя упрощающие предположения о характере потока вызовов и длительности обслуживания вызова, решение системы уравнений вероятностей состояний для реальных структур представляется невозможным. Тем сложнее изучение систем коммутации в реальных условиях обслуживания поступающих потоков вызовов. В ряде работ рассматриваются точные аналитические методы расчета вероятностных характеристик [I,9, I2], но результаты получены для сравнительно простых коммутационных структур.

Приближенные методы расчета основываются на упрощениях либо относительно структуры схемы, либо относительно процесса установления соединения. Большинство из них предполагает функциональную и статистическую независимость между распределениями вероятностей, описывающими число одновременно занятых промежуточных линий между звеньями и выходами в направлении [8, I0, II, I3, I4, I5, I7, I8, 2I, 22]. Эти методы дают достаточно хорошие результаты в области потерь до 2 %. Ряд методов основан на понятии эффективной доступности, когда многозвеньевая схема сводится к однозвеньевой, имеющей такое же число входов и выходов и пропускающую такую же нагрузку при одинаковом качестве обслуживания [8]. Дальнейшим развитием понятия эффективной доступности являются методы CIRB, CLIG5-A , CLIG5-B , раз-

работанные Лотце [I9, 25] . Эти методы позволяют определить вероятностные характеристики в многозвеньевых схемах коммутации в случае как полнодоступного, так и неполнодоступного включения. Одним из приближенных методов расчета является метод вероятностных графов, предложенный Ли [I6] . Он удобен для предварительной оценки пропускной способности коммутационных схем с целью сравнения различных вариантов группообразования и выбор из них оптимального. Точность результатов зависит от структуры схемы и величины нагрузки. Применение аналитического выражения Ли ограничивается вычислительными трудностями, возникающими в связи со сложностью реальных структур. Приближенные методы расчета не позволяют определить величину ошибки в оценке вероятности потерь. Определение погрешности возможно путем сравнения с результатами статистического моделирования процесса функционирования коммутационных схем на ЭВМ. Этот метод, впервые изложенный в [23,24] получил повсеместное распространение, как наиболее точно отражающий процессы, происходящие в системах коммутации. В СССР работы по статистическому моделированию проводятся Г.П.Башариным и его учениками [2,3] в ИППИ АН СССР, в ВЦ ЛГУ [4,5,6,7] , где получен ряд практически важных результатов. В работах 7-го конгресса по телетрафику получены результаты для большого разнообразия структур, что еще раз подчеркивает распространение этого метода. Но при всех преимуществах метод статистического моделирования требует значительных затрат труда при составлении программ, а также машинного времени на проведение счета. Поэтому представляется целесообразным применить с некоторыми модификациями программу NEASIM [20] для моделирования на ЭВМ вероятностных графов построенных по принципам Ли. На основе [20] нами разработана для ЭВМ БЭСМ-4 программа исследования пропускной способности многозвеньевых систем коммутации. Программа требует незначительного времени по сравнению с временем моделирования идентичной коммутационной схемы и по-

зволяет оценить пропускную способность достаточно широкого класса полнодоступных и неполнодоступных структур, работающих в режиме группового (ГИ), свободного (СИ) и линейного искания (ЛИ) с явными отказами. По данной программе проводились расчеты для оценки пропускной способности 2-х, 3-х, 4-х, 8-и звеньевых систем коммутации интегральной сети связи.

2. КРАТКОЕ ИЗЛОЖЕНИЕ МЕТОДА ЛИ

Как уже было отмечено, программа моделирования вероятностного графа основывается на использовании модели Ли. Вероятностный граф Ли является упрощенной математической моделью реальной коммутационной структуры, где узлам графа соответствуют коммутаторы, а ребрам - промежуточные линии или выходи в направлении. Граф представляет собой всевозможные соединительные пути от заданного входа к требуемым выходам или выходу. Упрощением при описании графа является предположение о независимости занятия промежуточных линий, что не соответствует реальности, т.к. соединительный путь представляет собой последовательную цепочку ребер графа. Но при небольшой вероятности блокировки в разветвленных структурах погрешность результатов удовлетворительна для инженерной оценки вероятности блокировки. Пример 3-звеньевой коммутационной схемы в режиме группового искания и соответствующего ей графа приведен на рис. I и 2 соответственно.

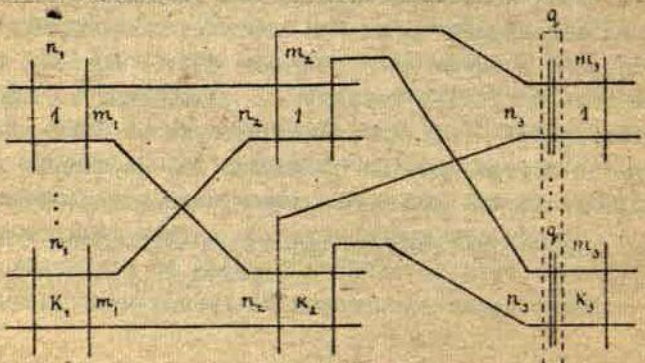


Рис.I. 3-звеньевая схема в режиме ГИ

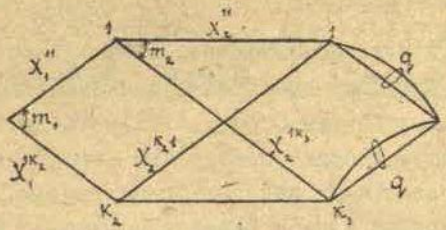


Рис.2. 3-звеньевая схема в режиме ГИ

В модели Ли каждому ребру графа сопоставляется случайная величина X_i^{il}, которая может принимать значение I с вероятностью q_i^{il}, что соответствует свободности промежуточной линии, и значение 0 с вероятностью $p_i^{il} = 1 - q_i^{il}$, что соответствует занятости промежуточной линии. Причем, j соответствует номеру звена, i - номеру коммутатора в этом звене, l - номеру коммутатора в $j+1$ звене. Путь между входом и выходом графа может быть представлен в виде формального произведения величин X_i^{il}, что соответствует последовательно включенным линиям реальной коммутационной схемы. Например, некоторый k - ый соединительный путь, проходящий через I-ый коммутатор I-ого звена, 2-ой коммутатор 2-ого звена, 4-ый коммутатор 3-его звена может быть записан, как $a_k = X_1^{12} X_2^{29} X_3^{44}$. Число возможных путей определяется структурными параметрами коммутационной схемы и для режима ГИ и ЛИ соответственно могут быть определены: $L_{ГИ} = m_1 m_2 m_3 \dots m_{k-1} q_k$; $L_{ЛИ} = m_1 m_2 m_3 \dots m_{k-1}$;
При этом использованы обозначения: k - число звеньев; q_i - число линий в направлении; m_i - число выходов из коммутатора i - ого звена. Согласно выражению Ли, вероятность блокировки определяется:

$$\Pi = 1 - \sum_{n=1}^{L} S_n^* (-1)^{n+1} \quad , \qquad (I)$$

$$S_1^* = a_1 + a_2 + \ldots + a_L ;$$

$$S_2^* = a_1 a_2 + a_1 a_3 + \ldots + a_2 a_3 + \ldots + a_3 a_4 + a_3 a_5 + \ldots$$

$$S_3^* = a_1 a_2 a_3 + a_1 a_2 a_4 + \ldots + a_2 a_3 a_4 + a_2 a_3 a_5 + \ldots$$

. . . . .

$$S_L^* = a_1 a_2 a_3 a_4 \ldots a_L$$

Причем при вычислении a_i необходимо учесть, что

$$\left(X_i^{km}\right)^n = X_i^{km} .$$

Например, для графа на рис.3 вероятность потерь определится.

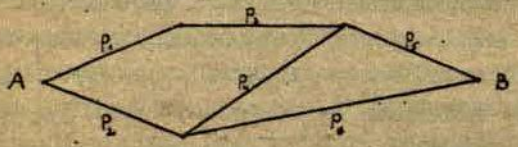


Рис.3. Граф 4-звеньевой схемы коммутации в режиме ЛИ

Положим, что $p_1 = p_2 = p_3 = p_4 = p_5 = p_6 = 0,3$; тогда вероятность свободности ребер $q_1 = q_2 = \ldots = q_6 = 0,7$;
Вероятность свободности путей:

$a_1 = q_1 q_3 q_5 = 0,343$; $a_2 = q_2 q_4 q_6 = 0,343$; $a_3 = q_2 q_6 = 0,49$;

$S_1^* = a_1 + a_2 + a_3 = 1,176$;

$S_2^* = a_1 a_2 + a_1 a_3 + a_2 a_3 = q_1 q_3 q_5 q_2 q_4 + q_1 q_2 q_5 q_2 q_6 + q_2 q_4 q_5 q_6 =$
$= 0,7^5 + 0,7^5 + 0,7^4 = 0,168 + 0,168 + 0,240 = 0,576$;

$S_3^* = a_1 a_2 a_3 = 0,1176$; $\Pi = 1 - 1,176 + 0,576 - 0,1176 = 0,2824$;

Возникающие вычислительные трудности при расчете вероятности блокировки по выражению (I) очевидны в связи с требуемым объемом памяти ЭВМ при исследовании реальных структур.

3. ОПРЕДЕЛЕНИЕ ПОТЕРЬ НА ОСНОВЕ ПРОГРАММЫ

Опишем алгоритм, составленной нами программы на ЭВМ БЭСМ-4, реализованной аналогично программе *NEASIM* [20] на основе метода Ли [16] . Программа оценивает потери в вероятностном графе, отражающем многозвеньевую коммутационную схему. Пример представления коммутационной схемы рис. 4 в режиме линейного искания в виде вероятностного графа приведен на рис.5.

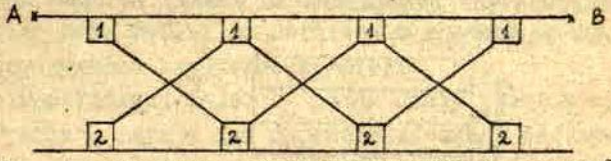


Рис.4. 4-звеньевая схема коммутации в режиме ЛИ

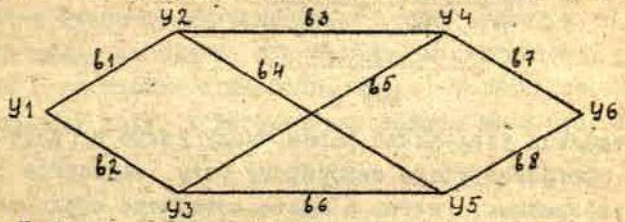


Рис.5. Граф 4-звеньевой схемы коммутации в режиме ЛИ

Представление коммутационной схемы в виде вероятностного графа основано на предположениях: потери для схемы аналогичны потерям между некоторой парой полюсов A - B ; потери π или вероятность того, что нет пути между абонентами A-B $p(A,B)$ в схеме не зависит от времени; вероятность $p(A,B)$ не зависит от абонентов A и B и равен вероятности потерь π ; вероятность занятий любой соединительной линии не зависит от вероятности занятий любой другой соединительной линии; в коммутационной схеме нет соединений между коммутаторами в пределах одного звена.

3.I. Задание структуры графа

Структура графа, т.е. порядок соединения коммутаторов в схеме задается при помощи таблиц: таблица узлов А, таблица узел-линия, таблица соединений, таблица узлов Б, таблица ветвей. Размер таблицы узлов А равен общему числу узлов в графе. В ячейку, соответствующую некоторому узлу, записывается информация об узлах, которые предшествуют данному узлу (рис.6)

у1		
у2	У1	
у3	У1	
у4	У2	У3
у5	У2	У3
у6	У4	У5

Рис.6

Размер таблицы узел-линия равен числу узлов в графе. В ячейку, соответствующую некоторому узлу, заносится следующая информация: метка о месте в таблице соединений, где представлена дальнейшая информация, и число ветвей, исходящих из данного узла.

у1	L1	2
у2	L2	2
у3	L3	2
у4	L4	1
у5	L5	1
у6		

Рис.7

Таблица соединений имеет размер, равный общему числу ветвей графа. В таблице соединений записаны номера ветвей и номер узла в таблице узлов, к которому данная ветвь ведет (рис.8).

L1	β_1	y_2
	β_2	y_3
L2	β_3	y_4
	β_4	y_5
L3	β_5	y_4
	β_6	y_5
L4	β_7	y_6
L5	β_8	y_6

Рис.8

Содержание этих таблиц не меняется в течение всего процесса определения вероятности потерь.

Размер таблицы узлов B равен числу узлов в графе. Данная таблица служит для определения числа блокированных путей для отдельного цикла алгоритма. Содержание данной таблицы в процессе моделирования графа меняется. В начале каждого цикла алгоритма все разряды каждой ячейки данной таблицы заполняются единицами, кроме ячейки, соответствующей первому узлу, в которую засылаются нули (рис.9).

y_1	0	0	...	0
y_2	1	1	...	1
y_3	1	1	...	1
y_4	1	1	...	1
y_5	1	1	...	1
y_6	1	1	...	1

Рис.9

Таблица ветвей содержит K ячеек, где K - число ветвей. В i - той ячейке данной таблицы в каждом разряде записывается единица с вероятностью p_i, соответствующей средней занятости промежуточной линии между коммутаторами двух последовательных звеньев (рис.10).

t_1	0	11010
t_2	1	111011
t_3	0	1010
t_4	0	0011
t_5	1	0110
t_6	1	1101
t_7	0	00111
t_8	0	10110

Рис.IO

3.2. Алгоритм моделирования вероятностного графа

Алгоритм моделирования вероятностного графа включает в себя алгоритм генератора вероятностей и назначения состояний, а также алгоритм поиска пути.

Основной задачей алгоритма генератора вероятностей является получение случайных чисел по заданным значениям нагрузок на ветви. В результате действия данного алгоритма любой разряд ячейки x будет содержать "I" с заданной вероятностью. Содержимое ячейки x пересылается в соответствующую ячейку таблицы ветвей. Алгоритм заканчивает работу после того, как все ячейки таблицы ветвей будут заполнены.

Основной задачей алгоритма поиска является определение для каждого назначения состояний ветвей наличия или отсутствия пути через граф. Для этого осуществляется следующая последовательность действий, которую мы рассмотрим на примере графа рис.5.

I. Установление счетчика попыток на нуль ($i = 0$);
2. Вхождение в таблицу узлов; запоминание y_1 ; выделение соответствующего значения y_i .
3. По y_1 вхождение в таблицу узел-линия и выделение номера метки L_i и числа ветвей, исходящих из данного узла.

4. По метке $L1$ вхождение в таблицу соединения; определение $Y2$; сравнение выделенного значения, т.е. $Y2$ с номером, полученным по пункту I; если они совпадают, тогда выделение значения $\delta 1$; если нет, тогда переход к значению $L1+1$ и повторение п.4.
5. По найденному значению $\delta 1$ вхождение в таблицу ветвей; по определенному значению $Y1$ вхождение в таблицу узлов B ; выполнение $[Y1] \vee [\delta 1]$ по выделенному значению $Y2$ (в табл.узлов B); выполнение $([Y1] \vee [\delta 1]) \wedge [Y2]$ и значение результата в таблицу узлов B в ячейку $Y2$.
6. Вхождение в таблицу узлов, запоминание $Y3$, выделение $Y1$ и далее аналогично.
7. В результате действия этого алгоритма в ячейке $Y6$ таблицы узлов B число единиц будет равно числу блокированных попыток. Суммирование числа единиц.
8. $i = i + 1$; проверки условия $i = N$; N – заданное число попыток; если это условие не выполняется, тогда переход к пункту 9, в противном случае к п.10.
9. Обращение к алгоритму генератора вероятностей и назначения состояний и повторений данного алгоритма.
10. Вычисление вероятностей потерь

$$\pi = \frac{\Sigma \text{ числа единиц}}{45 \cdot N}$$

Для оценки правильности работы программы моделирования вероятностного графа было проведено сравнение результатов счета по программе и результатов, полученных с использованием аналитического выражения Ли для различных структур. Эта оценка показала достаточно близкое совпадение значений вероятности блокировки, полученных обоими методами, что иллюстрируется табл.I.

Выше уже отмечались недостатки вероятностей модели Ли [16, 20] , предполагающей независимость занятия ребер графа. Для ряда звеньевых схем было проведено сравнение результатов, полученных различными методами – приближенными, точными аналитическими, методом статистического моделирования.

Это сравнение показало, что точность результатов по разработанной программе зависит от структуры схемы, величина нагрузки и потерь. В области малых потерь результаты дают заниженное значение, в области больших потерь (десятки процентов) завышенное значение потерь. Тем не менее, для качества обслуживания, обычно используемого в телефонных системах, метод дает достаточную точность результатов, что иллюстрируется табл.2.

Разработанная нами программа моделирования вероятностного графа, является реализацией одного из приближенных методов оценки пропускной способности коммутационных схем, позволяет достаточно эффективно определить вероятность блокировки многозвеньевых схем в режиме группового, линейного и свободного искания. Преимуществом этого метода являются незначительные затраты машинного времени для получения результатов, используемых в инженерных расчетах. Например, для 4-звеньевой схемы при моделировании вероятностного графа требуется t =I5 мин., а при статистическом моделировании t = I час. Кроме того, программа позволяет оценить пропускную способность схемы практически любой конфигурации. Отсюда вытекает целесообразность применения этого метода для предварительной оценки схем группообразования на стадии проектирования.

Разработанная программа использовалась для оценки пропускной способности многозвеньевых структур единой аналого-цифровой сети связи на стадии эскизно-технического проекта и позволила достаточно эффективно определить ряд приемлемых вариантов группообразования, каждый из которых моет быть затем исследован более точными методами, например, методом статистического моделирования.

Таблица I

Расчет вероятности блокировки методом Ли и методом моделирования вероятностного графа.

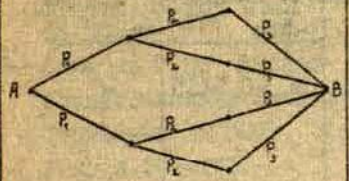
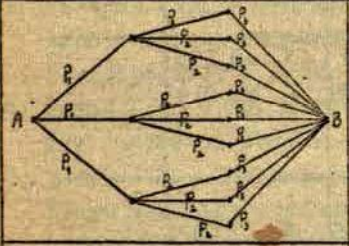
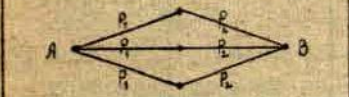
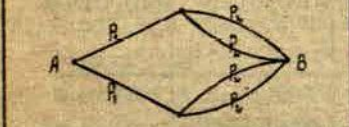
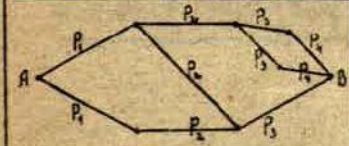
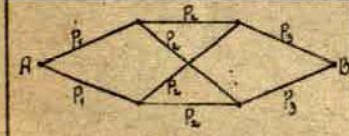
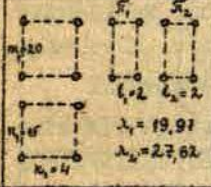
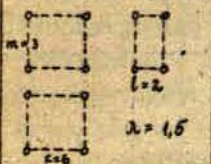
Конфигурация графа	Значения вероятностей	Метод Ли	Моделирование вероятностного графа
	$P_1 = 0,2$ $P_2 = 0,509$ $P_3 = 0,509$	0,0432	0,0440
	$P_1 = 0,I$ $P_2 = 0,05$ $P_3 = 0,05$	0,00I02	0,00I33
	$P_1 = 0,2$ $P_2 = 0,3$	0,085I	0,088I
	$P_1 = 0,2$ $P_2 = 0,I$	0,0432	0,0443
	$P_1 = 0,I$ $P_2 = 0,2$ $P_3 = 0,2$ $P_4 = 0$	0,094	0,095
	$P_1 = P_2 = P_3 = 0,5$ $P_1 = P_2 = P_3 = 0,4$ $P_1 = P_2 = P_3 = 0,2$ $P_1 = P_2 = P_3 = 0,I$	0,63I8 0,434I 0,II59 0,0262	0,5666 0,4077 0,II79 0,027I

Таблица 2

Сравнение методов расчета вероятности блокировки для различных схем коммутации.

№ п.п.	Схема коммутации	Моделирование в ВЦ ЛГУ	Метод Ли	Моделирование вероятностного графа	Аналитический расчет блокировки
I		$\pi_1 = 0,0171$	$\pi_1 = 0,091$	$\pi_1 = 0,0115$	-
2	$\kappa_1 = 10$; $n_1 = 10$; $m_1 = 6$; $\ell_1 = \ell_2 = 2$. $\lambda_1 = 9,515$; $\lambda_2 = 7,506$	$\pi_1 = 0,2144$ $\pi_2 = 0,0625$	$\pi_1 = 0,209$ $\pi_2 = 0,0578$	$\pi_1 = 0,175$ $\pi_2 = 0,0141$	-
3		$\pi = 0,0056$	$\pi = 0,0055$	$\pi = 0,0024$	$\pi = 0,007$
4	$\kappa = 5$; $m = 2$; $\ell = 2$; $\lambda = 0,75$	$\pi = 0,01450$	$\pi = 0,0152$	$\pi = 0,01144$	$\pi = 0,01653$
5	$m = 20$; $n = 10$; $\ell = 1$; $\lambda = 16,83$	$\pi = 0,163$	$\pi = 0,112$	$\pi = 0,2158$	

Литература

I. Башарин Г.П. Об аналитическом определении и методах вычисления вероятности потерь в коммутационных схемах. - "Проблемы передачи информации", вып.9, М., АН СССР, I96I, с.5-47.

2. Башарин Г.П., Швальб В.П. О моделировании действия коммутационных систем методом Монте-Карло на ЭЦВМ. - "Известия АН СССР. Энергетика и автоматика", I962, № 3, с.I43-I53.

3. Башарин Г.П. О статистических оценках вероятности потерь и других характеристик коммутационных схем. - "Электросвязь", I962, № 3, с.6-I2.

4. Ионин Г.Л. и др. Статистическое моделирование телефонных систем на ЭВМ, Рига, ЛатИНТИ, I968. I00 с.

5. Ионин Г.Л., Седол Я.Я. О зависимости пропускной способности неполнодоступных схем с потерями от вида поиска. - Проблемы передачи информации, т.3, вып.I, М.,АН СССР, I967, с.82-85.

6. Ионин Г.Л., Седол Я.Я. Программирование и статистическое моделирование на БЭСМ-4, Рига, ЛГУ им.П.Стучки, I976. I92 с.

7. Шнепс М.А. Численные методы теории телетрафика. М., "Связь", I974. 232 с.

8. Харкевич А.Д. Приближенный метод расчета числа соединительных устройств в АТС координатной системы. - "Электросвязь", I959, № 2, с.55-63.

9. Bretschneider G. Modern concepts concerning single- and two-stage switching arrays as derived from exact loss calculations.-"JTC4", 1964, doc.22. (London).
10. Bininda N., Daisenberger D., Didlaukis M. Verlustberechnung fur Zwischenleitungsanordnungen beliebiger Stufenzahl.-"Nachrichten-techn", 1965, H.11, S.634-636.
11. Bininda N., Wendt W. Die effektive Erreichbarkeit für Abnehmerbünder hinter Zwischenleitungsanordnungen, 1961, Bd.14, H.1, S.40.
12. Elldin A. On equations of state for two-stage link systems.-"Ericsson Techn", 1956, v.12, No.1, p.61-104, (Stockholm).
13. Fortet R. Les fonctions aleatoires en telephonie automatique. Probabilites de perte en selection conjuguee. -"Ann.telecomm.", 1956, v.11, No.7/8, p.85-88.
14. Fortet R., Cancelli B. Probabilites de perte en selection conjuguee.-"Teleteknik", 1957, No.1, p.41-55. (Paris).
15. Pröhl G. Gefahrzeil Zweistufiger Linksysteme in der Fernsprechvermittlungstechnik.-"Nachrichtentechnik", 1962, Bd.12, H.1, S.24-31.
16. Lee C.Y. Analysis of switching networks.-BSTJ, 1955, v.34, American Telephone and Telegraph Company, p.1287-1315.
17. Gall P.Le. Etude du blocage dans des systems de commutation telephonique automatique utilisant commutateurs electroniques du type crossbar.-"Ann.telecomm.", 1956, v.11, No.7/8, p.159-171, No.9, p.180-194, (Paris).
18. Gall P.Le. Methode de calcul de l'encombrement dans les systemes telephoniques automatiques a marquage.- "Ann.telecomm.", 1957, v.12, No.11, p.374-386. (Paris).
19. Lotze A. 3.Bericht über verkehrstheoretische Untersuchungen (CIRB). Stuttgart, Institut für Nachrichtenvermittlung und Datenverarbeitung der TH, 1963.

20. Grantges R., Sinowitz S. NEASIM. A General-purpose computer simulation program for Load-Loss analysis of multistage central office switching networks. -BSTJ, 1963, v.43, American Telephone and Telegraph Company, p.965-1004.

21. Bosse L.G. Some recursive aids for switching network blocking computation.-"JTC4", 1964, doc.21, (London).

22. Jacobaeus C. A study on congestion in link systems.- "Ericsson Techn.", 1950, No.48, p.1-68. (Stolcholm).

23. Neovius G. Artificial traffic trials using digital computer.-"Ericsson Techn.", 1955, v.2, p.1-32. (Stockholm).

24. Wallstrom B. Artificial traffic trials on a two-stage link system using a digital computer.-"Ericsson Techn.", 1958, v.5, (Stockholm).

25. Bazlen D., Kampe G., Lotze A., On the influence of hun ting mode and link wiring on the loss of link systems.-"JTC7", University of Stuttgart Federal Republic of Germany, 1973, p.232/1-232/12.

ИСПОЛЬЗОВАНИЕ РЕЗУЛЬТАТОВ СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ ПРИ ПРОЕКТИРОВАНИИ АТС

Розитис Т.Я.
(Министерство связи СССР)

Качество обслуживания в автоматических телефонных станциях (АТС) определяется многими факторами: надежностью действия оборудования, отказами из-за отсутствия свободных обслуживающих устройств и недостаточной пропускной способности коммутационного поля, возможностью организации обходных путей, наличием эффективной проверки оборудования АТС и др. Из вышеперечисленных факторов особое место занимают потери, вызванные отсутствием свободных обслуживающих устройств и путей в коммутационном поле, ибо от количества оборудования и коммутационных устройств зависят стоимость и эксплуатационные расходы АТС. С ростом количества абонентов одной АТС увеличивается число каскадов в коммутационном поле и в связи с этим, определение состава оборудования существующими методами является затруднительным и неточным.

В настоящее время разработаны методы расчета количества обслуживающих приборов применительно к одноканадным и двухкаскадным коммутационным схемам, характерным для АТС малой емкости. Эти методы расчета учитывают влияние от количества источников нагрузки, распределение продолжительности обслуживания и также способ включения обслуживающих приборов. При помощи этих расчетов составлены различные таблицы для облегчения инженерных расчетов. Необходимо также отметить, что в этих расчетах не учитывается включение обслуживающих приборов в неполнодоступных включениях. Для небольшого количества ($\leqslant 10$) обслуживающих устройств, включенных в неполнодоступную схему, имеется возможность проведения точного аналитического расчета на ЭВМ. При ко-

личестве приборов больше 10 влияние распределения обслуживающих приборов на пропускную способность (т.е. на точность определения потерь в коммутационной системе) можно определить только статистическим моделированием [1].

Коммутационное поле АТС средней емкости построено по многокаскадной схеме. В этих схемах из-за внутренней блокировки возникают дополнительные затруднения при определении величины потерь. Имеются многие методы для инженерных расчетов многокаскадных коммутационных схем, но они, с целью упрощения расчетов, содержат различные допущения и поэтому не обеспечивают необходимую точность. Такие результаты могут быть использованы только на этапе эскизного проекта при выборе вариантов.

Уже продолжительное время в Вычислительном центре ЛГУ им.П.Стучки под руководством канд. физ.-мат.наук Г.Л.Ионина проводятся работы по статистическому моделированию вновь разработанных АТС с целью определения пропускной способности. Статистическому моделированию подвергались все разработанные на заводе ВЭФ координатные АТС. Таким образом, была подтверждена целесообразность разработки АТСК 50/200, на которой по сравнению с прототипом АТСК 40/80 получена экономия по основным коммутационным устройствам (МКС) до 40 %. Статистическим моделированием было подтверждено выполнение технических требований на АТСК 100/300, в этой АТС при помощи очень простого группообразования, кроме экономии многократных координатных соединителей (МКС), были оптимально упрощены (для АТС такого класса) управляющие устройства. Упрощение управляющих устройств АТС 100/300 позволило без дублирования обеспечить требуемую надежность работы АТС [2].

В настоящее время на основе технического задания проводятся разработки автоматических телефонных станций третьего поколения - квазиэлектронных автоматических телефонных станций (КЭ АТС). Техническое задание по группообразо-

вания коммутационного поля квазиэлектронных (КЭ) АТС содержит схемные предложения или исходные данные, необходимые для разработки коммутационного поля (интенсивность нагрузки, допускаемые потери, количество каскадов и др.). На основе технического задания и опыта проектирования разрабатывается частное техническое задание (ЧТЗ) для проведения статистического моделирования с целью исследования предполагаемого группообразования. При разработке ЧТЗ необходимо учесть некоторые упрощения проверяемого модуля по сравнению с реальной станцией. Упрощения уменьшают затраты на моделирование, а также ускоряют получение результатов. Обычно эти упрощения касаются количества направлений внешней связи, распределения времени обслуживания, исследования только некоторых комплектаций АТС и др. Такие упрощения относятся к факторам только незначительно влияющим на результаты (например, количество направлений внешней связи) или не подлежащие более точному математическому описанию (время обслуживания). Некоторые данные для статистического моделирования выдаются дважды: для начала моделирования ориентировочные и после получения первых результатов - уточненные. К таким относятся: интервал интенсивности нагрузки, в котором должно проводиться моделирование, и количество проверяемых точек. Количество отдельных обслуживающих приборов (регистров) иногда также подлежит уточнению. В техническом задании на разработку КЭ АТС предусмотрено проектирование нескольких вариантов АТС, отличающихся интенсивностью предлагаемой нагрузки, начальной и конечной емкостями источников нагрузки. Иногда для решения задания необходима проверка вариантов группообразования, которые отличаются по количеству каскадов. При одинаковом количестве каскадов коммутационные поля отличаются: а) по концентрации интенсивности нагрузки на блоке абонентских линий (БАЛ), которая обычно имеет значение 2:1, 4:1 и 8:1; б) по количеству входов и выходов на блоке соединительных линий, например 32 на 32,

64 на 64 и 128 на 128. Вышеупомянутые варианты также могут быть осуществлены на нескольких разновидностях коммутаторов (8x4, 8x8, 16x8).

Кроме определения основного показателя коммутационного поля - вероятности потерь при определенных интенсивностях нагрузки, моделированием исследуются: 1) распределение вероятности потерь по отдельным каскадам коммутационного поля и видам обслуживающих устройств; 2) отказы по причине занятого абонента; 3) зависимость потерь от способа поиска обслуживающих приборов и промежуточных путей; 4) количество попыток поиска промежуточных путей, гарантирующих выполнение требований технического задания по величине потерь.

Литература

1. Шнепс М.А. Численные методы теории телетрафика. М., "Связь", 1974, 232 с.

2. Ионин Г.Л. и др. Статистическое моделирование телефонных систем на ЭВМ. Рига, ЛатИНТИ, 1968. 100 с.

ОБ ИНЖЕНЕРНОЙ ПОСТАНОВКЕ ЗАДАЧ СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ ТЕЛЕФОННЫХ КОММУТАЦИОННЫХ СИСТЕМ

Болотин В.А.
(ЦНИИС, филиал, г.Ленинград)

Телефонная станция содержит определенное количество технических устройств, через которые устанавливаются телефонные соединения. Расчет количества устройств составляет одну из главных проблем проектирования станций. Мы имеем здесь дело с задачами массового обслуживания, более узко – теории телетрафика, а в технической терминологии – с задачами пропускной способности коммутационных структур, или коммутационных систем.

Решение технической задачи о пропускной способности телефонной коммутационной структуры состоит из нескольких этапов. Сначала по исходному техническому описанию задача формулируется в терминах теории телетрафика, иначе говоря, строится абстрактная /математическая/ модель реального процесса обслуживания. В подавляющем большинстве задач математическая модель не приводит к аналитическому решению и следующим этапом становится статистическое моделирование. Однако далеко не всякая математическая модель оказывается достаточно простой и для моделирования. Так появляются промежуточные этапы – упрощение и изменение абстрактной модели, направленные на то, чтобы приспособить задачу к существующей технике моделирования, в том числе к ограниченным возможностям ЭВМ. Это – первая из двух тем нижеследующих заметок.

Вторая тема касается задач инженера в его непосредственном взаимодействии с математиками, создающими алгоритмы и программы моделирования. Обе стороны стоят перед барьером, разделяющим язык инженера и язык математика. Каждый из этих языков является естественным продуктом сферы его применения. Особенной характеристикой каждого

является не только специальная лексика /терминология/, но и способы описания объектов исследования. Со стороны инженера трудности возникают начиная с построения математической модели задачи. Модель часто бывает несовершенной, ей не достает строгости, однозначной определенности. Трудности переходят затем и на формулирование исходных данных для моделирования и на требования, предъявляемые к результатам моделирования. В дальнейшем появляется и задача приема готовой программы. Прием пересекается с окончательной отладкой программы, инженерное понимание процесса обслуживания может эффективно облегчать отладку, но этому часто препятствует упомянутый барьер.

На протяжении 15-летнего сотрудничества инженеров ЛОНИИС и математиков Вычислительного центра Латвийского государственного университета обе стороны, работая в тесном взаимодействии, набирали опыт и совершенствовали свои методы решения общей задачи. В ЛОНИИС работы велись под руководством Б.С.Лившица. Первые и потому самые трудные контакты с ВЦ ЛГУ осуществлял К.П.Мельников; затем их продолжил автор. Методы и программы моделирования разрабатывались в отделе исследований операций ВЦ ЛГУ. Моделирование проводилось на БЭСМ-2 и, больше всего, на БЭСМ-4.

Обращаясь к возможностям статистического моделирования абстрактных моделей разной сложности, поясним главные трудности. Мы уже говорили, что подавляющее большинство задач по пропускной способности коммутационных систем требует обращения к статистическому моделированию и что, сверх того, задачи еще приходится приспосабливать к технике программирования и к ограниченным возможностям доступных ЭВМ.

Программирование больших задач требует многих месяцев работы одного-двух программистов, а попытки разделять задачу между большим числом программистов не приносят заметного успеха. Сравнительно небольшой объем оперативной памяти ЭВМ заставляет ограничивать параметры

моделируемых структур. Из-за небольшой скорости работы ЭВМ одна точка /несколько десятков тысяч вызовов/ моделируется в течение одного, двух и больше часов, а, как показал опыт, организация массового эксперимента становится очень трудной, часто практически невозможной, если на одну точку уходит больше 2-3 часов работы ЭВМ.

Применение языковых средств для моделирования телефонных процессов обслуживания на советских ЭВМ еще не развито. Такие алгоритмические языки, как АЛГОЛ, ФОРТРАН, даже в совокупности с языком ассемблера пригодны лишь для небольших задач. Специализированных языков высокого уровня нет, да они и не принесли бы большой пользы на ЭВМ относительно небольшой мощности. В вычислительном центре создан специализированный язык, который автоматизирует лишь часть работы программиста. В эту часть не входит трудоемкое программирование модели коммутационной структуры, сложной или просто объемистой, и установления соединений через неё. Между тем, моделирование структуры существенно зависит от опыта и искусства программиста. Вообще, возможности автоматизации программирования пока невелики. Поэтому основной путь в преодолении ограниченности возможностей заключается в упрощении математической модели задачи. Упрощение может относиться к структуре, к процессу обслуживания, к методам оценки характеристик обслуживания.

Интересным примером может служить способ упрощения многокаскадной структуры, примененный в [I]. Структура в данном случае восьмикаскадная, рассматриваемая применительно к квазиэлектронным АТС, имеет следующие характеристики. Она состоит из некоторого числа одинаковых коммутационных блоков одного типа - блоков абонентских линий, или, сокращенно, БАЛ, и из некоторого числа одинаковых коммутационных блоков другого типа - блоков

соединительных линий, или БСЛ. Каждый БАЛ связан с каждым БСЛ пучком междублочных соединительных линий. Число линий во всех пучках одно и то же, и все пучки, исходящие из одного и того же БАЛ, включены в БАЛ в известном смысле симметрично. То же можно сказать о пучках, входящих в один и тот же БСЛ. Каждый вызов обслуживаемого потока поступает на вход некоторого БАЛ и соединение должно быть установлено через данный БАЛ, через междублочный пучок и через некоторый БСЛ к выходу БСЛ. Другие соединения проходят в обратную сторону - от определенного выхода БСЛ ко входу БАЛ. Обычная структура, в которой может быть до 32 и даже до 64 БАЛ и столько же БСЛ, имеет очень много оставных элементов, и полную модель структуры нельзя построить в ЭВМ из-за недостатка быстродействующей памяти. Вместе с тем для моделирования процесса установления соединений внутри БАЛ и внутри БСЛ не требуется многократно повторять одну и ту же структуру блока. Поэтому в [I] и построена следующая примерно эквивалентная структура, удобная для моделирования. Пусть для определенности число блоков каждого типа равно 32, а каждый междублочный пучок содержит 32 линии. Эквивалентная структура состоит из двух БАЛ и двух БСЛ. Между каждым БАЛ и каждым БСЛ имеется 512 линий. Примерная эквивалентность исходной схемы с пучками по 32 линии достигается с помощью дополнительного условия: 512 линий разбиты на 16 пучков по 32 линии, причем эти 32-линейные пучки включены в БАЛ и в БСЛ так же, как включены 32-линейные пучки в исходной структуре. Каждому вызову, поступающему на вход определенного БАЛ /или БСЛ/ приписывается в случайном порядке по одному 32-линейному пучку в каждом из двух 512-линейных пучков данного БАЛ /БСЛ/. Только через эти 32-линейные пучки можно устанавливать соединение по данному вызову.

В наших исследованиях при подготовке задач к моделированию применялось несколько приемов упрощения, которые излагаются ниже.

Хорошие возможности открывает разбивка задачи на части. В каждой части без упрощений представлена лишь часть изучаемой структуры и процесса обслуживания. Остальные части заменены некоторыми примерными эквивалентами. Данные, полученные при моделировании одной части задачи, используются в качестве исходных данных для другой части задачи.

Так, например, ступень группового искания /ГИ/ координатной АТС состоит из некоторого числа /до 20 и больше/ однотипных коммутационных структур, называемых блоками группового искания /блоками ГИ/ и некоторого числа пучков линий /до 20 пучков/. На входы блоков ГИ поступают вызовы обслуживаемых потоков, причем в каждом потоке вызовы расходятся по входам всех блоков ГИ. Каждый пучок линий включен неполнодоступно в выходы всех блоков ГИ. Таким образом, структура едина и, строго говоря, её нельзя разбить на части и моделировать каждую часть отдельно. Вместе с тем модель всей ступени просто невозможна из-за ограниченного объема оперативной памяти. Однако с очень хорошим приближением можно получить нужные результаты сделав не программу моделирования полной ступени ГИ, а две гораздо более простые программы - для моделирования одного неполнодоступного пучка с учетом в этой модели влияния остальной части ступени ГИ и для моделирования одного блока ГИ также с учетом влияния остальной части ступени ГИ. Программа моделирования одного неполнодоступного пучка позволяет, сделав достаточное число экспериментов, определить нужные характеристики во всем возможном диапазоне параметров неполнодоступного включения. Программа моделирования одного блока ГИ также позволяет при достаточном числе экспериментов получить все необходимые данные. В результате такого подхода объем памяти для

модели структуры уменьшается на один-два порядка. Две программы в совокупности значительно проще, чем одна сложная программа. Их можно делать независимо и одновременно. Моделирование становится реализуемым.

Поясним подробнее, каким образом две отдельные программы приводят к нужному результату, т.е. к оценке пропускной способности сложной ступени при разных наборах параметров.

В первой программе на входе каждой нагрузочной группы имеется "вероятностный фильтр", который с заданной вероятностью создает отказы, причем вероятность зависит от состояния пучка, а именно, от числа занятых линий среди линий, доступных данной нагрузочной группе. Фильтр действует как эквивалент блока ГИ в смысле блокировок между входами и выходами блока. Фильтр задается числовым вектором, который получается в результате моделирования одного двухкаскадного блока по второй программе. В свою очередь, вторая программа функционирует с данными, полученными при моделировании неполнодоступных пучков. Эти данные вводятся как параметры эквивалентных потоков вызовов, поступающих на части неполнодоступных пучков, доступные через один блок ГИ.

Для согласования результатов обеих программ может потребоваться итеративное моделирование. Практически, однако, по приобретении небольшого опыта, можно обойтись одним-двумя шагами. В частности, хорошо начать с неполнодоступного включения, положив все вероятности фильтра равными 0, и использовать данные этой модели для определения фильтра по программе моделирования блока ГИ. Полученный фильтр уже достаточен для окончательного моделирования по программе неполнодоступного пучка.

Многие результаты можно получать, объединяя моделирование с вычислениями. Покажем пример. Пусть задана многокаскадная структура, в выходы которой могут

включаться пучки разной емкости. Для определения потерь на пучках разной емкости при разных их нагрузках нужно было бы моделировать много разных частных случаев. Однако можно воспользоваться формулой Лонгли для вероятности потерь вызовов в пучке, включенном через некоторую блокирующую схему. Эта формула представляет собой функциональную зависимость вида

$$p = \sum_{i=0}^{v} \varepsilon_i P_i(\lambda; \varepsilon_0, \varepsilon_1, \ldots, \varepsilon_v)$$

/подробнее см., например, [2] /.
Здесь v – число линий в пучке, ε_i – вероятность блокировки при условии, что в пучке занято i линий, P_i – вероятность занятости линий, зависящая от нагрузки пучка и от вероятностей блокировок. Нетрудно понять, что в нашем случае числа ε_i совпадают с вероятностями блокировки при $v-i$ попытках установить соединение. Таким образом, можно моделировать многокаскадную структуру без моделирования конкретных пучков /упрощается программа/ и, получив вероятности блокировок, вычислять потери в направлениях по простой формуле /уменьшается расход машинного времени/. В этом примере важно, в частности, что, сократив объем моделирования, можно получить практическую пользу от программы, по которой одна точка моделируется в течение нескольких часов.

Поясняющий числовой пример. Пусть в некоторой многокаскадной структуре с заданной общей нагрузкой на её каскадах вероятности блокировок при нуле, одной, двух и т.д. попытках суть $\varepsilon_v = 1{,}00$, $\varepsilon_{v-1} = 0{,}225$, $\varepsilon_{v-2} = 0{,}069$, $\varepsilon_{v-3} = 0{,}027$, $\varepsilon_{v-4} = 0{,}012$, $\varepsilon_{v-5} = 0{,}007$, $\varepsilon_{v-6} = 0{,}004$ и т.д. Покажем, что эта структура по-разному блокирует пучки разной емкости, но что для этого достаточно знать указанные значения ε_i, полученные однократным моделированием при данной общей загрузке структуры, и воспользоваться простыми вычислениями по формуле Лонгли, проводя

их отдельно для каждого пучка. Возьмем, например, пучки емкостью 10 и 80 линий. Если первый из них рассматривать при нагрузке 3,43 Эрл, а второй – при нагрузке 59,7 Эрл, то оба пучка в условиях полнодоступного их включения обслуживают вызовы с потерями 0,002. Используя формулу Лонгли при данной загрузке каскадов, т.е. при данных значениях ε_i , получим, что те же пучки, работая при тех же нагрузках не полнодоступно, а через многокаскадную структуру, создают потери соответственно 0,0082 и 0,0027. Отсюда видно, что чем меньше пучок, тем больше дополнительных потерь вносят блокировки при установлении соединений через данную структуру. При той же общей загрузке каскадов и потерях в полнодоступном включении, равных 0,05, потери с блокировками будут 0,068 и 0,053 для пучка с 10 и 80 линиями соответственно.

С помощью таких же простых расчетов можно выявить и другие особенности загрузки пучков через данную многокаскадную структуру. Например, можно определить потери при изменении нагрузки отдельных пучков, когда общая нагрузка структуры сохраняется или даже, когда она изменяется. В последнем случае нужно провести однократное моделирование структуры при новой нагрузке и получить значения ε_i .

Примеры объединения моделирования с расчетами можно было бы умножить. В расчетах могут использоваться всевозможные данные, полученные моделированием, например, распределение числа занятых линий, коммутаторов, звеньев и пр. В частности, сложная структура может быть разбита на отдельно моделируемые части и при моделировании могут измеряться распределения на стыках этих частей. После измерений вычисляются комбинации распределений, что дает требуемую характеристику целой структуры. Процесс решения задачи может быть итеративным, поскольку при моделировании каждой отдельной части используются

результаты моделирования других частей, как объяснялось раньше на примере ступени ГИ. Вычислительная программа может быть составной частью программы моделирования, обрабатывая входные или выходные данные этой программы.

Перейдем теперь к общим положениям постановки задачи для программирования. В диалоге между математиком и инженером от последнего требуется строгая постановка задачи. Постановка задачи состоит из трех основных частей: описания коммутационной структуры, описания исходных данных процесса обслуживания /потоки вызовов, время обслуживания, дисциплина обслуживания/, описания требуемых статистических данных.

Описание структуры лучше делать формальным. Телефонные термины можно, конечно, сохранять, однако все понятия нужно строго определить. Сложные структуры лучше задавать таким образом, чтобы задание каждого конкретного варианта моделирования было простым. Так, например, многокаскадная структура может быть задана таблицами, показывающими соединения между каскадами. Однако значительно лучше задавать структуру с помощью линейных зависимостей. Формулы могут обозначать, например, звенья, выражая номера входов коммутаторов каждого каскада в зависимости от номеров выходов и коммутаторов другого каскада. Задание конкретных вариантов становится очень простым. Кроме того, формулы могут подсказывать схему моделирования структуры, так как формулы можно использовать для адресации ячеек, в которых записываются состояния элементов структуры.

При задании потоков вызовов, времени обслуживания и дисциплины обслуживания нужно стремиться к тому, чтобы в сложных моделях процесс был марковским. Это упрощает и программирование, и моделирование.

Статистическая информация, получаемая в результате

моделирования, должна быть избыточной, что помогает отлаживать программу. Например, задав вариант моделирования так, чтобы какая-нибудь группа элементов была полнодоступной, можно сопоставить реализуемое распределение числа занятых элементов с эрланговским. Существует множество приемов такой проверки.

Описание статистических данных должно быть столь же строгим, как и другие части задания на моделирование. Лучше, если постановщик задачи будет задавать методику вычисления оценок, а не вероятности, которые требуется оценить.

В постановку задачи входит описание ввода и вывода данных. Лучше всего сразу показать точную форму записи входных данных на бланках. Форма ввода и, особенно, вывода данных практически согласуется еще в течение некоторого времени, пока программист изучает задачу и работает над алгоритмами. Обычно моделирование каждого варианта ведется несколькими независимыми сериями /например, по 10 тысяч поступивших вызовов/. Это самый простой способ получить не только статистические оценки требуемых характеристик, но и выборочную дисперсию оценок.

Особо следует сказать о последовательностях псевдослучайных чисел. Бывает, что с точки зрения инженера результаты моделирования оказываются неправдоподобными по единственной причине - из-за плохих последовательностей. Опыт работы ВЦ ЛГУ показал, что для реализации разных случайных событий подходят разные последовательности и выяснить, какие именно, можно только опытным путем. При этом общестатистические критерии могут оказаться недостаточными и нужно прибегнуть к оценке результатов моделирования, например, к оценке того, как реализуется распределение Эрланга или распределение времени ожидания при моделировании полнодоступного пучка.

В заключение скажем, что не нужно стремиться к универсальности программ, наоборот иногда лучше сразу задать конкретные числовые параметры сложной структуры, не

нужно стремиться к тому, чтобы "на всякий случай" программа давала избыточные возможности. Можно даже пойти на то, чтобы для быстроты решения задачи не реализовать весь нужный диапазон параметров и интерполировать, а иногда даже и экстраполировать результаты моделирования. Не нужно задавать много вариантов дисциплины обслуживания, если только задача не посвящена специально этому вопросу.

Изложенное понимание проблем моделирования, встречающихся инженеру, выработалось на основании многолетнего сотрудничества с математиками Вычислительного центра. При этом математики наталкивались на свои проблемы и много сделали для их разрешения. Без долгого целенаправленного труда математиков не было бы и данной работы, что с благодарностью помнит и ценит автор.

ЛИТЕРАТУРА

1. Takagi K., Itoh M. Internal Blocking Probability of Eight-Stage Link Systems for Electronic Switching Systems.—"Review of the Electrical Communication Laboratory", 1970, v.18, №11-12, p.25 - 30.

2. Башарин Г.П., Харкевич А.Д., Шнепс М.А., Массовое обслуживание в телефонии. М., "Наука", 1968. 246 с.

СОДЕРЖАНИЕ