ВВЕДЕНИЕ В ТЕХНИКУ РАБОТЫ С ТАБЛИЦАМИ РЕШЕНИЙ

БИБЛИОТЕКА ПО АВТОМАТИКЕ

Выпуск 603

ВВЕДЕНИЕ В ТЕХНИКУ РАБОТЫ С ТАБЛИЦАМИ РЕШЕНИЙ

Перевод с немецкого
М. Г. Гаазе-Рапопорта

Под редакцией
Д. А. Поспелова

МОСКВА «ЭНЕРГИЯ» 1979

ББК 32.973
В 24
УДК 681.3.06

EINFÜHRUNG IN DIE ENTSCHEIDUNGSSTABELLENTECHNIK

GERD FREITAG, WOLFGANG GODE, HORST JACOBI,
HEINZ LAUTZ, JOERG SIMON, ULRICH SPITTEL

VEB VERLAG TECHNIK, BERLIN, 1976

Введение в технику работы с таблицами решений.
В 24 Пер. с нем./ Фрайтаг Г., Годе В., Якоби Х. и др.— М.: Энергия, 1979, 88 с., ил. — (Б-ка по автоматике; Вып. 603).

35 к.

В книге описывается техника работы со специальными таблицами, которые используются в машинах третьего поколения для описания и организации процессов, связанных с операционными системами ЭВМ, и для решения задач, связанных с хранением и обработкой структурированной информации в организационных задачах, задачах технологического проектирования и т. п. Книга написана специалистами ГДР.

Книга предназначена для программистов, пользователей и студентов, специализирующихся в области математического обеспечения ЭВМ и решения сложных задач на ЭВМ.

В $\frac{30502\text{-}431}{051(01)\text{-}79}$ 180-79. 2405000000 ББК 32.973
6Ф7

ПРЕДИСЛОВИЕ РЕДАКТОРА РУССКОГО ИЗДАНИЯ

Существуют два типа языков программирования: процедурные и непроцедурные. Процедурные языки описывают саму процедуру решения задачи на ЭВМ, задают в явном виде алгоритм нахождения решения. К таким языкам принадлежат широко известные АЛГОЛ, ФОРТРАН, ПЛ/1 и многие другие. Весь начальный период развития программирования для ЭВМ прошел под знаком изобретения все новых и новых процедурных языков, число которых к настоящему времени с учетом всевозможных версий и подмножеств достигло десятка тысяч. Правда, широко распространенных процедурных языков существенно меньше — два-три десятка. Тем не менее отсутствие функциональной совместимости программ, написанных на различных процедурных языках, приводит к большому параллелизму в работах программистов и затруднению обмена результатами между специалистами, работающими на различных процедурных языках. Кроме того, поскольку ни один из существующих языков не является общепризнанно наилучшим или подавляюще распространенным, то, как и для естественных языков, по-видимому, нельзя построить общий язык типа эсперанто, на котором общались бы между собой программисты, специалисты и ЭВМ, использующие различные процедурные языки. Наконец, еще одной отрицательной чертой процедурных языков является то, что пользователь должен предварительно изучить этот язык, потратив на это изучение значительное время и не менее значительные усилия.

Достоинством процедурных языков является их строгость, заставляющая пользователя точно описывать интересующую его задачу, а также облегчающая формальный перевод (транслирование) записи алгоритма на процедурном языке во внутримашинное представление программы.

Непроцедурные языки, как теперь уже понятно, описывают не какой-либо конкретный алгоритм решения задачи, а ее точную постановку и необходимые условия ее решения. Сейчас существует целый спектр непроцедурных языков. В наиболее мощных языках подобного типа пользователь должен лишь точно сформулировать задачу из некоторой фиксированной проблемной области, а ЭВМ с помощью вложенных в нее средств (важнейшим из которых является планировщик, работающий на семантической сети, отражающей все связи между понятиями, входящими в описание задачи, и типовыми вычислительными и логическими модулями, хранящимися постоянно в памяти ЭВМ) сама строит алгоритм решения и превращает его в программу. Вот пример такого задания: «Найти площадь треугольника ABC, если известно, что $a=2$, угол $\beta=40°$, а угол $\gamma=60°$». При этом в семантической сети должны быть отражены все необходимые функциональные связи между элементами треугольника, позволяющие по этой информации строить алгоритм ре-

шения из таких типовых модулей, как ВЫЧИСЛЕНИЕ ПЛОЩАДИ ПО ФОРМУЛЕ ГЕРОНА, ТЕОРЕМА СИНУСОВ И ВЫЧИСЛЕНИЯ ПО НЕЙ, ОПРЕДЕЛЕНИЕ УГЛА ПО ДВУМ ДРУГИМ и т. п.

Такие непроцедурные языки по своей сути есть общечеловеческие языки, привычные пользователю, не требующие от него никаких дополнительных усилий для написания заданий ЭВМ. Однако недостатком их является значительная сложность математического обеспечения, не позволяющего использовать их возможности в полной мере. Оно может быть использовано лишь на мощных ЭВМ, имеющих большие ресурсы памяти и производительности. Для средних и малых ЭВМ в последние годы получили распространение непроцедурные языки более «низкого уровня». Самым популярным из них является язык таблиц решений, рассматриваемый в этой книге. Это уже вторая книга на русском языке, посвященная языку таблиц решений (первой был перевод книги известного программиста Э. Хамби «Программирование таблиц решений», М.: Мир, 1976). В отличие от Хамби автор данной книги много внимания уделяет таблицам решений с точки зрения пользователя, принципам их составления, применения в различных проблемно-ориентированных областях науки и техники. Язык книги прост. Она доступна любому специалисту, который до этого даже не сталкивался с проблемами решения задач на ЭВМ. Если для него важно только то, как он должен готовить свою задачу для ЭВМ, то он может не читать гл. 3 и 4. Для людей, знакомых с программированием, материал этих глав поможет понять, что стоит для ЭВМ с точки зрения ее математического обеспечения внедрение языка таблиц решений. В любом случае, как мне кажется, чтение книги Г. Фрайтага и его соавторов принесет несомненную пользу и натолкнет вдумчивого читателя на мысли о преимуществах непроцедурного программирования и путях построения языков подобного типа.

Д. А. Поспелов

ПРЕДИСЛОВИЕ

При подготовке к внедрению и эксплуатации ЭВМ третьего поколения становится все более важной систематизация процессов, связанных с обработкой данных. При этом, в частности, должны быть учтены и вопросы технической подготовки производства. Поэтому необходима разработка действенных методов для анализа и представления сложных структур и процессов обработки данных.

Одним из таких методов является метод, основанный на использовании таблиц решений (ТР), который излагается в настоящей книге в форме, доступной для широкого круга читателей.

Изложение элементарно и не предполагает специальных знаний, хотя некоторые сведения из области электронной обработки данных будут, конечно, полезны.

Изложенный материал сформировался в результате творческого содружества Исследовательского центра управления АУТЕФО (AUTEVO) в Йене и Народного предприятия «Машинные вычисления» (VEB Mashinelles Rechnen) в Лейпциге.

Авторы благодарят редактора, дипломированного математика Г. Паулина и рецензента, дипломированного инженера Ю. Рейхенбаха за многочисленные полезные указания при работе над рукописью.

Авторы

ВВЕДЕНИЕ

Таблицы решений (ТР) используются для анализа, описания и документирования сложных процессов, структур и задач. Они служат для той же цели, что и известные блок-схемы программ. Задача, описанная в виде блок-схемы, может быть представлена в виде ТР и наоборот. Несмотря на это, следует различать эти два способа описаний. Блок-схемы быстро становятся необозримыми, если выполняемые действия, операторы и соответственно решения в рамках одного процесса зависят от многочисленных связанных между собой условий. При этом затрудняется контроль их полноты и истинности. Использование ТР существенно упрощает решение этих задач.

Кроме того, ТР допускают дальнейшую формализацию и могут использоваться для непосредственного ввода информации в ЭВМ. Написанные в такой форме программы легко поддаются изменениям и обеспечивают простое документирование. Для менее сложных задач при этом, как и ранее, адекватным средством представления могут служить блок-схемы.

Настоящая книга, помимо введения, содержит пять глав. В гл. 1 излагаются основы техники ТР, поясняемые на простых примерах. Предлагаемая концепция ТР применима для описания многих процессов, при этом временные затраты на обработку данных с помощью ЭВМ вполне приемлемы.

В гл. 2 рассматриваются основные черты метода формирования ТР и приводится большой поясняющий пример.

В гл. 3 описываются некоторые приемы, которые могут использоваться при обработке ТР на ЭВМ. Из множества известных методов здесь выбираются некоторые типичные представители.

В гл. 4 обсуждаются вопросы программирования (от ручного программирования до автоматической трансляции) при использовании ТР.

Глава 5 содержит два больших примера: первый (из области технологии) демонстрирует простоту описываемой техники для больших задач; второй (экономическая задача) показывает метод решения для достаточно часто встречающегося типа задач.

ГЛАВА ПЕРВАЯ

ОСНОВНЫЕ ПОЛОЖЕНИЯ

В этой главе будут пояснены основные понятия техники ТР и связи между ними. Изложение ведется на очень доступном для достаточно широкого круга специалистов уровне. Все теоретические вопросы поясняются примерами.

1. ВВОДНЫЙ ПРИМЕР

Рассмотрим простую задачу. Пусть вычислительная машина должна решать квадратные уравнения вида $AX^2+BX+C=0$. Коэффициенты A, B и C — целые. Их значения вводятся в ЭВМ с помощью перфокарт. Первая перфокарта содержит признак карты КК=1, а также коэффициенты A, B и C первого квадратного уравнения. Вторая карта содержит снова признак карты КК=1 и коэффициенты второго квадратного уравнения и т. д. Конец пакета карт обозначен картой с признаком КК≠1 (например, КК=0). Эта карта не содержит значений коэффициентов. Как должно быть организовано управление вычислениями, чтобы ЭВМ по мере надобности выполняла необходимые операции?

Поясним это на примере.

Если КК=1, A=0, B=0 и C=0, то любое комплексное число является решением соответствующего уравнения.

Если КК=1, A=0 и B≠0, то имеет место линейное уравнение и любая информация о C не имеет значения.

Если КК=1, A≠0 и $D=B^2-4AC<0$, то должны быть вычислены комплексные решения.

Для того, чтобы быть уверенным, что каждой мыслимой ситуации поставлено в соответствие правильное действие, составим таблицу в соответствии с рис. 1, в верхней части которой записаны определенные условия, а в нижней действия (названия процедур или типов действий).

При составлении таблиц приняты следующие обозначения: Y — ДА (от англ. yes); N — НЕТ (от англ. no); «—» (тире) — НЕСУЩЕСТВЕННО (в дальнейшем мы будем это обозначение называть словом «пробел»); × — действие, стоящее в соответствующей строке, подлежит выполнению; Ri — порядковый номер правила.

Иногда целесообразно пустые позиции нижней части таблицы (часть действий) заполнить пробелами.

Таблица рис. 1 используется следующим образом.

Если КК=1, A=0, B=0 и C=0, то выполняется действие ПРОИЗВ (сокращение, означающее произвольное решение). Условие D<0 в этом случае не проверяется, что обозначается знаком «—»

ТР 1	R1	R2	R3	R4	R5	R6
КК = 1	Y	Y	Y	Y	Y	N
A = 0	Y	Y	Y	N	N	—
B = 0	Y	Y	N	—	—	—
C = 0	Y	N	—	—	—	—
D < 0	—	—	—	Y	N	—
ПРОИЗВ	×					
ОШИБКА		×				
ЛИНЕЙН			×			
КОМПЛЕКС				×		
ВЕЩЕСТВ					×	
КОНЕЦ						×

Рис. 1. Возможные варианты решения уравнения $AX^2+BX+C=0$.

(пробел) на соответствующем месте. Если в этом решении (столбце) имеется хоть одно несовпадение условий, то проверяется следующее решение (столбец). Если оно не имеет места, то переходим к следующему и т. д. Как только (в описанном случае при движении слева направо) встретится соответствующая комбинация условий, таблица покидается и выполняются действия, указанные крестиком. Если это действие не КОНЕЦ. то считывается новая карта и таблица обрабатывается заново.

Такая таблица носит название *таблицы решений.*

2. ОСНОВНЫЕ ПОНЯТИЯ И ОБОЗНАЧЕНИЯ

Таблицей решений с простыми (ограниченными) входами (в дальнейшем для краткости *таблицей решений* или *таблицей*) называется табличное упорядочение правил решений в форме, изображенной на рис. 2. Входы b_{ik}, как и на рис. 1, могут иметь значения Y, N или «—» и имеют тот же смысл. Строки верхней части таблицы называются *условиями*[1], столбцы правой части *правилами.* Правый верхний квадрант будем называть *указателем условий* (*матрицей условий*), правый нижний — *указателем действий* (*матрицей действий*). Векторы $S=(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ с $\varepsilon_i=Y$ или N будем называть *ситуациями.* При n условиях существует ровно 2^n ситуаций. То обстоятельство, что ТР для ситуации S указывает действие (действия), по правилу Rk будем иногда записывать в форме TP(S) = Rk. Если

[1] В последующем будем считать, что условия таблицы решений могут проверяться независимо друг от друга. Это допущение не приводит к существенным ограничениям.

Наименование таблицы	Правило 1	Правило 2	. . .	Правило m
Условие 1	b_{11}	b_{12}	. . .	b_{1m}
Условие 2	b_{21}	b_{22}	. . .	b_{2m}
•	.	.	. . .	. . .
•	.	.	. . .	. . .
•	.	.	. . .	. . .
Условие n	b_{n1}	b_{n2}	. . .	b_{nm}
Действие 1	×			
Действие 2	×	×		×
•				
•				
•				
Действие k				×

Рис. 2. Общая структура ТР.

такого правила не существует, то будем писать TP(S)=0. Это означает, что ситуация S не принадлежит к области определения ТР.

Для чисел k, m и n справедливы соотношения: $1 \leqslant m \leqslant 2^n$, $n \geqslant 0$; $k \geqslant 1$. Таблица решений, таким образом, имеет по крайней мере одно правило и по меньшей мере одно действие. Однако из формальных соображений допустимо существование ТР без условий (n=0). Основания для этого будут приведены позже. Чтение (интерпретация) этой таблицы осуществляется так же, как и в рассмотренном ранее примере (рис. 1). Пусть дана ситуация $S=(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$. Если таблица без условий (n=0), то ей соответствует первое (и единственное) решение. В противном случае текущая ситуация покомпонентно сравнивается с соответствующими элементами (Y или N) первого правила (R1). При совпадении имеет место запись: TP(S)=R1. При несовпадении ищется другое правило, которое таким же образом сравнивается с S, т. е. имеет место запись: TP(S)=0.

Из такого понимания таблиц вытекает одно важное *следствие*:

Пусть (Ri, Rk) — пара правил с i<k. Если при этом не существует индекса j, такого, что $(b_{ji}, b_{jk})=(Y, N)$ или (N, Y) и соответственно $(b_{ji}, b_{jk})=(Y, —)$ или (N, —), то правило Rk никогда не будет использоваться и может быть вычеркнуто.

Пример 1. Пусть S1=(Y, Y, N), т. е. ALPHA=0, BETA=1 и GAMMA≠2 (рис. 3). В силу того, что ε_2=Y и b_{21}=N, правило R1 не действует. Переходим к R2, а затем к R3 и находим, что

$$TP3(S1)=R3.$$

Для S2=(Y, N, Y) получаем TP3(S2)=R1; для S3=(N, Y, Y) справедливо TP3(S3)=0. Правило R5 является лишним, так как в элементах пары (R3, R5) не встречается ни одной из требуемых

пар (Y, N), (N, Y), (Y, —), (N, —). Это означает, что все ситуации, соответствующие R5, уже встречались в R3, что позволяет условно написать

$$R5 < R3.$$

Во многих практических случаях встречаются обстоятельства, в силу которых простых входов (Y, N и «—») в условиях и знаков × (кресты) в матрицах действий недостаточно для представления задачи.

ТР3	R1	R2	R3	R4	R5
ALPHA = 0	Y	N	Y	N	Y
BETA = 1	N	Y	—	N	—
GAMMA = 2	—	N	—	N	Y
A = 1	×	×		×	
B = 2		×	×	×	×

Рис. 3. Преобразование ТР (пример).

Пример 2. Для опорожнения цистерн должны выбираться вентили в соответствии со следующей логикой:

Для цистерн (ЦИСТ) с водой выбирается вентиль (ВЕНТ) V1, если давление не превышает 5 кгс/см²; при давлении (ДАВЛ) между 5—10 кгс/см² предусмотрен вентиль V2. Для цистерн с нефтью используется вентиль V3, а если при этом давление больше 8 кгс/см² — вентиль V4. Для цистер с кислотой всегда используется вентиль V5. В любом другом случае должна быть предусмотрена новая конструкция. Это положение может быть описано таблицей, изображенной на рис. 4.

При этом действие НОКОНСТ должно отсылать к очередной новой конструкции. Такая таблица представляет описание правил вы-

ТР4	R1	R2	R3	R4	R5	E
ЦИСТ	ВОДА	ВОДА	НЕФТЬ	НЕФТЬ	КИСЛОТА	
ДАВЛ	⩽5	⩽10	⩽8	—	—	
ВЕНТ	V1	V2	V3	V4	V5	
НОКОНСТ						×

Рис. 4. Выбор вентилей для цистерн.

бора. Таблицы такого типа будем называть *таблицами решений с расширенными входами* (или *расширенными таблицами решений*). Таблицы, в которых одновременно встречаются и простые и расширенные входы, будем относить к расширенным таблицам [1].

Вернемся к рис. 4. Какой вентиль должен быть выбран для цистерны, в которой находится вода под давлением 4 кгс/см²? Для этого подходят вентили V1 и V2. Однако следует однозначно выбрать вентиль V1, так как таблица упорядочена по решениям слева направо.

Поясним смысл шестого правила, которое обозначено через E — ИНАЧЕ (от англ. else). Оно означает, что если ни одна из описанных в правилах R1—R5 ситуаций не имеет места, то должно выполняться специальное действие, отмеченное в столбце E (ИНАЧЕ), без выполнения дальнейших проверок. В нашем случае должен быть сконструирован и включен в общий набор (ассортимент) новый вентиль.

В таблицу очень легко ввести соответствующие изменения, для этого новый столбец обозначают R6 и сдвигают правило ИНАЧЕ на один столбец правее. Это бесспорное преимущество ТР, позволяющее очень просто изменять и совершенствовать их. Естественно, и для таблиц с ограниченными входами может быть использовано правило ИНАЧЕ.

При этом:

а) таблица всегда покидается в результате некоторого специфического действия. Случай TP(s) = 0 не имеет места;

б) при намерении автоматизировать подготовку производства в соответствии с желаниями потребителей часто встречается следующее положение.

При большом числе вполне детерминированных правил, позволяющих описать большую часть классов изделий, конструктивных и технологических операций, допускающих их полную автоматизацию, не исключены случаи, представление которых связано со значительными трудностями. В таких случаях ТР с правилом ИНАЧЕ чрезвычайно удобны. Исследованные случаи выделяют в группу начальных правил, а нерассмотренные случаи объединяют с помощью правила ИНАЧЕ. Действие по этому правилу является специальным сигналом для дальнейшего функционирования ЭВМ.

Рассмотрим несколько примеров для иллюстрации этого положения.

Пример 3. Для выбора электрических приборов должны быть установлены следующие величины: W (мощность), I (ток), U (напряжение) и R (сопротивление). При этом чаще всего задаются две величины, а две остальные вычисляются по следующим формулам:

$$F1: R = U/I; \quad W = UI;$$

$$F2: I = U/R; \quad W = U^2/R;$$

$$F3: R = U^2/W; \quad I = W/U;$$

$$F4: U = IR; \quad W = I^2R;$$

$$F5: U = \sqrt{R\,W}; \quad I = \sqrt{W/R};$$

$$F6: R = W/I^2; \quad U = W/I.$$

---

[1] Встречающееся в литературе понятие «смешанная таблица» мы рассматриваем как практически неоправданное.

Опишем с помощью ТР, когда какие формулы должны быть использованы. Ошибочным случаем будет тот, когда задано менее двух величин. Объединяя эти случаи в правило ИНАЧЕ, предусмотрим действие ОШИБКА.

На рис. 5 представлено решение задачи. Конечно, существуют и другие способы составления ТР, но они дадут подобную приведенной таблицу. Однако было бы ошибкой заменить все знаки [1] «—» на N.

ТР5	R1	R2	R3	R4	R5	R6	E
W	Y	Y	Y	—	—	—	
U	Y	—	—	Y	Y	—	
I	—	Y	—	Y	—	Y	
R	—	—	Y	—	Y	Y	
ИСПОЛЬЗУЙ	F3	F6	F5	F1	F2	F4	
ОШИБКА							×

Рис. 5. Выбор формул для расчета значений мощности, напряжения, тока и сопротивления.

В этом примере мы рассматривали задачу, решение которой может быть описано с помощью различных ТР. Это типично. Нетипично то, что различия несущественны и что они легко обозримы. Существуют ТР, которые выглядят «сильно различающимися» и тем не менее выражают одно и то же.

Рассмотрим в связи с этим следующий пример.

Пример 4. Для того, чтобы во время спортивных соревнований иметь объективные возможности для сравнения результатов, вводятся различные программы соревнований. Женщины W (мужчины M), активно занимающиеся спортом в свободное время, должны выполнить программы 1 (2). Участники, не занимающиеся активно спортом, делятся на возрастные группы: женщины (мужчины) возрастной группы k (k=1, 2, 3) выполняют программы 1k (2k).

Как представить это положение с помощью таблицы решений?

Условиями, определяющими выбор программы соревнований (ПРОГР), являются: пол (ПОЛ), возрастная группа (ВОГР) и активное занятие спортом (АКТИВ). С учетом этого получается таблица, изображенная на рис. 6.

Интересно построить таблицу так, чтобы в ней имела место другая последовательность условий (рис. 7).

Ясно, что ТР7 выражает то же, что и ТР6. Нетрудно преобразовать (редуцировать) ТР7. Правила R1, R5 и R9 отличаются толь-

[1] Читателям предлагается обосновать это утверждение.

ТР6	R1	R2	R3	R4	R5	R6	R7	R8
ПОЛ	W	W	W	W	M	M	M	M
АКТИВ	Y	—	—	—	Y	—	—	—
ВОГР	—	1	2	3	—	1	2	3
ПРОГР	1	11	12	13	2	21	22	23

Рис. 6. Выбор программы соревнований.

ко возрастными группами. Каждая из возможных возрастных групп встречается строго 1 раз. Это означает, что в этих правилах переменная ВОГР не имеет значения и может быть заменена прочерком (пробелом). Применим эти же соображения и к правилам R2, R6 и R10. Тогда, поменяв местами условия 1 и 3, а также некоторые правила, получим таблицу, изображенную на рис. 6.

ТР7	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
ВОГР	1	1	1	1	2	2	2	2	3	3	3	3
АКТИВ	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
ПОЛ	W	M	W	M	W	M	W	M	W	M	W	M
ПРОГР	1	2	11	21	1	2	12	22	1	2	13	23

Рис. 7. Выбор программы соревнований при изменении порядка условий.

Возникает, правда, вопрос, можно ли над таблицами решений производить подобные операции? Далее мы займемся и этой проблемой.

3. ПОЛНЫЕ ТАБЛИЦЫ РЕШЕНИЙ И ТАБЛИЦЫ ТИПА P[1]

Иногда бывает трудно установить, полон ли анализ задачи или имеются еще и другие варианты, которые следует рассмотреть. Техника таблиц решений дает формальные методы для выполнения та-

[1] Названы в честь Поллака (G. S. Pollack), написавшего основополагающие статьи об этом важном классе ТР.

кого анализа задач, заданных в форме таблиц, имеющих простые (ограниченные) входы в указателе условий. Как будет показано, таблицы с расширенными входами могут быть преобразованы в таблицы с ограниченными входами. Поэтому сделанное ограничение несущественно.

Необходимо иметь в виду два различных понятия полноты ТР.

Таблица решений с n *условиями называется формально полной, если для каждой из всех* 2^n *ситуаций* $S=(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ с $\varepsilon=Y$ *или* N *предусмотрено по одному решающему правилу.*

TP8	R1	R2	R3
C1 = 0	Y	Y	N
C2 = 0	Y	N	—
	M1	M2	M3

Рис. 8. Формально полная ТР с двумя условиями.

В соответствии с этим ТР с правилом ИНАЧЕ тривиальным образом формально полны и, следовательно, не представляют интереса. Для примера скажем, что таблица на рис. 8 формально полна, так как она предусматривает наличие правила для каждой из четырех ситуаций, а таблица же, изображенная на рис. 3, формально неполна, так как для ситуаций (N, Y, Y) и (N, N, Y) не существует правил.

Чтобы получить удобные критерии проверки формальной полноты для больших таблиц, введем следующее определение:

Таблица называется таблицей типа P, *если ее правила попарно разбиваются по меньшей мере одним соответствующим указателем условий* [1].

Пусть ТР — таблица решений с n условиями и k — натарульное число ($1 \leqslant k \leqslant n$). Тогда будем понимать под TPYk (TPNk) такие *частные таблицы*, которые получаются из ТР путем вычеркивания k-го условия и всех правил Ri с $b_{ki}=N$ ($b_{ki}=Y$). Для n=1 частные таблицы TPY1 и TPN1 безусловны.

Это определение частных таблиц распространяется и на таблицы с правилом ИНАЧЕ. Естественно, что обе частные таблицы должны содержать одно и то же правило ИНАЧЕ. Решающее правило для ситуации с Y(N) в k-м условии может поэтому находиться только в таблице TPYk (TPNk). Иначе говоря, TPNk(S)=0, если k-е условие в S имеет значение Y, в противном случае TPYk(S)=0

Таким образом, нетрудно разложить ТР на частные таблицы. Многие алгоритмы, превращающие ТР в машинные программы, основываются на этом положении.

Проиллюстрируем сказанное несколькими примерами.

[1] Безусловные таблицы будем относить к таблицам типа P.

1) таблица на рис. 8 относится к типу P, так как R1 и R2 отличаются во втором, R_1 и R_3, а также R_2 и R_3 в первом условиях;

2) таблица на рис. 1 также относится к типу P;

3) таблица на рис. 3, однако, не относится к типу P, так как, например, R1 и R5 не различаются ни в одном из соответствующих условий;

4) на рис. 9 приведены таблица TP9 и ее частные таблицы по второму условию. В силу того, что b_{21} = «—», правило R1 входит в обе частные таблицы.

TP9	R1	R2	R3	R4	E
B1 = 0	N	—	N	Y	
B2 = 0	—	Y	N	Y	
B3 = 0	Y	N	N	—	
	M1	M2	M3	M4	M5

а)

TP9Y2	R1	R2	R4	E
B1 = 0	N	—	Y	
B3 = 0	Y	N	—	
	M1	M2	M4	M5

б)

TP9N2	R1	R3	E
B1 = 0	N	N	
B3 = 0	Y	N	
	M1	M3	M5

в)

Рис. 9. Простая TP (*а*) с двумя частными таблицами, выделенными по второму условию (*б* и *в*).

На основании сказанного можно сформулировать следующие утверждения:

а) Таблица решений типа P с n условиями и m правилами только тогда формально полна, когда

$$\sum_{i=1}^{m} 2^{k_i} = 2^n. \qquad (1)$$

При этом k_i означает число пробелов (прочерков) в i-м правиле. Применяя эти выражения к таблице на рис. 1, получаем:

$$2^1+2^1+2^2+2^2+2^2+2^2=4\cdot 2^2+2^4=2^4+2^4=2^5.$$

Следовательно, таблица формально полна.

б) Таблица решений только формально полна, когда таблицы TPYk и TPNk для произвольного натурального k; $1 \leqslant k \leqslant n$ формально полны (см. задачу 3).

Не всегда вышеприведенное определение полноты имеет практический смысл. Покажем это на простейшем примере. Таблица на рис. 10 не является формально полной, так как отсутствует правило для ситуации (Y, Y). Так как ситуация RHO<1 и RHO>2 одновременно не может иметь место, вряд ли можно эту таблицу назвать неполной.

TP12	R1	R2	R3
RHO < 1	Y	N	N
RHO > 2	N	Y	N
GOTO10	×	×	
GOTO20			×

Рис. 10. Логически полная ТР.

Таблица решений называется логически полной, если она всем логически истинным ситуациям ставит в соответствие некоторые действия.

Каждая формально полная таблица является, конечно, и логически полной. Обратное, как показывает последний пример, неверно. Дальнейшие примеры приведены в упражнениях.

Практически более значимым является понятие логической полноты, однако формальная полнота легче проверяется. Нетрудно написать программу, осуществляющую проверку на формальную полноту и выдающую ситуации, для которых не существует правил. Логическая полнота связана с семантикой условий. Ее проверка с помощью ЭВМ возможна только тогда, когда ограничиваются полностью определенным классом условий.

Читатель ни в коем случае не должен понимать изложенное так, что следует все таблицы проверять на полноту. Иногда такая проверка интересна и полезна, но иногда она является лишней. Так, многие ТР имеют структуры типа: если R1, то M1, если R2, то M2, если Rm, то Mm, а во всех остальных случаях, которые часто бывают совсем неизвестны, Mm+1. Последнее действие при этом по большей части является особым действием (сигнал об ошибке и т. п.). В этих случаях всегда выписывают таблицы с правилом ИНАЧЕ.

4. ЭКВИВАЛЕНТНЫЕ ТАБЛИЦЫ РЕШЕНИЙ

Соображения, изложенные в этом параграфе, играют важную роль при переводе ТР в эффективные программы для ЭВМ. Как мы уже видели, одна и та же задача может быть описана различными ТР. Естественно поэтому ввести понятие эквивалентности, построить эквивалентные преобразования и по возможности обрисовать естественные, наиболее простые таблицы в одном классе эквивалентности.

Две таблицы решений TPx, TPy *с одинаковыми условиями называются эквивалентными* (TPx∼TPy), *если они ставят в соответствие одинаковым ситуациям одинаковые действия, т. е.*

$$TPx(S)=TPy(S).$$

Для доказательства эквивалентности двух таблиц решений достаточно вычислить и попарно сравнить их «значения» для всех различных ситуаций.

Часто встречаются более простые возможности. Рассмотрим рис. 11. Первые два правила TP13 (рис. 11,*а*) имеют одно и то же действие и отличаются лишь третьим условием. Таким образом, можно b_{31} заменить прочерком и вычеркнуть R2 и третье условие. Получим таблицу TP14 (рис. 11,*б*). Обе таблицы эквивалентны.

TP13	R1	R2	R3
A > 0	Y	Y	N
B > 0	Y	Y	—
C > 0	N	Y	—
	M1	M1	M2

а)

TP14	R1	R2
A > 0	Y	N
B > 0	Y	—
	M1	M2

б)

Рис. 11. Таблица решений с лишним третьим условием (*а*) и соответствующая эквивалентная таблица (*б*).

При больших таблицах подобные операции могут оказаться несколько затруднительными. Укажем поэтому некоторые преобразования, которые позволяют получить из заданных таблиц эквивалентные таблицы решений:

а) перестановка условий (см. сноску на с. 8);

б) перестановка правил, если таблица принадлежит к типу P (это предположение достаточно, не не необходимо);

в) выбрасывание и добавление лишних правил и неработающих условий;

г) правила Ri и Rk могут отличаться только входами b_{ji} и b_{jk} (т. е. вызывают одинаковые действия), помимо этого, таблица больше не содержит лишних правил. Тогда получим эквивалентную таблицу заменой b_{ji} «—» (прочерк) и вычеркиванием Rk (рис. 11);

д) таблица решений не содержит лишних правил.

Правило Ri управляет ситуациями S1, ..., Sg, ..., Sk, а правило Rj — ситуациями S1, ..., Sg, ..., Se (i<j, e>g).

Описанный случай встречается как раз тогда, когда правила Ri и Rj не различаются ни одним из соответствующих значений условий. Эквивалентную таблицу TP′ получают путем замены правила Rj e—g правилами Rg+1, ..., Re, которые в указателе условий совпадают с Sg+1, ..., Se и имеют действия, как при Rj.

К TP′ можно затем применить метод преобразования, описанный в п. «г», если имеют место упомянутые там условия. Правило

Ri и полученные из Rj правила попарно различаются по меньшей мере одним условием, не содержащим прочерка.

Этот процесс может продолжаться до тех пор, пока не будет реализована таблица, в которой все правила имеют хотя бы одно отличие в условиях.

Таким образом показано, что для каждой ТР с ограниченными (простыми) входами существует эквивалентная ей таблица типа Р, т. е. таблица с произвольно переставляемыми правилами.

ТР15	R1	R2	R3	R4
C1 = 0	Y	Y	N	—
C2 = 0	N	N	N	N
C3 = 0	—	—	N	N
C4 = 0	Y	—	—	Y
—	M1	M2	M3	M4

а)

ТР16	R1	R2	R3
C1 = 0	Y	Y	N
C2 = 0	N	N	N
C3 = 0	—	—	N
C4 = 0	Y	N	—
	M1	M2	M3

б)

Рис. 12. Таблица решений с лишним правилом R4 (*а*) и соответствующая ей эквивалентная таблица (*б*).

ТР17	R1	R2	R3	R4	R5	E
ЦИСТ ДЛЯ ВОДЫ	Y	Y	—	—	—	
ЦИСТ ДЛЯ НЕФТИ	—	—	Y	Y	—	
ЦИСТ ДЛЯ КИСЛОТЫ	—	—	—	—	Y	
ДАВЛ ⩽ 5	Y	—	—	—	—	
ДАВЛ ⩽ 10	—	Y	—	—	—	
ДАВЛ ⩽ 8	—	—	Y	—	—	
ВЕНТ V1	×					
ВЕНТ V2		×				
ВЕНТ V3			×			
ВЕНТ V4				×		
ВЕНТ V5					×	
НОКОНСТ						×

Рис. 13. Таблица решений с ограниченными входами, эквивалентная таблице на рис. 4.

Поясним сказанное следующим примером. В ТР15 (рис. 12,*а*) правило R1 включает ситуация S1=(Y, N, Y, Y) и S2=(Y, N, N, Y); R2, кроме S1 и S2, включает также S3=(Y, N, Y, N) и S4=(Y, N, N, N). Заменим R2 двумя правилами, содержащими S3 и S4 в указателе условий. Оба новых правила объединим в соответствии с п. «г» и получим R2 из ТР16 (рис. 12,*б*). Таким же образом поступим с R1 и R4 и найдем, что R4 заменяется на (N, N, N, Y). Но это правило в силу того, что R3=ТР15(N, N, N, «—») является лишним, может быть исключено. Таким образом получена ТР16, которая является таблицей типа Р.

В заключение покажем, как из таблицы с расширенными входами получается эквивалентная таблица с ограниченными входами. Рассмотрим для этого таблицу на рис. 4. Эта таблица содержит шесть различных расширенных входов. Из каждого из этих входов образуем новое условие. Опишем правила с помощью Y и прочерков в указателе условий, как показано в ТР17 (рис. 13). Число правил при таком преобразовании сохранится.

5. ТАБЛИЦЫ РЕШЕНИЙ И БЛОК-СХЕМЫ

Как правило, конечной целью является получение из ТР (соответственно из системы таких таблиц) программы для ЭВМ. Программы обычно представляются блок-схемами, и возникает вопрос о связи между блок-схемами и ТР.

Справедливо утверждение:

Из каждой полной таблицы решений или из системы таких таблиц можно получить эквивалентную блок-схему и, наоборот, для каждой блок-схемы можно найти эквивалентную полную таблицу решений или систему таких таблиц.

Если для множества блок-схем установить понятие эквивалентности, то сказанное будет означать, что каждому классу эквивалентных полных ТР можно поставить в соответствие некоторый класс эквивалентных блок-схем, и наоборот. Практический интерес представляют только те методы, с помощью которых можно осуществить такой переход.

Рассмотрим следующий метод, состоящий из двух шагов:

ШАГ 1. Таблица решений безусловна или эквивалентна безусловной таблице[1]. Тогда следует произвольно упорядочить действия, соответствующие правилу R1 в блок-схеме, и переход закончен. В противном случае следует перейти к шагу 2.

ШАГ 2. Таблица решений содержит n условий ($n \geqslant 1$). Тогда следует начать блок-схему с проверки произвольного условия Bk и провести ветвь Y к частной таблице TPYk и ветвь N — к TPNk. Для каждой из полученных частных таблиц следует повторить операцию. Так как число условий в таблицах TPYk и TPNk меньше, чем в первоначальной ТР, алгоритм завершится после выполнения конечного числа шагов. В результате получается блок-схема, эквивалентная первоначальной ТР.

Рассмотрим для пояснения примеры.

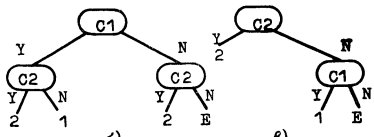
---

[1] В качестве примера любая ТР, у которой первое правило ни в каком условии не содержит определенных входов, отличных от прочерков, эквивалентна безусловной таблице.

1. На рис. 14 изображены ТР (рис. 14,*а*) с двумя условиями и две соответствующие им блок-схемы программ (рис. 14,*б* и *в*). Начиная с проверки первого условия, получим блок-схему *б*, в противном случае — *в*. Уже этот простейший пример показывает, что «качество» полученных блок-схем программ, а следовательно, и результирующая программа зависят от последовательности, в которой проверяются условия.

ТР18	R1	R2
C1 = 0	Y	—
C2 = 0	N	Y
ТРА =	1	2

а)



б) *в*)

Рис. 14. Таблица решений (*а*) с двумя эквивалентными блок-схемами (*б* и *в*).

2. Более сложный пример приведен на рис. 15. Он получен в соответствии с описанным выше способом из ТР5 (см. рис. 5) при k=1. Для облегчения перепроверки даны первые частные таблицы ТР5Y1 и ТР5N1 (см. определение на с. 14). Для этого примера не существует блок-схемы, которая содержала бы меньшее число проверок. Это не удивительно, так как условия в ТР5 отличаются друг от друга лишь несущественно (в каждом условии три Y и три «—»). Поэтому ничего не изменится, если начать блок-схему с проверки другого условия, как в первом примере.

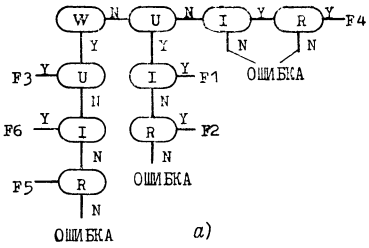
Рассмотрим обратную задачу.

Как получить по блок-схеме эквивалентную ей ТР или систему таких таблиц? Рассмотрим блок-схему программы, приведенную на рис. 15. Эта блок-схема кончается десятью действиями (F1, ..., F6 и 4 раза ОШИБКА). К каждому из этих действий от начала блок-схемы идет только один путь, который при рассмотрении в обратном направлении соответствует одному правилу ТР. Сведем эти правила в таблицу. Простоты ради мы выбираем последовательность условий и правил, соответствующую рис. 5. В результате получена ТР19, изображенная на рис. 16.

Эта таблица относится к таблицам типа Р. Размер ТР показывает, что последовательность правил, которую мы вначале выбирали произвольно, фактически не имеет значения. Кроме того, таблица формально полна.

Для примера применим соотношение (1). Отсюда следует, что правила с одним общим действием могут быть объединены в одно правило ИНАЧЕ. В нашем случае это действие ОШИБКА (рис. 16). Доказательство эквивалентности таблиц ТР5 и ТР19 представим читателю.

Все сказанное позволяет сделать вывод о том, что существует элементарный метод превращения ТР в блок-схемы программ. Правда, метод этот не всегда приводит к оптимальным результатам.



а)

TP5Y1	R1	R2	R3	R4	R5	R6	E
U	Y	—	—	Y	Y	—	
I	—	Y	—	Y	—	Y	
R	—	—	Y	—	Y	Y	
ИСПОЛЬЗУЙ	F3	F6	F5	F1	F2	F4	
ОШИБКА							×

б)

TP5N1	R4	R5	R6	E
U	Y	Y	—	
I	Y	—	Y	
R	—	Y	Y	
ИСПОЛЬЗУЙ	F1	F2	F4	
ОШИБКА				×

в)

Рис. 15. Блок-схема к TP на рис. 5 (*а*) и частные таблицы для ветвей Y (*б*) и N (*в*) после проверки первого условия. (Для сохранения связи с TP5 мы отступили от обычной нумерации правил.)

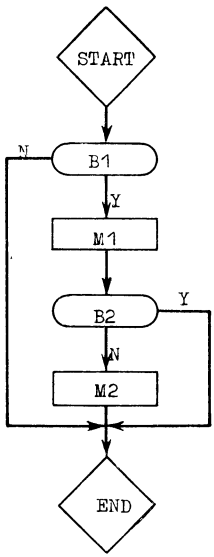
TP19	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	E
N	Y	Y	Y	N	N	N	Y	N	N	N	
U	Y	N	N	Y	Y	N	N	Y	N	N	
I	—	Y	N	Y	N	Y	N	N	Y	N	
R	—	—	Y	—	Y	Y	N	N	N	—	
ИСПОЛЬЗУЙ	F3	F6	F5	F1	F2	F4					
ОШИБКА							X	X	X	X	X

Рис. 16. Таблица решений типа P, эквивалентная TP5 на рис. 5.

Поэтому мы вернемся к этой проблеме в гл. 3. Кроме того, из примера видно, как некоторые блок-схемы (правда, довольно простые) можно представить в виде TP. В дальнейшем изложении этот метод будет развит.

6. ОБЪЕДИНЕНИЕ И РАЗЛОЖЕНИЕ ТАБЛИЦ РЕШЕНИЙ

Нетривиальная практическая проблема обычно характеризуется невозможностью представления ее в виде одной TP. Либо эта проблема оказывается настолько сложной и обширной, что одна таблица была бы необозримой, либо имеются основания, которые делают такое представление принципиально невозможным.

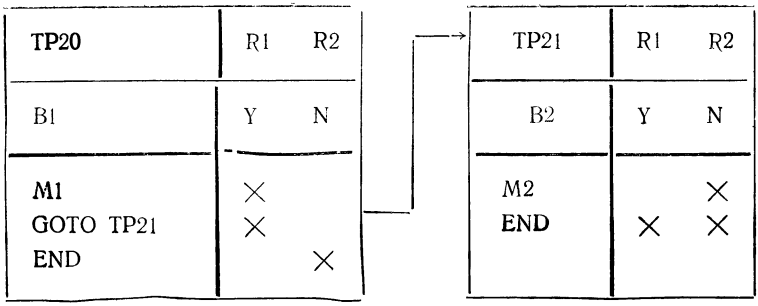


Рассмотрим для этого рис. 17. Любая попытка списать эту блок-схему с помощью одной эквивалентной TP оказывается бесплодной, так как проверка B1 зависит от действия (операции) M2. Но нетрудно эту задачу представить в виде двух связанных таблиц (рис. 18).

Рассмотрим несколько более сложный пример (рис. 19), который ставит перед нами новую задачу.

В этой блок-схеме в зависимости от трех условий должны выполняться два действия, между которыми помещена подпрограмма (ПП). Затем следуют две последующие проверки. Для того, чтобы иметь возможность сформулировать эту задачу в виде TP, введем понятие *замкнутой таблицы.*

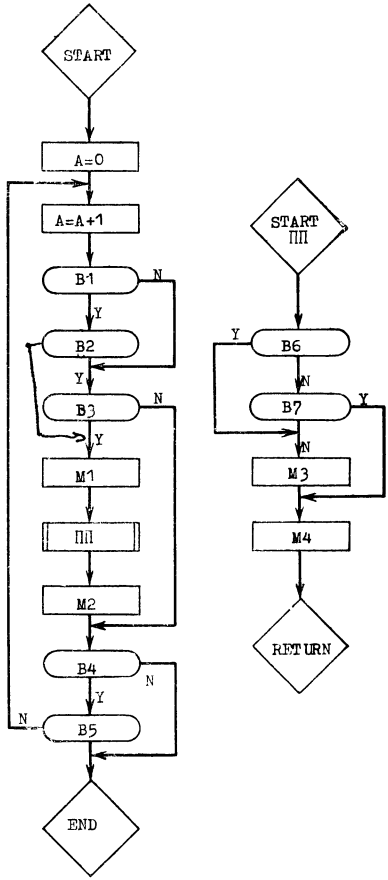
Рис. 17. Пример блок-схемы.



а) *б*)

Рис. 18. Две связанные ТР, эквивалентные блок-схеме, изображенной на рис. 17.

Рис. 19. Блок-схема программы с подпрограммой. (На схеме основной программы не показан переход от проверки B2 на N к входу оператора M1.)



Замкнутая таблица решений может быть вызвана с помощью любого действия обычной таблицы решений.

После обработки ее выполняется следующее действие, отмеченное знаком × в выбранном правиле, и с этого места продолжается обработка.

Содержательно замкнутая таблица соответствует ПП при программировании. Естественно, что такая таблица должна быть формально обозначена. Это может быть осуществлено с помощью оператора обращения [GOTO для обычной таблицы и DO для замкнутой таблицы (в качестве последнего действия во всех правилах замкнутой таблицы помещается оператор RETURN)] или с помощью обозначения замкнутой таблицы специальным именем.

Первый способ предпочтительнее. После сказанного представление нашей задачи в виде ТР не составляет труда (рис. 20).

Таблица ТР22 является безусловной таблицей, которую удобно использовать для ввода и установления значений исходных данных. Таблица ТР23 реализует первую группу условий. Она формально полна, принадлежит к типу P и может быть упрощена путем перестановки правил R1 R5 и последующей замены правил R1, R2 и R3 правилом ИНАЧЕ. Замкнутая таблица ТР25 также может быть упрощена. Таблица ТР24 уже обработана в соответствии с описанным методом.

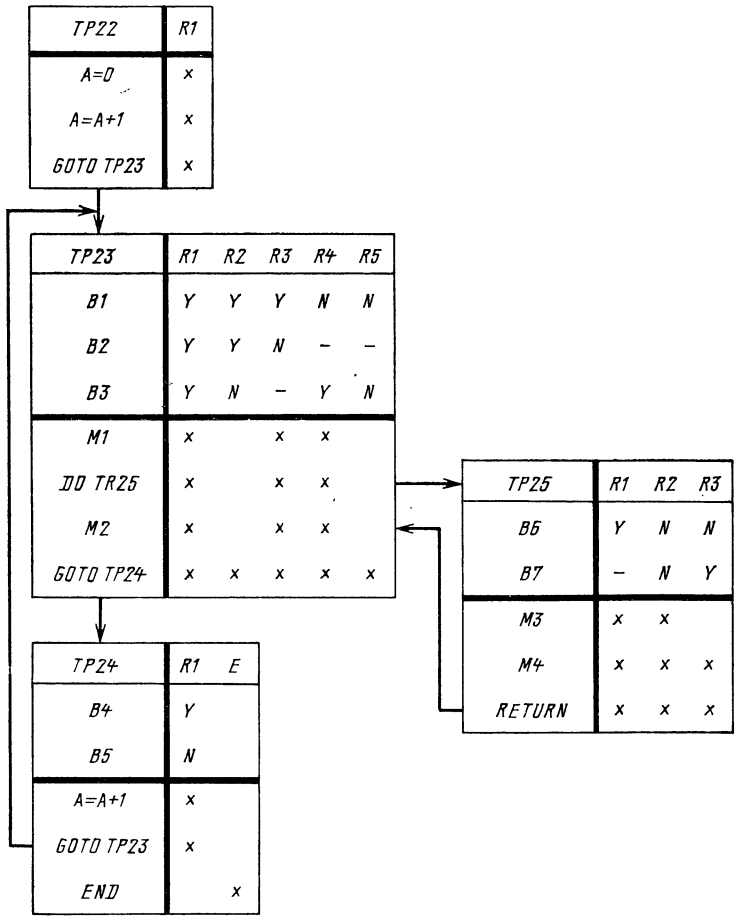


Рис. 20. Система ТР, эквивалентная блок-схеме на рис. 19.

Эти два примера показывают, как описать данную задачу системой связанных между собой ТР. Простоты ради мы начали здесь с блок-схемы. Это, однако, несущественно. Словесно сформулирован-

ная задача может быть представлена с помощью таблиц. Естественно, что это осуществляется не так просто, как в нашем примере. К этой проблеме мы вернемся в гл. 2. Описанный метод позволяет всегда обходиться достаточно малыми таблицами. Несмотря на это, даже опытный программист не всегда может избежать больших необозримых таблиц. В таких случаях следует порой обращаться к применению методов Chapin'a [2] и Strunz'a [3].

Поясним их на примере. Анализ задачи показал, что пять действий и конечная программа находятся в зависимости от шести условий. При этом установлено:

а) Если выполняются B2 и B6 и не выполняется B4, то должно исполняться M1.

б) Если B2 и B6 не выполняются, то следует перейти к M5.

в) Если B1 выполняется, а B3 — нет, то переходим к M2.

г) Если B3 не выполняется, должно быть реализовано M3, если не выполняется B5, реализуется M4.

д) Если в некоторой ситуации должны выполняться несколько действий, то естественный порядок не должен быть нарушен, т. е. Mi выполняется ранее Mk, если $i<k$.

е) Конечная подпрограмма должна выполняться в каждой ситуации, но никогда не выполняться перед M1 ... M5.

Опишем это простой таблицей, изображенной на рис. 21.

Читатель видит, что это изображение не является ТР (чтобы это подчеркнуть формально, действия помещены в первой строке).

Если смотреть на эту таблицу, как на ТР, то ситуации S= =(Y, Y, N, N, N, Y) соответствовало бы только одно действие M1, в то время как требуются действия M1, M2, M3, M4. Нетрудно (но довольно скучно) из этой таблицы получить ТР. Для этого следует выписать все 64 правила таблицы решений с шестью условиями и поставить знаки × в каждом правиле для необходимых действий в нужной последовательности. После этого следовало бы попытаться методами, описанными в § 4, преобразовать таблицу к более удобному виду.

Существует и другой, более простой, способ. Действия M1 и M5 зависят только от условий B2, B4 и B6, и эти условия не влияют ни на какие другие действия. Можно написать таблицу с этими тремя условиями B2, B4 и B6, а остальные условия перенести

	M1	M2	M3	M4	M5
B1		Y			
B2	Y				N
B3		N	N		
B4	N				
B5				N	
B6	Y				N

Рис. 21. Табличное упорядочивание условий и действий описанной задачи.

TP26	R1	R2	E
B2	Y	N	
B4	N	—	
B6	Y	N	
M1	X		
DO TP27	X	X	X
M5		X	
END	X	X	X

TP27	R1	R2	R3	R4	R5	R6
B1	—	—	Y	Y	N	N
B3	Y	Y	N	N	N	N
B5	Y	N	Y	N	Y	N
M2			X	X		
M3			X	X	X	X
M4		X		X		X
RETURN	X	X	X	X	X	X

Рис. 22. Две связанные TP, из которых одна (*б*) замкнута.

в другую таблицу и обе таблицы связать. Таким образом, получены таблицы TP26 и TP27, представленные на рис. 22. Указатель действий таблицы TP26 очень интересен. Первым действием второго правила является вызов замкнутой таблицы и именно TP27, которая содержит действия M2, M3 и M4. Сразу после возврата будет выделено M5. Таким образом реализуется требование п. «д». Описанным методом можно представить в виде таблиц решений любую «разумную блок-схему» и наоборот. (Точная формулировка требуемых вспомогательных средств следует из теории графов, что вывело бы нас за рамки настоящей книги.)

Таким образом, установлено, когда представление с помощью таблиц предпочтительнее. Таблицы решений имеют преимущества тогда, когда решение задачи зависит от многочисленных, связанных между собой условий, так как в этом случае формальное соображение о полноте обеспечивает дополнительную гарантию надежности результата. Таблицы обозримы и допускают легкие изменения. Дальнейшие преимущества их будут ясно показаны в следующей главе. Следует сказать, что TP неэффективны, если задача не допускает легкого разделения между действиями и условиями (по крайней мере по группам) или действий намного больше. В этих случаях следует, как и ранее, использовать классические вспомогательные средства.

7. ЗАДАЧИ ДЛЯ УПРАЖНЕНИЙ

1. Пусть дана таблица TP9 (см. рис. 9).

а) Определите TP9(Y, Y, N) и TP9(Y, N, Y).

б) Для каких ситуаций работает правило ИНАЧЕ?

в) Что произойдет, если поменять местами правила R2 и R4?

2. Составьте TP с расширенными входами для определения максимума трех различных вещественных чисел.

3. Обоснуйте критерии полноты, приведенные в § 3.

4. Составить из TP с расширенными входами задачи 2 простую таблицу способом, описанным в § 4. Будет ли полученная TP:

а) типа P?

TP28	R1	R2	R3	R4	R5
$A < 0$	Y	Y	N	N	Y
$B \geqslant 0$	Y	N	Y	N	N
$B < -1$	—	Y	N	—	N
C =	1	2	3	4	5

Рис. 23. Таблица решений к задаче 5.

б) формально полной?
в) логически полной?

5. Является ли TP28 (рис. 23) логически полной?

6. Составить TP, эквивалентную блок-схеме, изображенной на рис. 24, и упростить ее способом, описанным в § 4. Полученную таблицу представить снова в виде блок-схемы.

7. Составить TP с минимальным числом правил для скорости резания C при нарезке резьбы. Скорость резания зависит от материала (в данном случае алюминий, вязкая и хрупкая латунь) и от свойств инструмента (резца). При нарезке резцами из быстрорежущей стали обычно выбирают C=25 м/мин; только для хрупкой латуни скорость не может быть выше 16 м/мин. Скорость 16 м/мин допускается также при нормальном инструменте для хрупкой латуни и алюминия. Вязкая латунь должна обрабатываться инструментом из нормальной рабочей стали со скоростью не выше C= =10 м/мин. Другие материалы не используются. В качестве условий выбрать:

SSS, т. е. изготовляется ли инструмент из быстрорежущей стали?

MS, т. е. является ли материалом латунь?

SMS, т. е. является ли материалом хрупкая латунь?

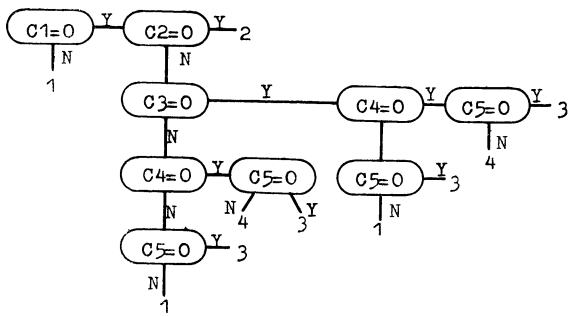


Рис. 24. Блок-схема к задаче 6.

8. Доказать, что частные таблицы таблицы типа Р также являются таблицами типа Р.

9. Описать задачу, заданную в § 6 и на рис. 21, с помощью только двух ТР, содержащих соответственно (В1, В3, М2, М3) и (В2, В4, В5, М1, М4, М5).

ГЛАВА ВТОРАЯ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ФОРМИРОВАНИЮ ТАБЛИЦ РЕШЕНИЙ

Эффективность использования ТР существенно зависит от целого ряда факторов.

Успехи чаще всего имеют место, если аналитик:

а) убежден в преимуществах техники работы с ТР по сравнению с техникой работы с блок-схемами;

б) правильно выбрал область применений;

в) применяет эффективные методы формирования ТР;

г) имеет в своем распоряжении средства обработки данных (ЭВМ и программное обеспечение).

Пункты «а»—«в» будут подробнее рассмотрены в следующих параграфах. Дальнейшие положения по п. «г» приведены в гл. 4.

8. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ТАБЛИЦ РЕШЕНИЙ И БЛОК-СХЕМ

Современная практика решения прикладных задач имеет один существенный недостаток — постановка задачи формулируется, как правило, в предметной области. Для этого используются словесное описание данных и грубые блок-схемы процессов их обработки. Дальнейшее решение задачи осуществляется вычислительными центрами. Встречающиеся на этом стыке трудности взаимопонимания ведут к неясным рабочим определениям и описаниям программы. Следствием всего этого являются ошибки в программах, которые могут быть устранены лишь в результате длительного дополнительного анализа и обсуждений. Использование ТР дает возможность аналитику полно и однозначно сформулировать постановку задачи.

Программисту при этом легко перевести эти таблицы в форму, удовлетворяющую требованиям ЭВМ. Таблица решений является надежным средством организации взаимопонимания между прикладниками и программистами. Ответственность за анализ задачи переносится на аналитика.

Кроме того, использование ТР позволяет проверку многих условий и соответствующее упорядочивание действий осуществить в комплексе (в блок-схемах это происходит последовательно, отдельными шагами), причем представление остается обозримым и легко может быть перепроверено на полноту.

Большое преимущество блок-схем заключается в том, что они по сравнению с ТР более распространены. Они изучаются в высших и профессиональных учебных заведениях и узаконены многочисленными стандартами.

Соответствующие публикации об опыте использования ТР и обучение (с изданием соответствующих стандартов) должны позволить ТР сделаться вспомогательным средством, равноценным блок-схемам.

9. ОБЛАСТИ ПРИМЕНЕНИЯ ТАБЛИЦ РЕШЕНИЙ

Уже с конца 50-х годов ТР используются для рационализации работы с ЭВМ. Возможности внедрения не ограничиваются областью обработки данных, так как таблицы решений являются средством, охватывающим систематические связи, позволяющим документировать и осуществлять автоматическую обработку.

Следует различать две основные области применения ТР:

а) анализ и документирование задачи;

б) программирование.

Наибольший эффект достигается, когда обе области связаны, т. е. когда задача для автоматической обработки анализируется и документируется с помощью ЭВМ. Перечислим в общем виде преимущества, достигаемые при использовании ТР:

а) компактная, обозримая форма анализа задачи;

б) единое (вместо последовательного) описание задачи;

в) легкая изучаемость;

г) требуемая ясность представления задачи, позволяющая легко установить, где отсутствует информация;

д) простые возможности контроля полноты и содержательной корректности;

е) простая актуализация.

Подробный обзор многочисленных областей применения приведен в [4].

10. ФОРМИРОВАНИЕ ТАБЛИЦ РЕШЕНИЙ

Известно очень мало работ (см., например, [5]), которые посвящены этой проблеме. Довольно трудно изложить общий порядок действий для формирования ТР. Для отдельных случаев применений всегда указываются различия, так как:

а) эффективными могут быть в некоторых случаях простые, в некоторых расширенные ТР;

б) исследуемый процесс может быть известным и оказаться приемлемым или должен заново разрабатываться;

в) использование таблиц может служить либо для систематизации, либо для последующей автоматизированной обработки.

Рассмотренные далее шесть шагов, поясняемые на практическом примере, могут быть использованы при первоначальном представлении и обработке задач с помощью ТР.

Для первого формирования ТР следует выбирать частные задачи, для которых:

а) рассматриваемыей объект (задача) содержательно хорошо известен;

б) необходимые данные почти полностью получены.

После успешного решения первого примера каждый аналитик для последующих примеров и практических случаев может использовать свой собственный метод, который, однако, будет представлять собой лишь специфическую модификацию перечисленных ниже шести шагов.

ШАГ 1. Ограничение задач. Большая задача, например, представление деятельности по технической подготовке производства некоторого предприятия с ее вариантами и взаимосвязями, не может быть с первого раза описана с помощью ТР. Поэтому необходимо осмысленно установить границу задачи. Этот шаг очень важен и должен постоянно уточняться в зависимости от текущего состояния обработки. Если задача выбрана слишком большой, решение оказывается сразу же поставленным под вопрос. Если границы слишком узки, решение не дает новых знаний (не приносит познавательной пользы) или эффекта для рационализации. Поэтому следует организовать процесс так, чтобы задача была целесообразно расширяема и чтобы уже составленные ТР могли быть объединены.

Пример 5. На предприятии изготавливаются четыре типа внутришлифовальных станков. Затраты на оснастку их оборудованием, обусловленным потребителями, особенно велики. Таким оборудованием, например, являются:

а) зажимное устройство;
б) распределитель охлаждающей жидкости;
в) шлифовальный круг;
г) измерительное устройство и т. д.

Выбор обусловленного потребителями оборудования может быть рационализирован с помощью ТР.

Сначала ограничимся двумя типами шлифовальных станков (МТА и МТВ).

Оба типа шлифовальных станков используются главным образом для обработки подшипников качения. Оборудование, выбираемое потребителем, должно точно соответствовать изготавливаемым потребителями обоймам подшипников. Спрос на подобные станки постоянно растет.

С помощью алгоритма выбора в форме ТР можно определить для 18 типов групп обойм подшипников оборудование, необходимое потребителям. Соответствующие этим обоймам размеры колец подшипников ограничены следующими значениями:

Ширина кольца RBR, мм	8—50
Внутренний диаметр DI, мм	$\leqslant$80
Внешний диаметр DA, мм	$\leqslant$120
Общая масса, г	$\leqslant$800

В этих пределах заключено около 95% всех требований потребителей. Для остальных типов обойм и колец, которые выходят за пределы этих границ в бо́льшую или меньшую сторону, выбор, если он вообще возможен, осуществляется по-прежнему. Эти ограничения должны быть окончательно установлены в процессе алгоритмизации с помощью ТР. Так как границы для обработки приблизительно установлены, анализ производства может быть более целенаправленным и точным.

Шаг 2. Представление отдельных решений и их связи. Реальные решения редко представимы отдельными ТР. Должно иметься много элементарных решений и выполняемых действий, чтобы найти одно решение, и должны быть представлены элементарные решения, а также связи их друг с другом. Те элементарные решения, которые представляют собой условия для последующих элементарных решений, должны быть определены в первую очередь.

Основной целью этих шагов соответственно является разложение общей задачи на относительно самостоятельные элементарные

решения. Эти элементарные решения являются первой основой для обработки отдельных ТР. Связи и оценка издержек, требуемых для представления элементарных решений с помощью ТР, являются основанием для того, чтобы еще раз уточнить границы задач.

Пример 6 *. Существенным представляется элементарное решение по выбору части или узла оборудования, связанное с запросами пользователей. Так как задача очень сложна (около 80 элементарных решений), следует отказаться от ее полного представления. В основном процессе поэтому описывается только выбор типов изме-

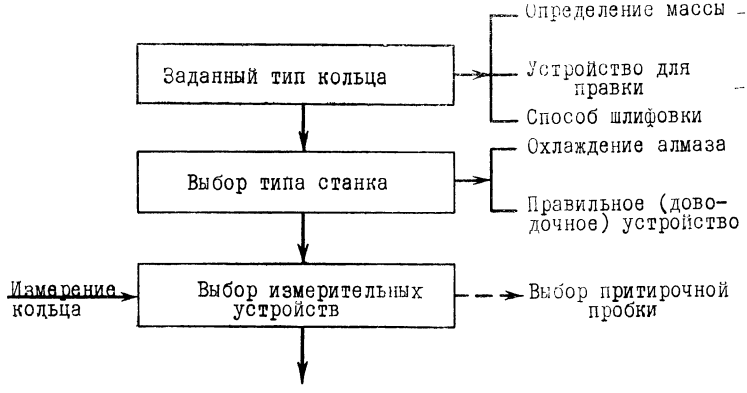


Рис. 25. Место измерительных индикаторов в технологическом процессе.

рительных устройств (датчиков). Это элементарное решение существенно зависит от типа кольца (RT), размеров обоймы подшипника и типа станка. В пределах всей системы выбор измерительных устройств занимает положение в соответствии с рис. 25.

Шаг 3. Характеристика отдельных решений. Для отдельного решения должны быть найдены все необходимые условия и установлены все практически возможные комбинации условий. Здесь предпочтителен лексикографический порядок (ср. пример в § 6). Работа с ТР с простыми входами имеет смысл, если число возможных комбинаций условий не слишком велико (рекомендация: число возможных правил не превосходит 32). В более сложных случаях (см. § 5) следует работать с расширенными таблицами. Тогда, однако, теряется преимущество простого рассмотрения полноты и несколько ограничивается обозримость.

При представлении хорошо известных процессов часто оказывается проще сначала выделить действия и затем упорядочить соответствующие условия.

Пример 7. На станках типа MTA изготавливаются кольца типов от RTI до RTVII. При этом могут оказаться нужными измерительные устройства типов от 01 до 05. Кольца типа RTVI требуют спе-

* Примеры 6—10 являются продолжением примера 5. — *Прим. ред.*

циальной конструкции измерительных устройств, кольца RTV требуют выбора измерительных устройств для станка МТВ. Для колец типов RTI—RTIII тип измерительного устройства зависит от ширины кольца и глубины желоба LBT, служащего для укрепления катящихся элементов, например шариков или роликов (табл. 1).

Таблица 1

Зависимость типа измерительного устройства от ширины кольца и глубины желоба для колец типов RTI — RTIII и RTVII

Ширина кольца RBR, мм	Глубина желобка LBT, мм	Тип измерительного устройства	Ширина кольца RBR, мм	Глубина желобка LBT, мм	Тип измерительного устройства
8—23	0—1	01	31—40	2—3	05
23—31	0—1	04	8—23	3—4	04
31—40	0—1	05	23—31	3—4	04
8—23	1—2	02	31—40	3—4	05
23—31	1—2	04	8—23	4—5	05
31—40	1—2	05	23—31	4—5	05
8—23	2—3	03	31—40	4—5	05
23—31	2—3	04			

Глубина желоба должна вычисляться из соотношений:

$$LBT = (DL - DI)/2 \qquad (А)$$

для RTI и RTIII;

$$LBT = (DL - XA)/2 \qquad (Б)$$

для RTII;

$$LBT = (XA + XB)\ SIN\ (\varphi) - XC \qquad (В)$$

для RTVII.

Ширину кольца для RTIII следует определять из соотношения

$$RBR = LBR(1 - COS\ (\varphi))/2. \qquad (Г)$$

Здесь DL, XA, XB, XC, LBR и φ — заданные размеры кольца.

Для RTIV выбор измерительного устройства зависит от глубины желоба (табл. 2). Глубина желоба определяется по формуле (А).

Таблица 2

Зависимость типа измерительного устройства от глубины желобка для колец типа RT IV

Глубина желобка LBT, мм	Измерительное устройство	Глубина желобка LBT, мм	Измерительное устройство
0—1	01	3—4	04
1—2	02	4—5	05
2—3	03		

Для станков типа В (МТВ) с возможными измерительными устройствами 06, 07, 08, 09 их выбор зависит главным образом от внутреннего диаметра DI кольца. Эти зависимости читатель может получить из ТР на рис. 30.

Полученные представления о существе вопроса позволяют выделить следующие условия для выбора измерительных устройств:

а) тип станка, МТА или МТВ;

б) тип колец: от RTI до RTXVIII;

в) ширина колец: от 8 до 23 мм; от 23 до 31 мм; от 31 до 40 мм;

г) глубина желоба: от 0 до 1 мм; от 1 до 2 мм, ..., от 4 до 5 мм;

д) внутренний диаметр: для выбора при МТВ.

Представление о возможных комбинациях условий может быть получено из ТР29—ТР32.

Шаг 4. Формирование таблиц решений. На данном шаге осмысленные комбинации условий должны быть поставлены в соответствие действиям. При этом должны быть представлены также и связи с другими ТР. Следует обратить внимание на то, чтобы были использованы все необходимые действия.

Если при выполнении третьего шага в основу были положены действия, то упорядочены должны быть комбинации условий. При этом надо следить, чтобы одному действию или одному комплекту действий почти всегда соответствовало несколько комбинаций условий.

ТР29	*R*1	*R*2	*R*3	*R*4
RT $\leqslant$ VII	Y	Y	N	N
RT $\leqslant$ XVIII	Y	N	Y	N
MT = A	×			
MT = B			×	
ОШИБКА		×		
СПЕЦКОНСТ				×

Рис. 26. Таблица решений к примеру 8 (выбор типа станка).

Пример 8. Представление связей в одной ТР уже при 30 условиях было бы необозримым. Поэтому рекомендуется выбор типов станков, выбор типов колец, упорядочивание измерительных устройств представлять раздельно.

С помощью ТР29 (рис. 26) осуществлен выбор типов станков в зависимости от заданных типов колец.

При двух условиях целесообразно представить все четыре возможные комбинации условий, хотя правило R2 практически не может иметь места.

ТР30	R1 R2 R3 R4 R5 R6 R7
RT = I	Y
RT = II	Y
RT = III	Y
RT = IV	Y
RT = V	Y
RT = VI	Y
RT = VII	Y
LBT СЧИТАТЬ ПО ФОРМУЛЕ (А)	× × ×
LBT СЧИТАТЬ ПО ФОРМУЛЕ (Б)	×
LBT СЧИТАТЬ ПО ФОРМУЛЕ (В)	×
RBR СЧИТАТЬ ПО ФОРМУЛЕ (Г)	×
СПЕЦКОНСТ ИЗМЕРИТЕЛЬНОГО УСТРОЙСТВА	×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО ДЛЯ МВТ	×

Рис. 27. К расчету недостающих размеров колец, обрабатываемых на станках МТА.

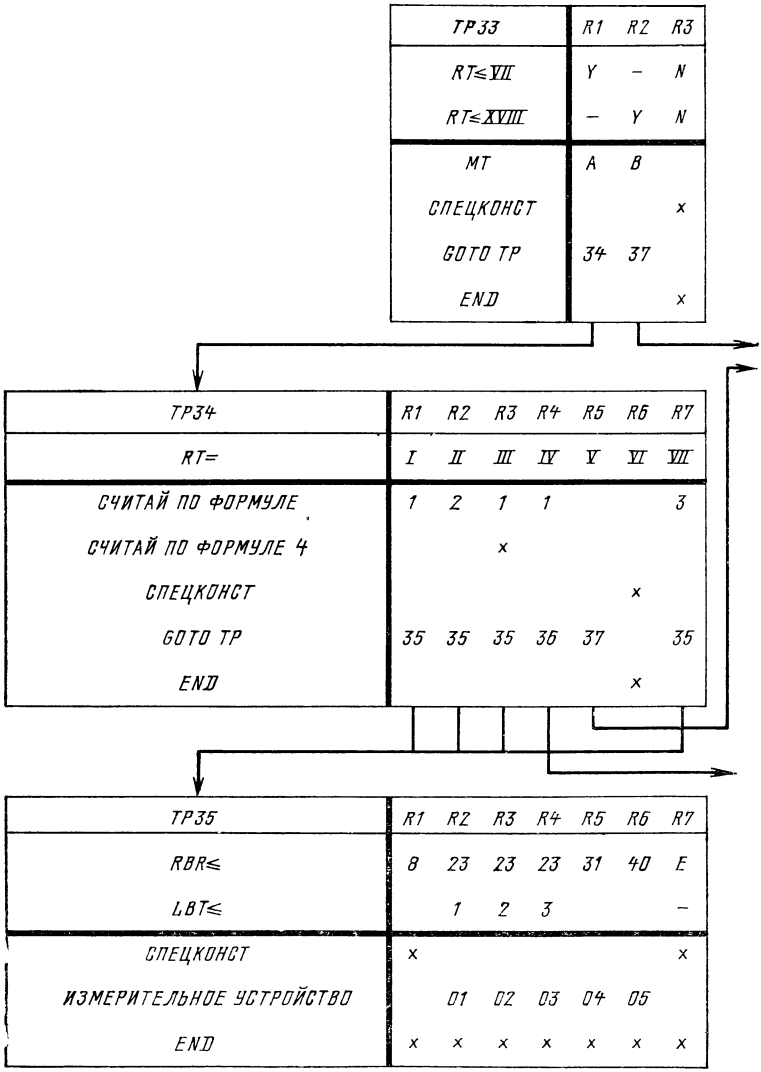
ТР32	R1 R2 R3 R4 R5 R6 R7 R8
RBR < 8	Y
RBR > 50	Y
LBT ⩽ 1	Y
LBT ⩽ 2	Y
LBT ⩽ 3	Y
LBT ⩽ 4	Y
LBT ⩽ 5	Y N
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО СПЕЦКОНСТ	× ×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 01	×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 02	×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 03	×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 04	×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 05	×

Рис. 29. Выбор измерительных индикаторов для колец типа RTIV.

ТР31	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18
RBR < 8	Y																	
RBR ⩽ 23		Y	Y	Y	Y	Y												
RBR ⩽ 31							Y	Y	Y	Y	Y							
RBR ⩽ 40												Y	Y	Y	Y	Y	N	
LBT ⩽ 1		Y					Y					Y						
LBT ⩽ 2			Y					Y					Y					
LBT ⩽ 3				Y					Y					Y				
LBT ⩽ 4					Y					Y					Y			
LBT ⩽ 5						Y					Y					Y		N
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО СПЕЦКОНСТ	×															×		×
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 01		×																
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 02			×															
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 03				×														
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 04					×		×	×	×	×								
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО 05						×					×	×	×	×	×	×		

Рис. 28. Выбор измерительных индикаторов для колец типов RTI, RTII, RTIII, RTIV.

Способы расчета глубины желоба и при необходимости ширины для всех типов колец, обрабатываемых на станках МТА, показаны в ТР30 (рис. 27). Здесь сразу выбрана такая форма ТР, при которой представлены только осмысленные комбинации условий. Подобные диагональные таблицы в практике встречаются очень часто. Необходимые для расчетов формулы берутся из описания шага 3.



В ТР31 (рис. 28) упорядочен выбор вариантов измерительных устройств для колец типов RTI—RTIII и RTVII. В силу большего числа условий полное представление всех комбинаций условий не имеет смысла. Ограничение практически возможными комбинациями условий приводит к логически полной таблице. Лексикографический порядок комбинаций условий при этом сохраняется. Таблица ТР31 может быть упрощена.

Для колец типа RTIV упорядоченный выбор измерительных устройств представлен в ТР32 (рис. 29), для которой справедливо все, сказанное о ТР31. Выбор измерительных устройств для станков типа МТВ, а также для колец типа RTV будет представлен чуть ниже (см. рис. 30).

Шаг 5. Связывание таблиц решений. Отдельные ТР должны быть связаны друг с другом при помощи вводимых управляющих операторов (GOTO, DO, RETURN). При этом они должны быть еще раз проверены на содержательную правильность и полноту и при возможности расчленены для лучшей обозримости (см. § 5). Следует еще раз установить наличие в нескольких таблицах одинаковых условий, что часто дает возможность сократить число таблиц или уменьшить число условий в отдельных таблицах.

ТР37	R1	R2	R3	R4	R5	R6	R7
D1≤	16	–	21	21	80	80	E
RBR≤	–	8	15	50	13	50	
СПЕЦКОНСТ	x	x					x
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО			07	06	09	08	
END	x	x	x	x	x	x	x

ТР36	R1	R2	R3	R4	R5	R6	R7	R8
RBR	≤8	≥50	–	–	–	–	–	E
LBT≤	–	–	1	2	3	4	5	
СПЕЦКОНСТ	x	x						x
ИЗМЕРИТЕЛЬНОЕ УСТРОЙСТВО			01	02	03	04	05	
END	x	x	x	x	x	x	x	x

Рис. 30. Комплекс ТР для выбора измерительных индикаторов.

Общий комплекс ТР должен быть проверен на содержательную правильность вручную на примерах, которые были обработаны традиционными методами.

Пример 9. На рис. 30 приведен возможный комплекс ТР для выбора типа измерительных устройств. Представление выполнено в форме расширенных таблиц для наглядного сравнения с простыми таблицами ТР29—ТР32. Особую наглядность дает обозначение связей между таблицами с помощью стрелок. В общем процессе выбора соответствующего оснащения действие END (КОНЕЦ) не встречается при выборе измерительных устройств, так как должен быть предусмотрен переход к последующим ТР с помощью оператора GOTO.

Шаг 6. Обработка таблиц решений. При ручной обработке, например при составлении каталогов, может быть использовано полученное в шаге 5 окончательное описание задачи. При машинной обработке (возможности ее см. в гл. 4) ТР должны быть приспособлены (припасованы) к используемым методам и средствам обработки, что обычно выполняется программистами.

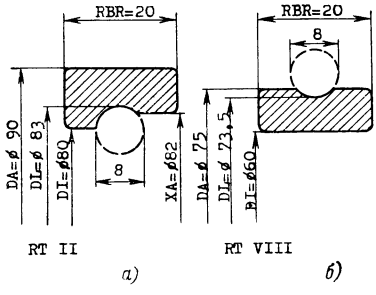


Рис. 31. Примеры возможных форм и размеров колец подшипников.

Пример 10 (ручная обработка). Пользователю нужен внутришлифовальный станок для обработки колец подшипников качения, размеры и формы которых приведены на рис. 31.

Для кольца по рис. 31,*а* в соответствии с ТР на рис. 30 получаем:

$$\text{TP33(RTII)} = \text{R1}.$$

Отсюда следует, что необходим станок МТА.

Так как

$$\text{TP34(RTII)} = \text{R2},$$

то

$$\text{LBT} = (\text{DL} - \text{XA})/2 = 0{,}5 \text{ мм};$$

$$\text{TP35(RBR} = 20,\ \text{LBT} = 0{,}5) = \text{R2},$$

откуда вытекает необходимость в измерительном устройстве 01.

Таким образом, нужное пользователю кольцо подшипника изготавливается на станке типа А; глубина желоба 0,5 мм, шлифовальный станок должен быть снабжен измерительным устройством 01.

Для кольца по рис. 31,*б* можно получить следующие результаты: кольцо необходимо обрабатывать на станке типа В при наличии измерительного устройства 08. Читатель может легко проверить этот результат.

ГЛАВА ТРЕТЬЯ

ИНТЕРПРЕТАЦИЯ И ПРОГРАММИРОВАНИЕ ТАБЛИЦ РЕШЕНИЙ

В этой главе мы рассмотрим некоторые алгоритмы, с помощью которых оказывается возможной обработка ТР на ЭВМ. Другие алгоритмы можно найти, в частности, в [7—11].

Итак, рассмотрим различные реализации алгоритмов.

11. АЛГОРИТМ КИРКА (МЕТОД МАСКИ)

С помощью этого алгоритма [12] обрабатываются ТР с простыми входами в указателе условий.

Пользователь должен представить задачу в форме простой таблицы, а при наличии расширенной таблицы преобразовать ее в простую способом, описанным в § 4.

Известно несколько модификаций этого алгоритма (см. [13]). Поясним этот алгоритм на примере.

ТР38	R1	R2	R3	R4	R5	R6	E
A = 0	Y	Y	Y	—	N	N	
B = 0	—	N	Y	N	N	Y	
C = 0	N	—	N	Y	N	—	
D = 0	Y	—	N	Y	—	—	
	M1	M2	M3	M4	M5	M6	M7

Рис. 32. Таблица решений к примеру 11.

Пример 11. Пусть дана ТР (рис. 32). Закодируем эту таблицу в виде двух двоичных матриц, содержащих только 0 и 1. Первая матрица T получается путем замены символов Y на 1 и помещения нулей в остальные места. Вторая матрица — матрица масок A содержит 0 вместо «—», а в остальных местах 1. Для заданной ТР получим:

$$T = \begin{Vmatrix} 1110000 \\ 0010010 \\ 0001000 \\ 1001000 \end{Vmatrix}; \quad A = \begin{Vmatrix} 1110110 \\ 0111110 \\ 1011100 \\ 1011000 \end{Vmatrix}.$$

Следует иметь в виду, что при таком кодировании должно быть включено и правило ИНАЧЕ. При этом следует учесть, что правило ИНАЧЕ всегда может быть представлено в виде правила, у которо-

го входы для каждого условия являются прочерками. Пусть S — заданная ситуация. Тогда закодируем S как матрицу T и образуем произведения Ak·S. Здесь Ak — k-й столбец A, и операция умножения между Ak и S выполняется по-компонентно по правилам [1]:

$$1 \cdot 1 = 1;\ 1 \cdot 0 = 0 \cdot 1 = 0 \cdot 0 = 0.$$

В результате имеем TP(S) = Ri тогда, когда i является первым индексом, для которого справедливо равенство

$$Ai \cdot S = Ti.$$

Пусть S = (N, N, Y, Y) = (0, 0, 1, 1), тогда для нашего примера получим:

$$A1 \cdot S = (0, 0, 1, 1) \neq (1, 0, 0, 1) = T1;$$

$$A2 \cdot S = (0, 0, 0, 0) \neq (1, 0, 0, 0) = T2;$$

$$A3 \cdot S = (0, 0, 1, 1) \neq (1, 1, 0, 0) = T3;$$

$$A4 \cdot S = (0, 0, 1, 1) = (0, 0, 1, 1) = T4.$$

Это означает, что TP38(S) = R4. Результат легко проверить непосредственно по TP38. Описанный в § 17 интерпретатор TP в основном основывается на алгоритме маски.

12. АЛГОРИТМ ПОЛЛАКА (POLLACK)

Алгоритм Поллака [14] ставит в соответствие каждой TP типа P эквивалентную блок-схему, точнее эквивалентное двоичное дерево решений. Так как такую блок-схему не составляет труда записать на надлежащем языке программирования, то мы будем называть ее просто программой. Ограничение таблиц типом P несущественно, так как в случае необходимости может быть применено описанное в § 4 эквивалентное преобразование. Так как это преобразование необходимо делать довольно часто, то алгоритм частично теряет свою эффективность. Алгоритм Поллака можно применить непосредственно к таблицам, не являющимся таблицами типа P, но и здесь следует считаться с значительными потерями эффективности. Поэтому мы опишем ниже (см. § 13) локальный алгоритм, который ставит в соответствие таблицам, не принадлежащим к типу P, необходимую эквивалентную блок-схему.

Итак, пусть задана TP типа P. Мы можем считать, что TP не содержит пустых (лишних) условий. Далее можно принять, что TP имеет более одного условия (случай, когда n = 1, — тривиален). Ядром алгоритма является следующая процедура выбора.

Каждому столбцу поставим в соответствие *коэффициент столбца*

$$\sigma_k = 2^{r_k}, \tag{2}$$

где r_k — число пробелов в k-м столбце.

Каждой строке поставим в соответствие *пробельное число*

$$\beta_j = \sum_{b_{ij}=-} \sigma_i. \tag{3}$$

[1] Производится логическое умножение (машинная операция в ЭВМ!) матриц Ak и S. — *Прим. ред.*

TP39	R1	R2	R3	R4	R5	E	β	ζ
B1	Y	Y	N	N	N		0	1
B2	Y	N	Y	N	N		0	3
B3	Y	—	Y	Y	N		4	
B4	Y	—	—	Y	N		6	
	M1	M2	M3	M4	M5	M6		
σ	1	4	2	1	1			

а)

TP39Y2	R1′	R3′	E	β	ζ
B1	Y	N		0	2
B3	Y	Y		0	4
B4	Y	—		2	
	M1	M3	M6		
σ	1	2			

б)

TP39N2	R2′	R4′	R5′	E	β	ζ
B1	Y	N	N		0	
B3	—	Y	N		4	
B4	—	Y	N		4	
	M2	M4	M5	M6		
σ	4	1	1			

в)

Рис. 33. Таблица решений к примеру 12 с введенными пробельными числами β, коэффициентами столбцов σ и наименьшими строчными коэффициентами ζ (*а*) и частные ТР (*б* и *в*), полученные из ТР39 по второму условию.

Суммируются все коэффициенты σ_i тех столбцов, которые в выбранной строке содержат «—». Если рассматриваемая строка не содержит «—», то $\beta_j = 0$. Пусть ТР имеет строку с наименьшим пробельным числом (i — номер этой строки). Проверим это условие, разделяя при Y на TPY и при N на TPN, и снова применим наш алгоритм к полученным частным таблицам. Если не существует минимального пробельного числа, то для строк вычисляется наименьший *строчный коэффициент*

$$\zeta_i = \left| \sum_{b_{1i}=Y} \sigma_i - \sum_{b_{1i}=N} \sigma_i \right|. \qquad (4)$$

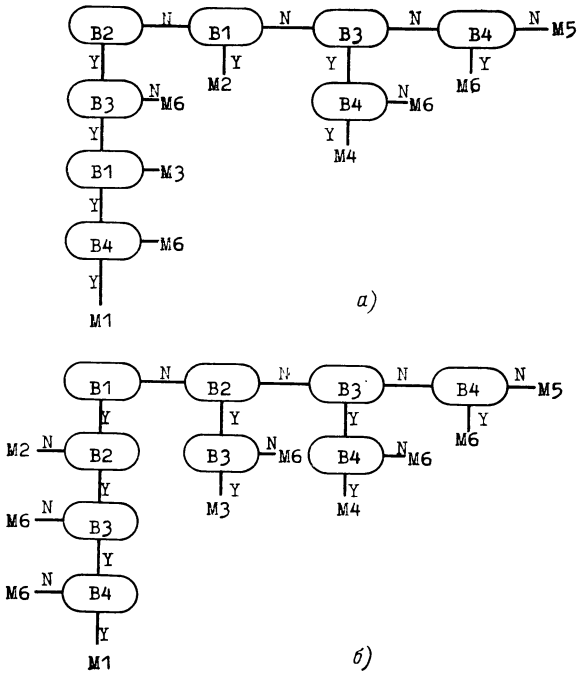


Рис. 34. Блок-схемы, эквивалентные ТР39.

Среди строк с наименьшим пробельным числом сначала проверяются те, которые имеют наибольшие строчные коэффициенты. Если существует несколько таких строк, предпочтение отдается тем строкам, которые в таблице помещены выше, и затем продолжается программирование соответствующих частных таблиц. Поясним алгоритм на примере.

Пример 12. Рассмотрим для этого таблицу TP39 на рис. 33,*а*. В нашем примере $\sigma_1=2^0=1$, $\sigma_2=2^2=4$, так как первое правило не содержит пробелов, а во втором их два. Из соотношения (2) получаем:

$$\beta_3=\sigma_2=4;\ \beta_4=\sigma_2+\sigma_3=6 \text{ и т. д.}$$

Строчные коэффициенты ζ_1 и ζ_2 (остальные не представляют интереса) получим из соотношения (4):

$$\zeta_1=\sigma_1+\sigma_2-\sigma_3-\sigma_4-\sigma_5=1;$$
$$\zeta_2=\sigma_1+\sigma_3-\sigma_2-\sigma_4-\sigma_5=3.$$

В соответствии с требованиями алгоритма следует начинать со второго условия. Ветвь Y приводит к TP39Y2 (рис. 33,*б*). В этой частной таблице совпадают два первых пробельных числа. Строчный коэффициент приводит к продолжению по третьему условию. Последние две проверки осуществляются легко. Аналогичным образом обрабатывается TP39N2. В итоге получится блок-схема, приведенная на рис. 34,*а*. Если бы мы начали с первого условия, мы получили бы блок-схему, изображенную на рис. 34,*б*.

Конечно, полученная программа далека от совершенства. В § 14 будут сформулированы точные критерии, позволяющие оценить полученные из TP программы.

13. ЛОКАЛЬНЫЙ АЛГОРИТМ

В этом параграфе описывается алгоритм, который любой простой TP [1] ставит в соответствие эквивалентную блок-схему и составляет «хорошие» блок-схемы для таблиц, не относящихся к типу P. Кроме того, этот алгоритм допускает некоторые уточнения. Они не рассматриваются, чтобы не усложнять описание. Некоторые подобные уточнения внимательный читатель может найти сам [см. также задачу для упражнений 11,*б*)].

Мы опишем алгоритм с помощью TP. Для этого введем следующие операции с двумя правилами:

$(Ri \cap Rj)_k = \underline{0}$ тогда и только тогда, когда Ri и Rj отличаются в первых k компонентах по крайней мере одним значимым[2] указателем. (5)

В противном случае $(Ri \cap Rj)_k \neq \underline{0}$.

В качестве примера рассмотрим TP38 на рис. 32, для которой $(R2 \cap R3)_3 = \underline{0}$; $(R1 \cap R2)_4 \neq \underline{0}$; $(R1 \cap R5)_2 = \underline{0}$.

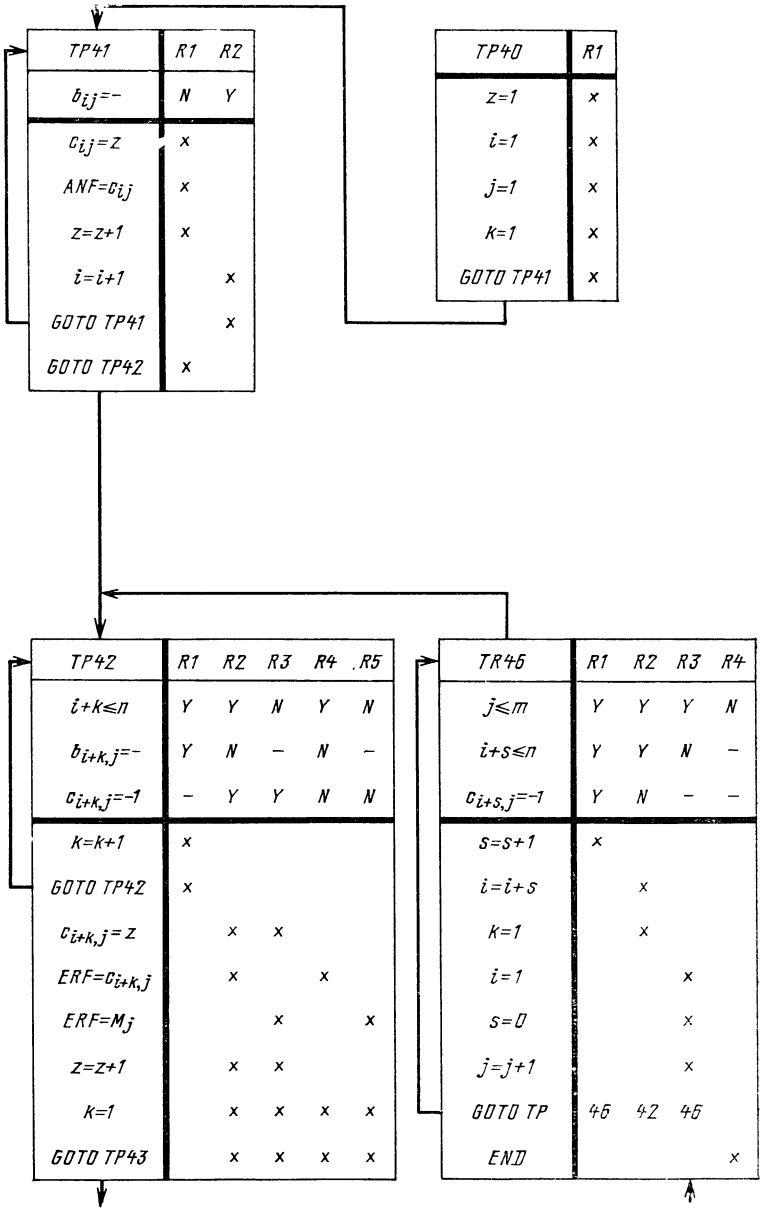
Кроме того, будет использоваться следующее отношение порядка:

$(Ri \leqslant Rj)_k$ тогда и только тогда, когда (b_{ri}, b_{rj}) принадлежит множеству {(—, —), (Y, Y), (N, N), (Y, —), (N, —)} для $1 \leqslant r \leqslant k$. (6)

---

[1] Предполагается, что таблица не безусловная и соответственно не эквивалентная безусловной. — *Прим. автора.*

[2] Отличие N от «—» и Y от «—» в расчет не принимается. — *Прим. пер.*


TP41
R1
RZ
b_{ij}=-
N
Y
c_{ij}=Z
x
ANF=c_{ij}
x
Z=Z+1
x
i=i+1
x
GOTO TP41
x
GOTO TP42
x
TP40
R1
Z=1
x
i=1
x
j=1
x
K=1
x
GOTO TP41
x
TP42
R1
R2
R3
R4
R5
i+k≤n
Y
Y
N
Y
N
$b_{i+k,j}$=-
Y
N
–
N
–
$c_{i+k,j}$=-1
–
Y
Y
N
N
K=K+1
x
GOTO TP42
x
$c_{i+k,j}$=Z
x
x
ERF=$c_{i+k,j}$
x
x
ERF=M_j
x
x
Z=Z+1
x
x
K=1
x
x
x
x
GOTO TP43
x
x
x
x
TR46
R1
R2
R3
R4
j≤m
Y
Y
Y
N
i+s≤n
Y
Y
N
–
$c_{i+s,j}$=-1
Y
N
–
–
S=S+1
x
i=i+s
x
K=1
x
i=1
x
S=0
x
j=j+1
x
GOTO TP
46
42
46
END
x

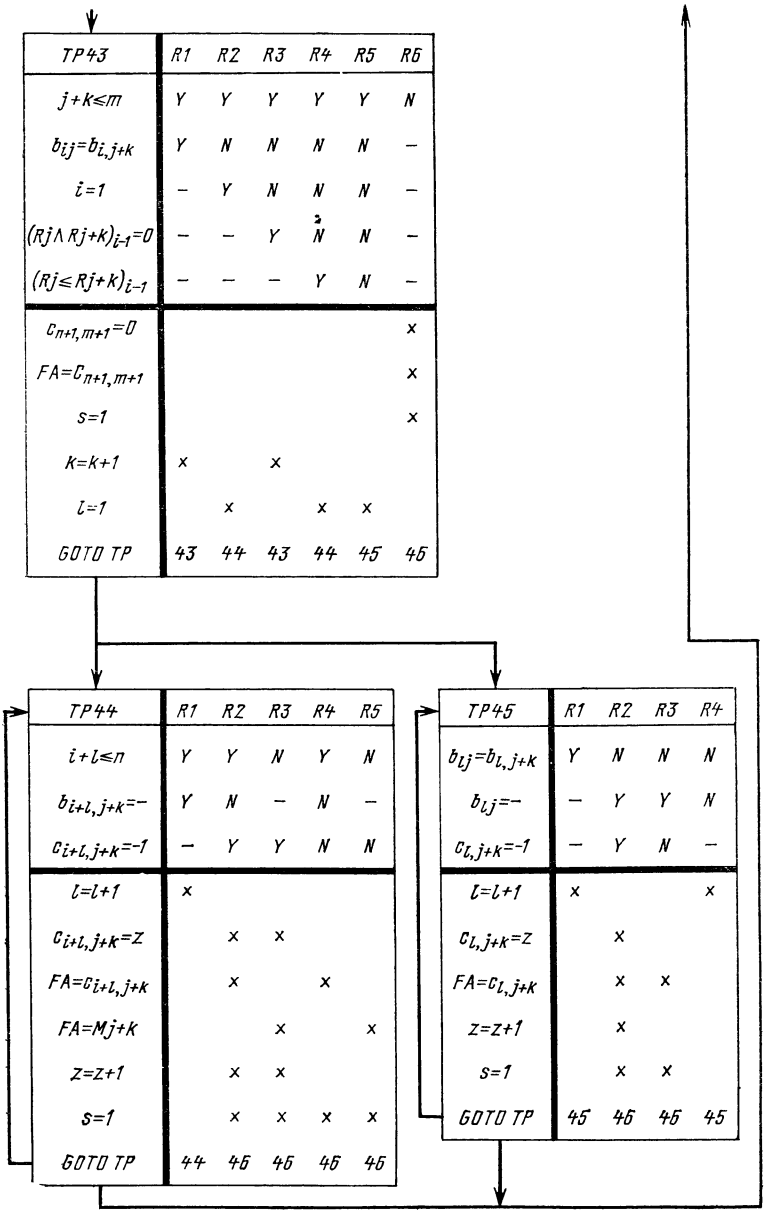


Рис. 35. Описание локального алгоритма с помощью ТР.

Для ТР38 это означает, что

$$(R2 \leqslant R4)_2;\ (R3 \leqslant R4)_1;\ (R5 \leqslant R6)_1.$$

Наряду с матрицей условий $B = \|b_{ij}\|$ введем матрицу $C = \|c_{rs}\|$ $(1 \leqslant r \leqslant n+1;\ 1 \leqslant s \leqslant m+1)$. Она служит для получения необходимых меток. До начала работы все элементы матрицы C принимаются равными —1. Алгоритм может быть описан с помощью ТР, представленных на рис. 35 (ТР40—ТР46).

Три действия на этом рисунке требуют пояснения:

1. $ANF = c_{ij}$ — Указание первой проверки блок-схемы (ANF — начало), при котором проверяется условие Bi. Значение c_{ij} вводится в проверку, как показано на рис. 37,*а*.

2. ERF= ... Найти в блок-схеме проверку Bi с меткой c_{ij}. Если b_{ij}=Y(N), то связать ветвь Y(N) этой проверки:

1) с проверкой B_{i+k}, если $ERF = c_{i+k,j}$;

2) с действием Mj, если $ERF = M_j$. Значение $c_{n+1,j}$ используется в последнем случае в качестве метки для действия; ERF называется *результирующим адресом.*

3. FA= ... Найти проверку Bi с меткой c_{ij} в блок-схеме. Если b_{ij}=Y(N), то следует связать ветвь Y(N) этой проверки:

1) с проверкой B1, если $FA = c_{1,j+k}$;

2) с действием Mj+k, если FA=Mj+k. И в последнем случае с помощью значений $c_{n+1,j+k}$ выделяется некоторое действие. FA называется *адресом ошибки* [1].

Таблицы на рис. 35 выполняют следующие функции:

ТР40 является безусловной ТР. Она засылает известные начальные значения;

ТР41 ищет первый, отличающийся от пробела вход в первом правиле, который формирует начало блок-схемы;

ТР47	R1	R2	R3
B10<	Y	N	—
B2 ⩽ 1	—	N	Y
B3 ≠ 4	N	—	—
B4 = 1	N	—	N
BETA =	6	10	21

Рис. 36. Таблица решений к примеру 13.

ТР42 ищет по b_{ij} результирующий адрес. Он задается следующим отличным от пробела входом в правиле Rj, который лежит ниже b_{ij}. Это или снова вход в указатель условий, или действие Mj;

ТР43 ищет правило, в котором должен быть адрес ошибки, и устанавливает, лежит ли он выше i-го условия (ТР45) или ниже его (ТР44);

ТР44 определяет «низкие» адреса ошибок;

ТР45 сопределяет «высокие» адреса ошибок;

ТР46 устанавливает новые начальные значения для таблицы ТР42.

[1] Каждому адресу ошибки FA однозначно ставится в соответствие вектор (z, c_{ij}, b_{ij}, соответственно Mj или 0). Зная единственную компоненту этого вектора, можно вычислить две другие. Поэтому ясно, почему мы обозначаем адрес ошибки в одном случае числом 4, а в другом b_{31} или M2.

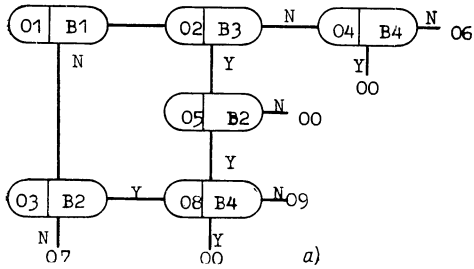
То же справедливо и для результирующего адреса. — *Прим. автора.*

Таблица 3

Рабочий протокол превращения таблицы TP47 **в блок-схему с помощью локального алгоритма**

Строка	Таблица	c_{ij}	z	i	j	k	s	l	ANF	ERF	FA	TP
1	TP40/R1	—	1	1	1	1	—	—	—	—	—	41
2	TP41/R1	$c_{11}=1$	2	—	—	—	—	—	c_{11}	—	—	42
3	TP42/R1	—	—	—	—	2	—	—	—	—	—	42
4	TP42/R2	$c_{31}=2$	3	—	—	1	—	—	—	c_{31}	—	43
5	TP43/R2	—	—	—	—	—	—	1	—	—	—	44
6	TP44/R2	$c_{22}=3$	4	—	—	—	1	—	—	—	c_{22}	46
7	TP46/R1	—	—	—	—	—	2	—	—	—	—	46
8	TP46/R2	—	—	3	—	1	—	—	—	—	—	42
9	TP42/R2	$c_{41}=4$	5	—	—	1	—	—	—	c_{41}	—	43
10	TP43/R3	—	—	—	—	2	—	—	—	—	—	43
11	TP43/R5	—	—	—	—	—	—	1	—	—	—	45
12	TP45/R4	—	—	—	—	—	—	2	—	—	—	45
13	TP45/R2	$c_{23}=5$	6	—	—	—	1	—	—	—	c_{23}	46
14	TP46/R2	—	—	4	—	1	—	—	—	—	—	42
15	TP42/R3	$c_{51}=6$	7	—	—	1	—	—	—	M1	—	43
16	TP43/R3	—	—	—	—	2	—	—	—	—	—	43
17	TP43/R1	—	—	—	—	3	—	—	—	—	—	43
18	TP43/R6	c_{54}	—	—	—	—	1	—	—	—	c_{54}	46
19	TP46/R3	—	—	1	2	—	0	—	—	—	—	46
20	TP46/R1	—	—	—	—	—	1	—	—	—	—	46
21	TP46/R2	—	—	2	—	1	—	—	—	—	—	42
22	TP42/R1	—	—	—	—	2	—	—	—	—	—	42
23	TP42/R1	—	—	—	—	3	—	—	—	—	—	42
24	TP42/R3	$c_{52}=7$	8	—	—	1	—	—	—	M2	—	43
25	TP43/R4	—	—	—	—	—	—	1	—	—	—	44
26	TP44/R1	—	—	—	—	—	—	2	—	—	—	44
27	TP44/R2	$c_{43}=8$	9	—	—	—	1	—	—	—	c_{43}	46
28	TP46/R1	—	—	—	—	—	2	—	—	—	—	46
29	TP46/R1	—	—	—	—	—	3	—	—	—	—	46
30	TP46/R3	—	—	1	3	—	0	—	—	—	—	46
31	TP46/R1	—	—	—	—	—	1	—	—	—	—	46
32	TP46/R2	—	—	2	—	1	—	—	—	—	—	42
33	TP42/R1	—	—	—	—	2	—	—	—	—	—	42
34	TP42/R4	—	—	—	—	1	—	—	—	c_{43}	—	43
35	TP43/R6	$c_{54}=0$	—	—	—	—	1	—	—	—	c_{54}	46
36	TP46/R1	—	—	—	—	—	2	—	—	—	—	46
37	TP46/R2	—	—	4	—	1	—	—	—	—	—	42
38	TP42/R3	$c_{53}=9$	10	—	—	1	—	—	—	M3	—	43
39	TP43/R6	$c_{54}=0$	—	—	—	—	1	—	—	—	c_{54}	46
40	TP46/R3	—	—	1	4	—	0	—	—	—	—	46
41	END											

Пример 13. Рассмотрим таблицу TP47 (рис. 36). Входными величинами для нашего алгоритма будут: m=3; n=4; c_{rs}=—1 ($1 \leqslant r \leqslant 5$; $1 \leqslant s \leqslant 4$). Действие алгоритма представлено табл. 3. Первая строка этой таблицы дает начальные значения для TP40. В последнем столбце указано, какую таблицу следует обрабатывать с помощью этих значений. Этой таблицей является TP41, в которой по правилу R1 ищется начало. Оно сразу же наносится на схему (рис. 37,*а*). В строке 4 находится результирующий адрес, который



а)

01	–1	–1	–1
–1	03	05	–1
02	–1	–1	–1
04	–1	08	–1
06	07	09	00

б)

Рис. 37. Блок-схема, эквивалентная TP47 (*а*), и матрица меток для TP47 после обработки локальным алгоритмом (*б*).

связывается с началом ветвью Y. Соответственно найденный в строке 6 адрес ошибки связывается ветвью N с началом. Продолжая таким образом, получаем блок-схему, изображенную на рис. 37,*а*, которую можно записать на языке программирования (см. § 16, п. «в»).

На первый взгляд локальный алгоритм может показаться весьма трудоемким и утомительным: довольно сложная процедура, обширный протокол (табл. 3), блок-схема и, наконец, запись (кодирование) ее на языке программирования. Однако эти опасения неоправданны, так как основные положения алгоритма весьма просты. Уже после немногих учебных примеров читатель будет в состоянии получать блок-схемы без таблиц, приведенных на рис. 35, и без протокола. Возможно и дальнейшее упрощение. С помощью локального алгоритма можно непосредственно кодировать таблицы.

14. ОЦЕНКА АЛГОРИТМОВ

Алгоритмы для перевода TP в программы можно оценивать по различным критериям:

1) стоимость процесса перевода;
2) потребности в памяти у полученной программы;
3) времени работы полученной программы.

Само собой разумеется, что эти величины не могут быть оптимизированы независимо друг от друга. Если исходить из ограниченной стоимости перевода, то мы придем к выбору простого алгоритма. Программы с особенно высокими требованиями по отношению к двум остальным критериям вообще не могут быть получены. Для программ, которые часто используются, именно эти критерии и в особенности критерий 3 оказываются наиболее желательными. Естественно, что при этом значения критерия 1 должны оставаться в известных пределах.

В качестве оценки объема памяти программ, полученных из ТР, будем использовать количество проверок в программе, хотя эта величина лишь весьма грубо отражает фактическое состояние, так как она не учитывает ни количества действий в таблице, ни различия длин отдельных проверок.

По вышеустановленному критерию оба алгоритма, описанные в § 12 и 13, не оптимальны, так как существуют таблицы, для которых можно построить программы с меньшим числом проверок, чем получающиеся при пользовании этими алгоритмами.

В качестве оценки для времени работы программы выберем *среднюю длину ветви* программы. Она определяется следующим образом. Пусть Sk — некоторая ситуация, w(Sk) — число проверок, подлежащих выполнению для нахождения относящегося к этой ситуации действия. Тогда величина

$$\bar{w} = 2^{-n} \sum_{j=1}^{2n} w\,(Si) \qquad (7)$$

и будет средней длиной ветви программы. Например, для программы, изображенной на рис. 34,*а*, получим:

$$\bar{w} = (4 \cdot 2 + 2 \cdot 3 + 2 \cdot 4 + 4 \cdot 2 + 2 \cdot 4 + 2 \cdot 4)/16 = 46/16 = 2{,}875.$$

Ветвь B2—B3 проходится при ситуациях (Y, N, Y, Y), (Y, N, Y, N), (Y, N, N, Y) и (Y, N, N, N), ветвь B2—B3—B1 при обеих ситуациях (Y, Y, N, Y) и (Y, Y, N, N). Аналогично находят остальные члены соотношения (7). Интерпретация величины $\bar{w}$ очевидна. Для того, чтобы найти действие, соответствующее одной ситуации, в среднем требуется $\bar{w}$ проверок. В этом смысле $\bar{w}$ является мерой для времени реализации программы. Она отражает фактические отношения правильно тогда, когда все ситуации равновероятны. В остальных случаях $\bar{w}$ вычисляют по формуле

$$\bar{w} = \sum_{i=1}^{2n} \alpha_j w\,(Si), \qquad (8)$$

где α_i означает *относительную частоту ситуации* Si.

Во многих практических случаях оказывается возможным получить приемлемые оценки для величин α_i. При этом оказывается возможным существенно улучшить качество локального алгоритма, если правила, относящиеся к наиболее часто встречающимся ситуациям, поместить по возможности в начале таблицы. На алгоритм Поллака эта перестановка не оказывает влияния.

Итак, оба алгоритма дают хорошие оценки в отношении требуемых объема памяти и времени работы, но не во всех случаях с их помощью получаются оптимальные программы. Улучшение (за исключением некоторых уточнений локального алгоритма) оказывается возможными лишь при сильно возрастающих затратах времени их работы (ср. [6, 7]).

15. ЗАДАЧИ ДЛЯ УПРАЖНЕНИЙ

10. Используя локальный алгоритм, составьте блок-схему программы по ТР48 (рис. 38). Эквивалентную заданной таблицу ТР49 (рис. 39) обработайте по алгоритму Поллака и сравните результаты по:

TP48	R1 R2 E
C1 = 0	N —
C2 = 0	— N
C3 = 0	N —
BETA =	1 2
CONTINUE	×

Рис. 38. Пример ТР к задаче 10.

TP49	R1 R2 R3 E
C1 = 0	N Y N
C2 = 0	— N N
C3 = 0	N — Y
BETA =	1 2 2
CONTINUE	×

Рис. 39. Исходные данные при обработке ТР с помощью предтранслятора.

а) необходимому объему памяти;
б) по времени работы программы.

11. Дана ТР50 типа Р (рис. 40).

а) Составьте для нее программу по алгоритму Поллака и, пользуясь локальным алгоритмом, сравните результаты.

б) Запрограммируйте таблицу, которая получится из таблицы ТР50 путем изменения порядка следования правил на обратный, с помощью локального алгоритма, и подсчитайте среднюю длину ветви программы.

TP50	R1	R2	R3	R4	E
B1	—	N	Y	Y	
B2	N	—	Y	N	
B3	Y	—	—	N	
B4	Y	N	—	—	
	M1	M2	M3	M4	M5

Рис. 40. Исходные данные при объединении ТР.

TP51	R1	R2	R3	R4	R5	E
B1	N	—	Y	N	Y	
B2	N	Y	—	Y	N	
B3	—	N	N	Y	Y	
B4	—	N	Y	Y	N	
	M1	M2	M2	M2	M2	M3

Рис. 41. Таблица решений к задаче 12.

12. Покажите на примере ТР51 (рис. 41), что алгоритм Поллака и для таблиц типа Р не приводит к программе, оптимальной по объему памяти.

Указание. Запрограммируйте таблицу по алгоритму Поллака. Во второй программе начните с проверки B4 и затем к таблицам TP51Y4 и TP51N4 примените алгоритм Поллака.

ГЛАВА ЧЕТВЕРТАЯ

ОБРАБОТКА ТАБЛИЦ РЕШЕНИЙ

В этой главе рассмотрены некоторые методы обработки ТР. С помощью этих методов читатель получит возможность в соответствии со своими данными получить из ТР программу для машинной обработки. Некоторые методы описаны подробно, а для других даны только наметки. Описание методов обработки дано в порядке возрастания их сложности. Сначала рассмотрено ручное программирование, затем различные этапы автоматической обработки. Указаны преимущества и недостатки описываемых методов.

Для того чтобы впоследствии иметь возможность оценить суммарные затраты ручного труда, перечислим основные этапы вычислительной обработки задач:

1) составление подробной блок-схемы программы по данным анализа задачи и по грубой информационной блок-схеме;

2) представление подробной блок-схемы на проблемно- или машинно-ориентированном языке (получение исходной программы);

3) трансляция исходной программы в машинную программу, которая может быть реализована на ЭВМ (при этом осуществляется проверка синтаксической правильности программы);

4) проверка машинной программы на логическую правильность с помощью контрольных значений (исправления, как правило, вводятся в исходную программу, и дальнейшее протекание продолжается с п. 3);

5) документирование программы.

16. РУЧНОЕ ПРОГРАММИРОВАНИЕ

Если в распоряжении специалиста нет подходящих вспомогательных вычислительных средств для обработки ТР, то последние могут быть преобразованы в программу вручную. Для этого служит несколько методов, из которых мы рассмотрим три.

а) Условия разделения и их описание

Сначала рассмотрим метод двоичного кодирования для простых ТР, в которых не встречаются пробелы (ср. также [16]).

Пусть дана такая таблица. Каждому правилу этой таблицы ставится в соответствие некоторое число — *характеристика правила*:

$$Z(Ri) = 1 + a_{1i} \cdot 2^0 + a_{2i} \cdot 2^1 + \ldots + a_{ni} \cdot 2^{n-1}, \qquad (9)$$

где

$$1 \leqslant i \leqslant m;$$

$$a_{ji} = \begin{cases} 1 \text{ при } b_{ji} = Y; \\ 0 \text{ при } b_{ji} = N. \end{cases} \qquad (10)$$

В рассматриваемой программе сначала вычисляют характеристики правил Z(Ri). Затем в соответствии с значением Z(Ri) осуществляется переход к необходимому действию. Этот метод может быть запрограммирован с помощью оператора условного разветвления или с помощью оператора передачи управления. Например, здесь

может быть использован оператор **switch** (переключатель) при программировании на АЛГОЛе [17] или оператор GOTO при программировании на ФОРТРАНЕ [18].

Пример 14. Пусть необходимо запрограммировать методом двоичного кодирования на АЛГОЛе ТР52 (рис. 42). Из ТР52 следуют следующие зависимости:

Z(R1)=7. Выполнение действия 1 (A=1) с маркером m1;
Z(R2)=3. Выполнение действия 2 (B=B+1) с маркером m2;
Z(R3)=5. Выполнение действия 3 (B=0) с маркером m3;
Z(R4)=6. Выполнение действия 4 (C=0) с маркером m4.

ТР52	R1	R2	R3	R4	E
B1 > 0	N	N	N	Y	
B2 > 0	Y	Y	N	N	
B3 > 0	Y	N	Y	Y	
A=1	×				
B=B+1		×			
B=0			×		
C=0				×	
C=1					×

Рис. 42. Таблица решений к примеру 14.

Во всех остальных случаях выполняется действие 5 (C=1) с маркером m5.

Для вычисления коэффициентов a_{ji} из соотношения (9) используем следующий условный оператор:

if проверка **then** оператор 1
else оператор 2;

Он работает следующим образом. Если результат проверки (условие нашей таблицы) истинный (Y), выполняется оператор 1, в противном случае (N) выполняется оператор 2. С помощью оператора 1 засылается $a_{ji}=1$, а с помощью оператора 2 $a_{ji}=0$.

Кроме того, нам понадобится описание переключателя. Он имеет следующую структуру:

switch имя ветви := список меток ветвей;

К желаемой метке переход осуществляется с помощью оператора:

goto имя ветви [арифметическое выражение];

Этот оператор действует следующим образом:

Программа осуществляет переход по k-й метке, если значение арифметического выражения равно k, т. е. к первой, если k=1, ко второй, если k=2, и т. д.

В нашем случае переменной разветвления является z — арифметическое выражение от переменной r, и метками ветвления являются метки действий:

switch z:=m5, m5, m2, m5, m3, m4, m1, m5;

Таким образом, можно записать следующий фрагмент программы на АЛГОЛе:

if b1 > 0 **then** a1 := 1 **else** a1 := 0;
if b2 > 0 **then** a2 := 1 **else** a2 := 0;
if b3 > 0 **then** a3 := 1 **else** a3 := 0;
r := 1 + a1 + a2 × 2 + a3 × 4;

goto z [r];

```
m1 : a : = 1; goto m6;
m2 : b : = b + 1; goto m6;
m3 : b : = 0; goto m6;
m4 : c : = 0; goto m6;
m5 : c : = 1;
m6 : ...
```

б) Поразрядная связь (ПЛ/1)

Другой метод, используемый в языке программирования ПЛ/1, основывается на поразрядной связи двоичных переменных. Высказываниям — результатам проверки (условиям ТР) в этом случае ставят в соответствия значения одноразрядной двоичной переменной:

двоичная переменная = проверка;

Если результат проверки истинный (Y), двоичная переменная получает значение 1, в противном случае (N) — значение 0.

Составляют n-разрядное двоичное число (n — число проверок). Каждому разряду соответствует описанная выше одноразрядная двоичная переменная. В нашем случае длина (число разрядов) NV этой n-разрядной переменной равна 3. Каждому биту присвоим имена BV3, BV2 и BV1.

Двоичная переменная NV имеет значение Z(Ri)—1, где Z(Ri) — описанная выше характеристика правила. С помощью оператора ветвления, который в принципе работает как переключатель АЛГОЛа, получим следующий фрагмент программы для ТР52:

```
BV1=B1>0;
BV2=B2>0;
BV3=B3>0;
IF NV=6 THEN GOTO M1;
IF NV=2 THEN GOTO M2;
IF NV=4 THEN GOTO M3;
IF NV=5 THEN GOTO M4 ELSE GOTO M5;
. . . .
```

Здесь M1—M5 — метки действий, которые, как и в предыдущем случае, должны быть заранее выписаны.

Недостатком обоих описанных методов является возможность работы только с небольшими таблицами, не имеющими пробелов. Однако они позволяют формально преобразовывать ТР в программы, не используя подробных блок-схем программ (для программирования на ПЛ/1 ср. [19]).

в) Оператор условной передачи управления

Для ручного программирования ТР можно использовать алгоритмы, описанные в § 11 и 12. Составим программу для ТР47 (см. рис. 36) с помощью локального алгоритма. Его достоинством является возможность обработки таблиц любых размеров, содержащих пробелы. В качестве языка программирования выберем ФОРТРАН, однако может быть использован и любой другой проблемно- или

машинно-ориентированный язык. Для программирования используем оператор ФОРТРАНА **IF**:

IF (выражение) метка 1, метка 2, метка 3.

Выражение (условие нашей ТР) сравнивается с нулем. Если оно меньше нуля, осуществляется переход по метке 1, если выражение равно нулю — по метке 2, если оно больше нуля — по метке 3. Здесь следует заметить, что возможная структура условий ТР зависит от языка программирования. В основном ФОРТРАНе возможны лишь следующие условные структуры, соответствующие вышеупомянутым операторам:

арифметическое выражение = нуль.

В зависимости от результата осуществляется ветвление.

Для применения к таблицам решений возможная структура условий должна быть представлена в форме:

операнд 1 оператор операнд 2

При этом для операторов имеют силу одни и те же ограничения. В качестве операторов могут использоваться $=$, $\neq$, $\leqq$, $\geqq$, $<$ и $>$, которые могут быть реализованы для разделения по меткам. Это, однако, не позволяет непосредственно кодировать условия вида ЦИСТ НЕФТЬ (см. ТР4 на рис. 4).

Для ТР47 составим программу на ФОРТРАНе. Способ формирования матрицы меток С уже описывался в § 13. Эта матрица приведена на рис. 37,*б*. Без обращения к блок-схеме (см. рис. 37,*а*) может быть записан следующий фрагмент программы на ФОРТРАНе:

```
4701 IF (B1)        4703, 4703, 4702
4702 IF (B3 — 4)    4705, 4704, 4705
4704 IF (B4 — 1)    4706, 4700, 4706
4703 IF (B2 — 1)    4708, 4708, 4707
4705 IF (B2 — 1)    4708, 4708, 4700
4708 IF (B — 1)     4709, 4700, 4709
4706 BETA = 6
     GOTO4700
4707 BETA = 10
     GOTO 4700
4709 BETA = 21
4700 CONTINUE
     . . .
```

Как указано, существует целый ряд возможностей по переводу ТР в программы для ЭВМ. При этом достоинством является то, что в большинстве случаев оказывается возможным избежать составления подробной блок-схемы программы, которая нужна только при формировании связей между отдельными таблицами (см. рис. 20). Процесс программирования может быть описан формально в виде алгоритмов и может быть реализован с незначительными за-

тратами. При применении этих алгоритмов получаются хорошие программы. На этапах 3—5 программирования может быть достигнута небольшая экономия, например уменьшено время проверок, так как логические ошибки ТР были выявлены уже при анализе. Опыт программирования с помощью R300-MOPS изложен в [20].

17. ВСПОМОГАТЕЛЬНЫЕ СРЕДСТВА ПРИ РАБОТЕ С ТАБЛИЦАМИ РЕШЕНИЙ [1]

а) Макропрограммы

Макропрограммы являются подпрограммами, которые вводятся в соответствующие места программ и вызываются с помощью макрооператоров.

Макропрограммы составляются на машинно-ориентированном языке и только на нем могут быть вызваны. С помощью программы-ассемблера (машинно-ориентированной) можно обрабатывать любые таблицы. При этом в ЭВМ вводятся макрооператоры для макропрограмм ТР, в которых условия и матрица условий задаются в виде констант.

Для обработки с помощью макропрограмм пригодны только простые ТР. Недостатками их являются ограничение языка АССЕМБЛЕР (объединение с проблемно-ориентированными языками возможно лишь частично и при больших временных затратах) и довольно большой объем памяти, так как макропрограмма должна заново генерироваться для каждой ТР. Так как язык АССЕМБЛЕР ориентирован на определенные виды ЭВМ, вводимые макропрограммы могут быть использованы только для этого класса ЭВМ.

К числу макропрограмм для ТР можно отнести макропрограммы SETIT и TABLE, предназначенные для ЭВМ IBM/360 [21].

Макропрограмма TABLE содержит собственно алгоритм обработки для таблиц и осуществляет связь с множеством действий макропрограммы SETIT, кодирует вектор ситуаций, когда переключатели помещаются в одном поле. Для этой макропрограммы допустимы пробелы в матрице условий.

Следует иметь в виду, что главная часть макропрограммы TABLE генерируется в программе только 1 раз, тогда как ее меньшая часть и макропрограмма SETIT должны заново вводиться для каждой ТР. Таким образом экономится примерно половина объема памяти, которая была бы необходима для этих программ.

Подобные методы обработки удобны тогда, когда при решении задачи только отдельные ее части могут быть представлены в виде ТР. Для табличной части вместо подробной блок-схемы программы записываются лишь имя макропрограммы и ее параметры. Остальные части программы реализуются традиционными способами.

б) Интерпретатор

Интерпретатор является программой, написанной, как правило, на машинно-ориентированном языке, которая может обрабатывать ТР с простыми входами, содержащие пробелы. В противоположность другим методам обработки в этом случае ТР не переводится в про-

[1] Этот материал носит информационный характер и при первом знакомстве с книгой может быть опущен. — *Прим. ред.*

грамму, а сразу интерпретируется. Для этого необходим алгоритм, который при наличии таблицы в виде матрицы в оперативном запоминающем устройстве отыскивает в соответствии с заданной ситуацией верное правило и обращается к нужному действию. Процедура основывается на алгоритме масок, описанном в § 11. Помимо программы-интерпретатора, которая используется в программе только 1 раз, во время обработки в памяти ЭВМ должна быть записана сама таблица.

Места в памяти для нее требуется немного, но время на ее обработку в данном случае больше, чем при использовании макропрограмм. Ввод изменений здесь очень прост, так как изменяться должна лишь сама таблица. Техника интерпретации удобна для тех таблиц, которые обрабатываются однократно и часто заменяются. Интерпретатор весьма эффективен при использовании мини-ЭВМ.

Примером интерпретатора является транслятор автокода ТР (Autocoder Decision Table Translator) фирмы IBM, написанный для ЭВМ серии IBM/1400. Использование интерпретатора делает ненужными программу для ТР и подробную блок-схему программы. Так как ЭВМ сразу интерпретирует таблицу, это удешевляет ее обработку (не затрачивается время на преобразования).

18. ПРЕДТРАНСЛЯТОР

Предтранслятор дает широкую возможность для обработки ТР. При этом таблицы переводятся в программу, записанную на проблемно-ориентированном языке. Полученная программа обрабатывается обычным образом.

Грубая блок-схема обработки с использованием предтранслятора изображена на рис. 43.

Таблицы решений записываются на специальном языке. Он в основном соответствует применяемому языку программирования, расширенному за счет специфических для ТР ключевых слов и способов записи таблиц. Ключевые слова обозначают в простейшей форме операции или операторы, допустимые в используемом для ЭВМ языке. Таким языком ТР для ФОРТРАНа является FORTAB. Для языка ТР существуют специальные формуляры, на которые заносится программа ТР. Отперфорированная программа обрабатывается предтранслятором в следующей последовательности:

1) проверка заданных таблиц на наличие ошибок, которые могут возникнуть при применении языка ТР (например, многократное использование одних и тех же условий или правил в одной таблице);

2) подготовка таблицы для использования алгоритма переработки (например, преобразование ТР с расширенными входами в простые, составление внутренних матриц условий и матриц действий, изменение порядка строк, условий и соответственно столбцов);

3) переработка частей таблиц, содержащих условия, с помощью алгоритма в фрагмент программы, записанный на используемом языке (два алгоритма описаны в § 12 и 13);

4) переработка частей таблиц, содержащих действия, в фрагмент программы, который доступен транслятору ЭВМ.

Связь отдельных таблиц осуществляется с помощью действий.

При предтрансляции составляется протокол преобразования, который содержит исходные таблицы и составленную первичную программу и может быть по выбору выдан на перфокартах, магнитной

ленте или магнитных дисках. Последующая обработка осуществляется обычным способом, используемым при обработке подпрограммы, написанной на проблемно-ориентированном языке. Эта программа с помощью обычного транслятора переводится в программу для ЭВМ.

Использование предтранслятора дает пользователю ряд существенных выгод:

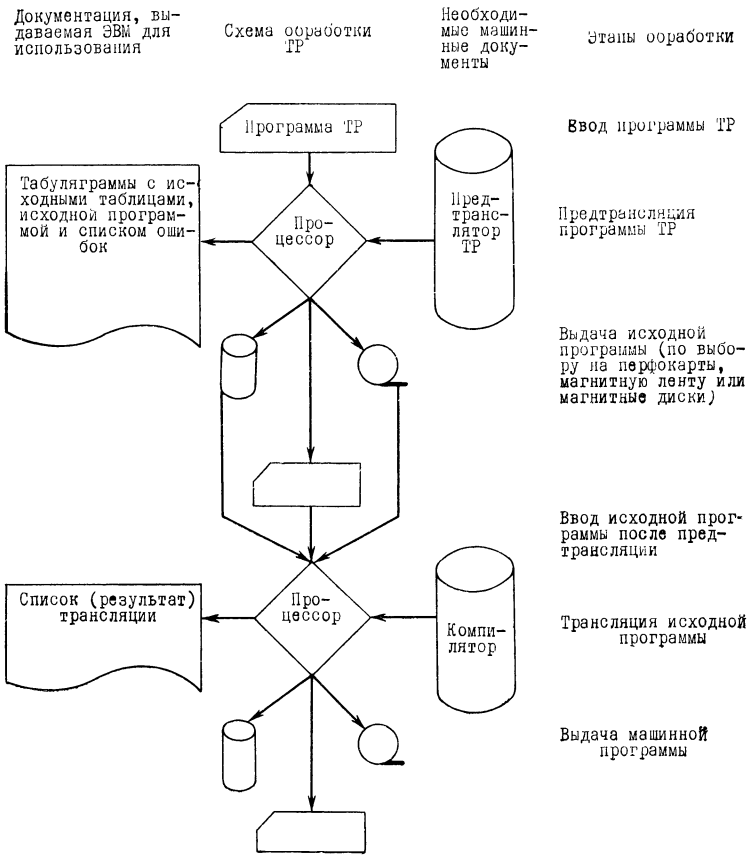


Рис. 43. Блок-схема обработки ТР с использованием предтранслятора.

таблицы решений записываются на специальном проблемно-ориентированном языке высокого уровня. Этот язык при знании основного языка программирования и даже без этого очень легко изучается;

язык ТР в силу своей структуры приспособлен для документирования;

возможна обработка ТР с простыми и с расширенными входами и с произвольно большим числом пробелов;

исходная программа в некоторых пределах оптимизируется;

при обработке машинной программы пользователь может наблюдать за ходом прохождения отдельных таблиц с помощью проверок или текущих указателей;

устранение большей части логических ошибок может осуществлять непрограммист (специалист, поставляющий таблицы решений) с помощью программ ТР. Синтаксические ошибки могут быть устранены как в программе ТР, так и в исходной программе.

Если при контроле машинной программы обнаруживаются логические ошибки, то необходимо привлечение опытного программиста.

Предтранслятор имеет также и недостатки, которые, однако, компенсируются большим числом достоинств. Недостатками являются:

Характеристики некоторых предтрансляторов [15, 22, 23]

Наименование предтранслятора	Язык программирования	Максимальные размеры таблиц			Вид таблиц	Оптимизация
		Условия	Действия	Правила		
AGENTA	КОБОЛ	50	100	40	Расширенные	Время работы
DTT	КОБОЛ	25	25	25	Расширенные	Нет
DETAB/65	КОБОЛ	50	50	50	Простые, формально-полные	Проверяются не все условия
VORELLE	КОБОЛ	32	64	64	Расширенные	—
DLT	ФОРТРАН	99	99	64	Расширенные	Объем памяти, время работы
FORTET	ФОРТРАН	99	99	99	Расширенные	Объем памяти, время работы
FOREST	ФОРТРАН	64	64	48	Расширенные	Объем памяти, время работы
PET	ПЛ/1	50	50	30	Простые	Действия
DECTAT	ПЛ/1 или КОБОЛ	64	64	64	Расширенные	Объем памяти, время работы, действия

увеличение времени обработки программ, так как к времени, необходимому для трансляции исходной программы в машинную, добавляется время, необходимое для предварительной трансляции;

чем «мощнее» транслятор (оптимизация, вспомогательные тесты, контроль ошибок), тем дольше длится процесс предтрансляции. Однако иногда некоторые возможности предтранслятора надо использовать лишь частично (например, выдача исходной программы не всегда необходима, особенно при пробной прогонке);

наличие предтранслятора устанавливает границы на использование тех или иных символьных обозначений в ТР, что несколько снижает возможности ТР.

Несмотря на указанные недостатки, предтрансляторы целесообразно применять для эффективной обработки ТР.

Сейчас существует 50—60 таких предтрансляторов. Некоторые из них приведены в табл. 4 (см. также [22, 23]).

Таблица 4

Контроль ошибок в таблицах	Вид алгоритма	Вспомогательный контроль	Примечания
Предусмотрен	Дерево решений и алгоритм масок	—	Расширенные таблицы переводятся непосредственно в программу
Предусмотрен	Вве условия проверяются перед ветвлением	Подсчет избыточности	Большая скорость преобразования
Предусмотрен	Алгоритм масок	—	Предтранслятор сам написан на КОБОЛе
Предусмотрен	—	TRACE, частность правил	Алгоритм реализуется с помощью подпрограммы-ассемблера
Предусмотрен	Локальный алгоритм	Нет	—
Предусмотрен	Алгоритм двоичных масок	Слежение за ходом процесса	—
Предусмотрен	Расширенный, локальный алгоритм	Слежение за ходом процесса	Написан для ЭВМ R21
Предусмотрен	Алгоритм Поллака	—	—
Предусмотрен	Алгоритм ветвящегося дерева	—	

При некотором опыте исходные данные могут быть записаны аналитиком непосредственно на языке ТР. Разработка подробной блок-схемы программы не требуется, и программирование на проблемно-ориентированном языке передается предтранслятору. Проверка программ на синтаксическую правильность в силу простоты языка ТР не является необходимой и подкрепляется многочисленными проверочными и корректирующими подпрограммами. Большинство операторов, содержащихся в исходной программе, синтаксически правильны, так как они записаны предтранслятором, и поэтому временные затраты на контроль ограничены.

Логические ошибки, обнаруженные при контроле машинной программы, должны быть устранены традиционным путем. Опыт показывает, что при использовании таблиц решений встречается меньше ошибок, чем в иных случаях. Документирование частично упрощается, так как программа, построенная для ТР, выдается предтранслятором в такой форме, что ее можно использовать как часть документации.

19. КОМПИЛЯТОР

Компилятор ТР переводит программу ТР непосредственно в машинную программу. Общая структура такого компилятора изображена на рис. 44.

Ввод и обработка таблицы приблизительно соответствуют работе предтранслятора, за исключением того, что проводится общий синтаксический контроль и формируемая программа является машинной программой. Программа ТР обрабатывается как программа на проблемно-ориентированном языке. Это означает, что компилятор

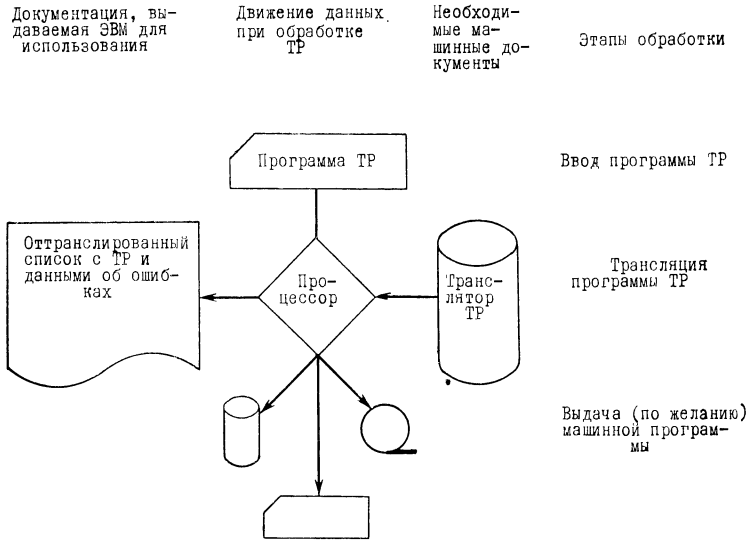


Рис. 44. Блок-схема обработки ТР с помощью транслятора.

ТР, кроме алгоритма для ТР, берет на себя также функции обычного компилятора. Это приводит к бо́льшим затратам времени при создании компилятора, бо́льшим временам трансляции и к повышенным затратам на обслуживание этой программной системы. Эти недостатки, как показывает опыт, не компенсируются достоинствами компилятора, к которым относятся:

перевод программы ТР сразу в машинную программу;

исключение работ с исходной программой;

разветвленная система контроля и исправление ошибок в программе ТР.

На первом этапе вычислительной реализации задачи программа должна быть представлена на языке ТР. Это остается необходимым и осуществляется так же, как и для предтранслятора. Второй и третий этапы отпадают и выполняются компилятором. Проверка машинной программы на логическую правильность должна, однако, выполняться обычным образом, причем использование ТР здесь существенно помогает.

Часть документации поставляется компилятором ТР. Примером компилятора ТР является GECOM/TABSOL. Язык ТР GECOM построен на базе КОБОЛа и АЛГОЛа [23].

ГЛАВА ПЯТАЯ

ПРИМЕНЕНИЕ ТАБЛИЦ РЕШЕНИЙ

20. ПРИМЕР ИЗ ОБЛАСТИ ТЕХНОЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

При одиночном и мелкосерийном производстве, зависящем от требований заказчиков, часто бывает необходимо разрабатывать рабочие документы на геометрически или функционально подобные изделия. Для этого заранее составляют стандартные рабочие графики, которые определяют последовательность рабочих операций. Задача состоит только в том, чтобы установить для каждой рабочей операции необходимые характеристики[1], которые либо являются константами, либо должны рассчитываться как переменные величины в зависимости от специфики обрабатываемых изделий. Такой расчет может быть выполнен с помощью подпрограмм или подтаблиц. Кроме того, должно быть учтено, что задачи по обработке по-разному формируются с помощью различных рабочих процессов в зависимости от обрабатываемых изделий.

Опишем процесс аналитической подготовки подобных задач и их реализации с помощью ТР.

Алгоритм составления требуемых ТР представлен на рис. 45. Исходным пунктом для составления конкретных ТР является изображенный на рис. 46 формуляр решений для анализа задачи.

Этот формуляр содержит столько правил, сколько имеется различных рабочих операций. Для расчета характеристик каждой рабо-

[1] Характеристиками рабочей операции являются данные, служащие для точного описания рабочей операции (например, название, цена, номер рабочего места). — *Прим. автора.*

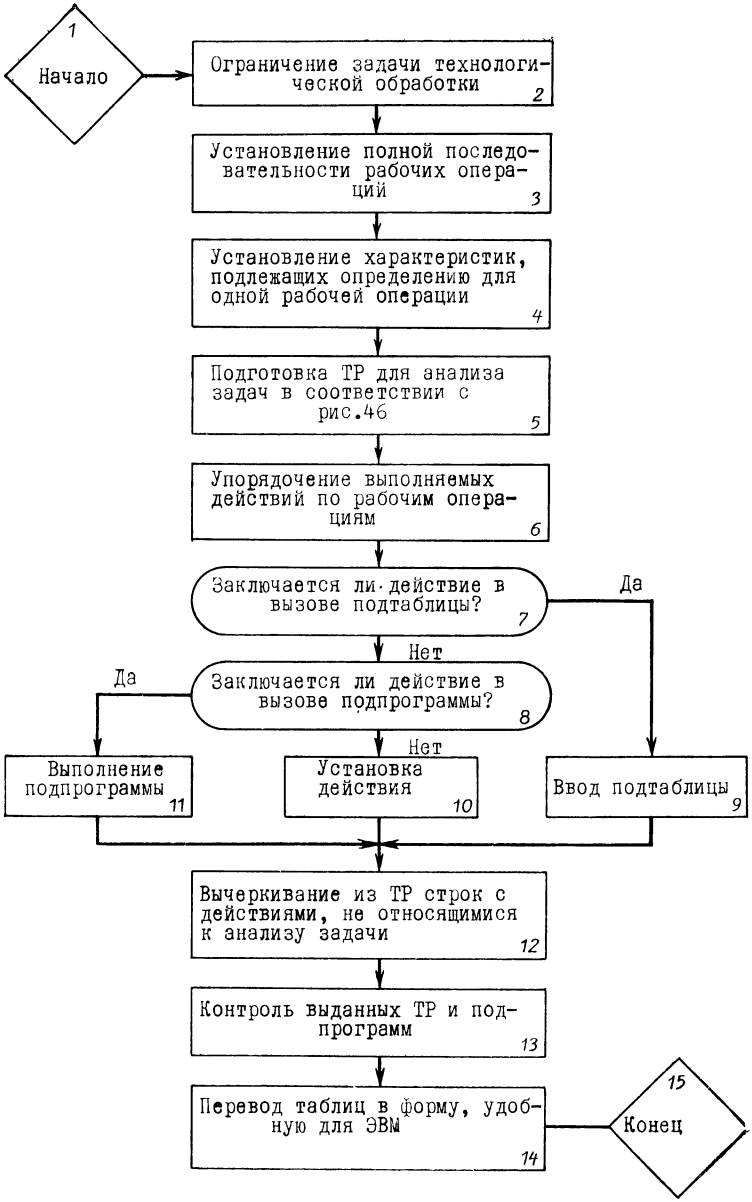


Рис. 45. Последовательность действий для составления ТР при технологическом проектировании.

Пояснения	TP	R1	R2	. . .	Rm — 1	Rm
Номер рабочей операции	РОП N=	N1	N2	. . .	Nm — 1	Nm
Характеристика рабочей операции 1	SET . . . = CALL UP . . . DO TP . . .					
Характеристика рабочей операции 2	SET . . . = CALL UP . . . DO TP . . .					
. . .	. . .					
Характеристика рабочей операции r	SET . . . = CALL UP . . . DO TP . . .					
	РОП N =	РОП N + (N2 — N1)	РОП N + (N3 — N2)	. . .	РОП N + (Nm—Nm—1)	
	GOTO TP	×	×		×	×
	END					

Рис. 46. Формуляр TP для анализа задач при технологическом проектировании.

чей операции необходимо одно действие. Сначала же их число будет в 3 раза больше, так как для каждой характеристики рабочей операции должны быть предусмотрены следующие вычислительные возможности:

1) указано определенное значение или обозначение;
2) характеристики должны быть определены из дальнейшей таблицы;
3) определение осуществляется путем расчетов с помощью подпрограммы.

Пример 15. При обработке литья часто оказывается, что первой операцией является ЧИСТ (чистка) (действие 1), в то время как последующие операции СТРОГ (строгание), ФРЕЗ (фрезерование) зависят от формы и размера детали и определяются с помощью TP (действие 2). Подготовительное время (TAZ) при этом рассчитывается с помощью подпрограммы (действие 3).

После полностью выполненного упорядочения действий для всех рабочих операций вычеркиваются все незанятые строки действий и получается малая таблица, относящаяся к конкретному случаю определения рабочих операций (см., например, рис. 48).

Последующий пример позволяет детально пояснить общий ход решения.

Постановка задачи. Для обработки на металлорежущих станках литых плат длиной до 630 мм и шириной до 630 мм должны быть установлены технологические операции. Платы должны быть обработаны со всех сторон (лицевая и тыльная стороны). Тыльная поверхность, кроме того, должна шлифоваться, в то время как лицевая сторона окрашивается.

Составление таблиц решений. На этом шаге определяется технологическая задача, связанная с постановкой исходной задачи (этап *1* по рис. 45). Необходимая последовательность рабочих операций представлена на рис. 47 (этим закончено выполнение этапа *2* по рис. 45). За число правил принимается количество рабочих операций. Для каждой рабочей операции должны быть определены следующие характеристики (в зависимости от техники обработки некоторые из них, может быть, и не потребуются) (этап *3* на рис. 45):

1) обозначение операции BEZ [обозначениями могут быть: ЧИСТ — чистка, СТРОГ — строгание, ФРЕЗ — фрезерование, СМАЧ — смачивание, ШЛИФ — шлифование, ТОЧ — точение (см. рис. 48—51)];
2) число проходов SAT (чтобы снять слой толщиной d, нужно последовательно снять n слоев толщиной d/n; число n является числом проходов);
3) количество одновременно обрабатываемых изделий MSP (если геометрические формы заготовок позволяют, то пытаются обрабатывать на станке одновременно несколько деталей);
4) группа заработной платы LGR;
5) индекс рабочего места APL (каждое рабочее место на предприятии обозначается пятизначным числом. Каждая цифра имеет точное значение, устанавливаемое стандартами предприятия);
6) стоимость KST;
7) подготовительное время TAZ;
8) штучное время TSZ;
9) многостаночное обслуживание MMB.

В зависимости от типа станков MGR для некоторых рабочих мест предусмотрено многостаночное обслуживание.

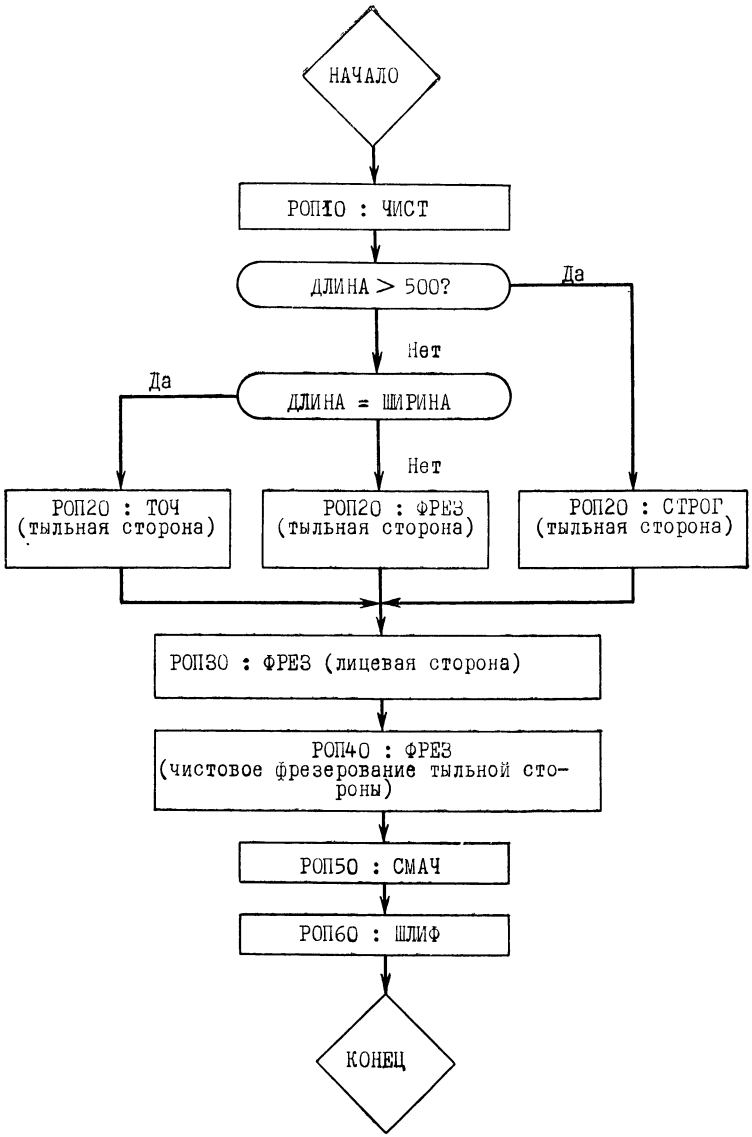


Рис. 47. Последовательность рабочих операций для поверхностной обработки литых прямоугольных плат.

TP53	R1	R2	R3	R4	R5	R6
POПN	010	020	030	040	050	060
SET BEZ =	ЧИСТ		ФРЕЗ	ФРЕЗ	СМАЧ	ШЛИФ
DO		TP BEZ				
DO		TP SAT1		TPSAT2		
DO		TP MSP				
CALL			ППMSP1			
SET LGR =		5	5	5		5
SET APL =	09770				09882	17573
DO		TPAPL	TPAPI	TPAPL		
SET KST =	510	511	511	511	522	511
SET TAZ =		30	30	20		20
CALL		ППTSZ1	ППTSZ2	ППTSZ3		ППTSZ4
DO		TPMMB	TPMMB	TPMMB		TPMMB
SET POПN	020	030	040	050	060	
GOTO TP53	×	×	×	×	×	×
END						

Рис. 48. Управляющая ТР для технологии изготовления литой платы.

С учетом подлежащих определению характеристик рабочих операций и установленной последовательности рабочих операций описанным выше образом должны быть обработаны управляющая таблица по рис. 48 и таблицы на рис. 49—54. Требуемые подпрограммы должны иметься в наличии (этапы *4—8* на рис. 45). Необходимые данные берутся из производственных документов или рассчитываются по формулам:

Подпрограмма ППTSZ1 : расчет штучного времени СТРОГ (РОП20);
" ППTSZ2 : " " " ФРЕЗ (РОП30);
" ППTSZ3 : " " " ФРЕЗ (РОП40);
" ППTSZ4 : " " " ШЛИФ (РОП60);
" ППMSP1 : " числа одновременно обрабатываемых деталей при ФРЕЗ (РОП30);
" ППMSP2 : то же при СТРОГ (РОП20).

TPBEZ	R1	R2	R3
ДЛИНА > 500	Y	N	N
ДЛИНА = ШИРИНА	—	Y	N
SET BEZ =	СТРОГ	ТОЧ	ФРЕЗ
RETURN	×	×	×

Рис. 49. Таблица решений для определения вида обработки поверхности.

TPSAT1	R1	R2	R3	R4	R5
BEZ =	СТРОГ	СТРОГ	СТРОГ	ФРЕЗ	ТОЧ
ШИРИНА < 500	Y	—	—	—	—
ШИРИНА < 630	—	Y	—	—	—
ШИРИНА = 630	—	—	Y	—	—
SET SAT =	2	3	4		2
CALL				ППSAT	
RETURN	×	×	×	×	×

Рис. 50. Таблица решений для установления и расчета числа проходов SAT.

TPMSP	R1	R2	R3	R4	R5
BEZ = ШИРИНА ⩽	ФРЕЗ 315	ФРЕЗ 400	ФРЕЗ 500	СТРОГ —	ТОЧ —
SET MSP = CALL RETURN	4 ×	3 ×	2 ×	 MSP2 ×	1 ×

Рис. 51. Таблица решений для установления и расчета количества одновременно обрабатываемых изделий MSP.

TPAPL	R1	R2	R3	R4	R5
BEZ = ДЛИНА ⩽ 400 ШИРИНА ⩽ 400	ФРЕЗ — Y	ФРЕЗ N N	СТРОГ Y —	СТРОГ N —	ТОЧ — —
SET APL = RETURN	13123 ×	13113 ×	15121 ×	15111 ×	14172 ×

Рис. 52. Таблица решений для выбора рабочего места ASP.

TP MMB	R1	R2	R3	R4	R5	R6	R7
APL =	17573	12302	13113	13123	14172	15111	15121
SET MMB = RETURN	4 ×	1 ×	2 ×	2 ×	2 ×	2 ×	2 ×

Рис. 53. Таблица решений для определения числа одновременно обслуживаемых станков MMB (определяется по номеру рабочего места).

TPSAT2	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
ДЛИНА ⩽	315	400	400	500	500	500	630	630	630	630
ШИРИНА ⩽	315	315	400	315	400	500	315	400	500	630
SET SAT =	4	6	8	8	10	12	10	12	14	16
RETURN	×	×	×	×	×	×	×	×	×	×

Рис. 54. Таблица решений для определения числа проходов SAT (зависит от размеров плат).

РОП N	BEZ	SAT	MSP	LGR	APL	KST	TAZ	TSZ	MMB
010	ЧИСТ				09770	510			
020	СТРОГ	4	2	5	15121	511	30	127	2
030	ФРЕЗ		5	5	13113	511	30	64	2
040	ФРЕЗ	16		5	13113	511	20	43	2
050	СМАЧ				09882	522			
060	ШЛИФ			5	17573	511	20	122	4

Рис. 55. Таблица результатов контрольного примера.

После этого следует вычеркнуть неиспользованные строки действий (этап *9* на рис. 45). Таблицы решений проверяются на правильность и полноту на конкретном числовом примере. Пусть следует определить последовательность рабочих операций и их необходимые характеристики для платы с размерами: длина = ширина = =670 мм. Результат проверки представлен в виде рис. 55 (этап *10* по рис. 45).

Таблицы, обработанные по общим правилам работы с ТР, пригодны для вычислений. Естественно, что в этом случае они должны быть приспособлены к требованиям обрабатывающей системы (этап *11* на рис. 45). Для того, чтобы сделать возможно меньшими затраты на этот шаг обработки, следует учесть последующий способ обработки уже при создании методики и форм представления данных. В нашем примере таблицы должны обрабатываться с помощью предтранслятора, ориентированного на ФОРТРАН. Структура действий и условий уже содержала необходимые для последующего вычисления. Некоторые изменения, конечно, еще необходимы (например, ввод и вывод, засылка начальных значений, передача параметров подпрограммами, корректировка имен).

21. УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ ПРОГРАММЫ

При обработке программ на ЭВМ часто требуется выбрать определенную последовательность из заданного множества формул, модулей, подпрограмм и т. п. При этом выбор и установление временно́й последовательности частей программы осуществляются в зависимости от определенных выходных и соответственно управляющих данных. С помощью ТР эти проблемы можно сделать обозримыми. При этом оказывается возможным установить также последовательность ручных операций и обеспечить документирование.

На приведенном ниже примере будет показано, как с помощью ТР можно представить протекание процесса в зависимости от комбинаций некоторых выходных величин.

Пример 16. Выполнить проверку и сравнение вариантов, основанных на различных комбинациях выходных величин заданного технологического процесса, и определить тенденции его развития, на которые влияют значения валовой и чистой прибыли, а также соотношения видов дохода. Целью этих исследований является выбор по мере надежности экономически наиболее целесообразных вариантов.

В расчеты входят следующие взаимозависимые величины:
цена единицы готовой продукции PRS;
общая себестоимость в плановом году GSK;
прибыль на единицу продукции GWE;
товарная продукция в плановом году WPN;
валовая прибыль в плановом году BGW;
чистая прибыль в плановом году NGW;
распределение налога с производственных фондов PFA;
производственные фонды PRF.

Значения этих величин либо заранее вводятся в память ЭВМ, либо рассчитываются по имеющимся начальным данным. Вычисле-

TP54	R1	R2	R3	R4
GWE > 0	Y	N	N	N
GSK > 0	—	Y	N	N
PRS > 0	—	—	Y	N
CALL	ПП7	ПП8	ПП9	
CALL	ПП6	ПП4	ПП5	
GOTO	TP55	TP55	TP55	TP56

а)

Рис. 56. Таблицы решений к примеру 16. TP54 разбивает задачу на две непересекающиеся подзадачи (случай 1: величины GWE, GSK и PRS заданы или могут быть вычислены; случай 2: величины GWE, GSK и PRS неизвестны) и передает обработку TP55 (случай 1) или TP56 (случай 2).

TP55	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
BGW > 0	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N
NGW > 0	Y	N	N	N	Y	Y	Y	Y	N	N	N	N	N	N	N	N
PFA > 0	—	Y	N	N	Y	N	N	N	Y	Y	Y	Y	N	N	N	N
PRF > 0	—	—	Y	N	—	Y	N	N	Y	Y	N	N	Y	Y	N	N
WPN > 0	—	—	—	—	—	—	Y	N	Y	N	Y	N	Y	N	Y	N
CALL	ПП2	ПП2	ПП2	ПП2	ПП11	ПП13	ПП15		ПП1		ПП1	ПП14	ПП1	ПП13	ПП1	
CALL	ПП10	ПП12	ПП13		ПП2	ПП11	ПП14		ПП12		ПП14		ПП13			
CALL	ПП14	ПП14	ПП12		ПП14	ПП2	ПП2				ПП12		ПП12			
ABBRUCH				×				×		×		×		×	×	×
END	×	×	×		×	×	×		×		×		×			

б)

Рис. 56. (Продолжение).

TP56	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
BGW > 0	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
WPN > 0	Y	N	N	N	N	Y	Y	Y	Y	Y	N	N	N
PFA > 0	—	Y	N	N	N	Y	Y	N	N	N	Y	N	N
PRF > 0	—	—	Y	N	N	—	—	Y	Y	N	—	Y	N
NGW > 0	—	—	—	Y	N	Y	N	Y	N	—	—	—	—
CALL	ПП3	ПП14	ПП13	ПП10		ПП14	ПП14	ПП13	ПП13		ПП14	ПП13	
CALL		ПП12	ПП12	ПП14		ПП16		ПП16					
CALL						ПП7		ПП7					
CALL						ПП8		ПП8					
CALL						ПП11		ПП11					
GOTO	TP54												
ABBRUCH		×	×	×	×		×		×	×	×	×	×
END						×		×					

в)

Рис. 56. (Продолжение)

ние всех величин программы должно оканчиваться либо оператором END (конец), либо оператором ABBRUCH (прерывание).

Для расчетов служат следующие выражения, которые вводятся как подпрограммы:

$$BGW = WPN * GWE \quad (ПП1)$$

$$WPN = BGW/GWE \quad (ПП2)$$

$$GWE = BGW/WPN \quad (ПП3)$$

$$GWE = PRS - GSK \quad (ПП4)$$

$$GSK = PRS - GWE \quad (ПП5)$$

$$PRS = GWE + GSK \quad (ПП6)$$

$$GSK = (GWE * (100 - R))/R \quad (ПП7)$$

$$PRS = (GSK * 100)/(100 - R) \quad (ПП8)$$

$$GWE = PRS * R/100 \quad (ПП9)$$

$$PFA = BGW - NGW \quad (ПП10)$$

$$BGW = NGW + PFA \quad (ПП11)$$

$$NGW = BGW - PFA \quad (ПП12)$$

$$PFA = PRF * 6/100 \quad (ПП13)$$

$$PRF = PFA * 100/6 \quad (ПП14)$$

$$PFA = (GWE * WPN) - NGW \quad (ПП15)$$

$$GWE = (PFA + NGW)/WPN \quad (ПП16)$$

Норма прибыли R — заданная константа.

Для решения сделаем следующие допущения. Все заданные величины положительны, а все остальные величины равны нулю. Это позволяет особенно просто сформулировать условия для ТР. Задача не содержит принципиальных трудностей. Таблицу решений следует записать с восемью условиями PRS>0, GSK>0, ..., PRF>0, ввести все правила и упорядочить соответствующие вычислительные инструкции. При этом требуется некоторая сноровка, чтобы не работать с $2^8=256$ правилами. Так, все правила, которые уже на PFA>0 и PRF>0 отвечают утвердительно (Y), могут быть сэкономлены, так как следует снова обратиться к выражениям ПП13 и ПП14. Таким образом экономятся 64 правила. Остальные выражения позволяют использовать аналогичные соображения. Чем больше сократится таким образом объем, тем затруднительнее будут последующие работы, особенно связанные с анализом полноты. Поэтому укажем несколько иной путь.

Результаты, изложенные в § 6, непосредственно неприменимы, так как не существует группы величин, связанных исключительно между собой. Несмотря на это, такие разложения облегчают представление и решение поставленной задачи.

Спрашивается, какие величины поставляют наибольшую информацию? Очевидно, это величины GWE, GSK или PRS, так как знание одной из этих величин позволяет с помощью ПП7—ПП9 вычислить две остальные. Из этих соображений строится ТР54 (рис. 56,*а*). Таблица однозначно ставит задачи для таблиц ТР55, ТР56

(рис. 56,*б* и *в*). В TP55 величины GWE, PRS и GSK известны, остальные могут быть вычислены. В таблице TP56 величины GWE, PRS и GSK неизвестны. Эти и пять остальных величин должны быть вычислены. Таким образом, задачу вычисления восьми величин мы расчленили на две аналогичные задачи с пятью величинами. Эти таблицы могут быть составлены описанным выше методом.

Таблицы TP55 и TP56 есть TP типа P. Поэтому формальная полнота может быть установлена по критерию, приведенному в § 3.

Протокол (табл. 5) подобно описанному в § 13 (см. табл. 3) содержит результаты выборочной ручной проверки. Каждая строка этого протокола использует результаты ранее идущих строк (соответственно входных данных) и дополняет их величинами, вычисленными в соответствующей таблице (имя таблицы в начале строки). В конце строки помещено имя следующей обрабатываемой таблицы (или ABBRUCH, или соответственно END). В каждой строке известные величины отмечены знаком ×.

Таблица 5

Рабочий протокол проверки программы TP

	PRS	GSK	GWE	WPN	BGW	NGW	PFA	PRF	TP
Вход	×		×	×			×		54
TP54	×	×	×	×			×		55
TP55	×	×	×	×	×	×	×	×	END
Вход		×		×	×				54
TP54	×	×	×	×	×				55
TP55	×	×	×	×	×			×	ABBRUCH
Вход	×		×		×		×		54
TP54	×	×	×		×		×		55
TP55	×	×	×	×	×	×	×		END
Вход				×	×		×		54
TP54				×	×		×		56
TP56			×	×	×		×		54
TP54	×	×	×	×	×		×		55
TP55	×	×	×	×	×	×	×	×	END

ЗАДАЧА ДЛЯ УПРАЖНЕНИЙ

13. Определить производственно-технологические характеристики литой платы длиной 450 мм и шириной 400 мм и подставить результаты в таблицу, соответствующую таблице на рис. 55. Необходимые для этого значения, вычисленные подпрограммами, — следующие: ППSAT—3, ППTSZ1—80, ППTSZ2—34, ППTSZ3—28, ППTSZ4—98 и ППMSP1—7.

РЕШЕНИЯ ЗАДАЧ ДЛЯ УПРАЖНЕНИЙ

1. Из рис. 9,*а* получим:

а) TP9(Y, Y, N) = R2; TP9(Y, N, Y) = E.

б) Правила R1—R4 охватывают шесть ситуаций: R1 и R2 по две, а R3 и R4 по одной. Тогда ситуация S = (Y, Y, N) используется правилом R2 и не должна приниматься во внимание при правиле R4. Для правила ИНАЧЕ имеются всего две ситуации: S1 = = (Y, N, Y) и S2 = (Y, N, N).

в) Так как TP9(Y, Y, N) = R2, то вызывается действие M2. Если поменять местами R2 и R4, то TP9 (Y, Y, N) = R4, т. е. вызывается действие M4.

2. Таблица решений TP57 на рис. 57 дает одно из возможных решений.

TP57	R1	R2	R3
A	> B	< B	—
A	> C	—	< C
B	—	> C	< C
MAX =	A	B	C

Рис. 57. К решению задачи 2.

3. Правила таблицы типа P исключают друг друга, следовательно, число ситуаций, которые содержит такая TP, равно сумме чисел, содержащихся в каждом правиле. Правило с i пробелами содержит точно 2^i ситуаций. Для таблицы с простыми входами с n условиями имеется точно 2^n различных ситуаций. Условие (1) является необходимым и достаточным для определения формальной полноты.

Если таблица TPYk (TPNk) формально полна, то это означает, что для всех ситуаций, чья k-я составляющая равна Y(N), существует одно правило. В этом случае TP, конечно, тоже полная.

Пусть, наоборот, TP полна. Тогда по определению для всех ситуаций, чья k-я составляющая равна Y(N), должно быть предусмотрено правило, т. е. TPYk (TPNk) полны.

Второй критерий, и это еще поясняется доказательством, не ограничивается таблицами типа P.

4. Решение показано в TP58 (рис. 58,*а*). В соответствии с предположениями все числа различны. Следовательно, справедливы высказывания A > B (соответственно A > C; B > C) или A < B (соответственно A < C; B < C).

При этих предположениях получим эквивалентную TP59 (рис. 58,*б*).

а) Это таблица типа P.

б) Таблица формально не полна, так как ситуации S1 = = (Y, N, Y) и S2 = (N, Y, N) не включены.

ТР58	R1	R2	R3
A > B	Y	—	—
A < B	—	Y	—
A > C	Y	—	—
A < C	—	—	Y
B > C	—	Y	—
B < C	—	—	Y
MAX =	A	B	C

а)

ТР59	R1	R2	R3
A > B	Y	N	—
A > C	Y	—	N
B > C	—	Y	N
MAX =	A	B	C

б)

Рис. 58. К решению задачи 4.

в) Таблица логически полна, так как ситуация S1 означает A>B; C>A и B>C; ситуация S2 означает B>A; A>C и C>B. Следовательно, обе ситуации невозможны.

5. Таблица относится к типу P и содержит семь ситуаций, а следовательно, формально не полна. В ней отсутствует ситуация S=(N, Y, Y), что означает A=0, B=A и B=—1. Так как эта ситуация невозможна, таблица логически полна.

6. Запишем таблицу решений и условий Ci=0 ($i=\overline{1,5}$). Правила получаются, если на рис. 24 пройти в лексикографической последовательности (от Y к N). В результате получается ТР60 (рис. 59,*а*). Для надежности перепроверим полноту таблицы. Для этого одинаково хорошо применимы оба критерия. На основании п. «г» § 4 правила R2 и R4 (так же как R3 и R7; R5 и R9) могут быть объединены. В результате получается ТР61 (рис. 59,*б*). Далее можно снова объединить правила R2 и R3 и получить ТР62 (рис. 59,*в*). Из последней таблицы следует блок-схема, изображенная на рис. 59,*г*, которая эквивалентна ТР, изображенной на рис. 23.

ТР60	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
C1 = 0	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
C2 = 0	Y	N	N	N	N	N	N	N	N	—
C3 = 0	—	Y	Y	Y	Y	N	N	N	N	—
C4 = 0	—	Y	Y	N	N	Y	Y	N	N	—
C5 = 0	—	Y	N	Y	N	Y	N	Y	N	—
ДЕЙСТВИЕ	2	3	4	3	1	3	4	3	1	1

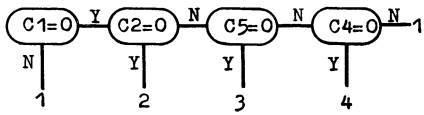
а)

ТР61	R1	R2	R3	R4	R5	R6
C1 = 0	Y	Y	Y	Y	Y	N
C2 = 0	Y	N	N	N	N	—
C3 = 0	—	Y	N	—	—	—
C4 = 0	—	—	—	Y	N	—
C5 = 0	—	Y	Y	N	N	—
ДЕЙСТВИЕ	2	3	3	4	1	1

б)

ТР62	R1	R2	R3	R4	R5
C1 = 0	Y	Y	Y	Y	N
C2 = 0	Y	N	N	N	—
C4 = 0	—	—	Y	N	—
C5 = 0	—	Y	N	N	—
ДЕЙСТВИЕ	2	3	4	1	1

в)



г)

Рис. 59. К решению задачи 6.

7. Возможное решение представлено таблицей ТР63 (рис. 60).

8. Пусть ТР относится к типу Р. Тогда все частные таблицы TPYk и TPNk ($1 \leqslant k \leqslant n$) также типа Р.

Если Ri′ и Rj′ — два правила из TPYk, не отличающиеся ни одним входом, не содержащим пробелов, то, так как ТР принадлежит к типу Р, исходные правила Ri и Rj таблицы в k-м условии должны различаться. При этом Ri′ и Rj′ не могут быть правилами частной таблицы TPYk.

TP63	R1	R2	R3	R4	R5	R6
SSS	Y	Y	Y	N	N	N
MS	Y	Y	N	Y	N	Y
SMS	N	Y	—	Y	—	N
C=	16	25	25	16	16	10

Рис. 60. К решению задачи 7.

TP64	R1	R2	R3	R4	R5	R6	R7	E
B2	Y	Y	N	N	Y	Y	N	
B4	N	N	—	—	Y	N	—	
B5	Y	N	N	N	N	N	Y	
B6	Y	Y	N	Y	—	N	N	
M1	×	×						
DO TP65	×	×	×	×	×	×	×	×
M4		×	×	×	×	×		
M5			×				×	
END	×	×	×	×	×	×	×	×

а)

TP65	R1	R2	E
B1	Y	N	
B2	N	N	
M2	×		
M3	×	×	
RETURN	×	×	×

б)

Рис. 61. К решению задачи 9.

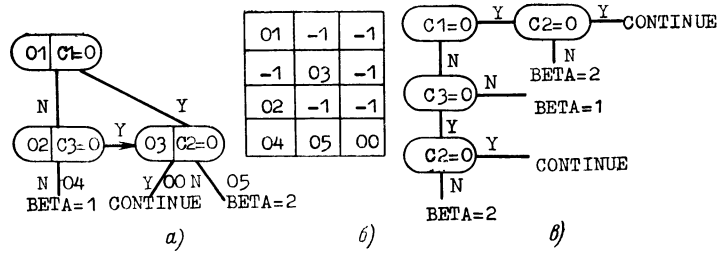


Рис. 62. К решению задачи 10.

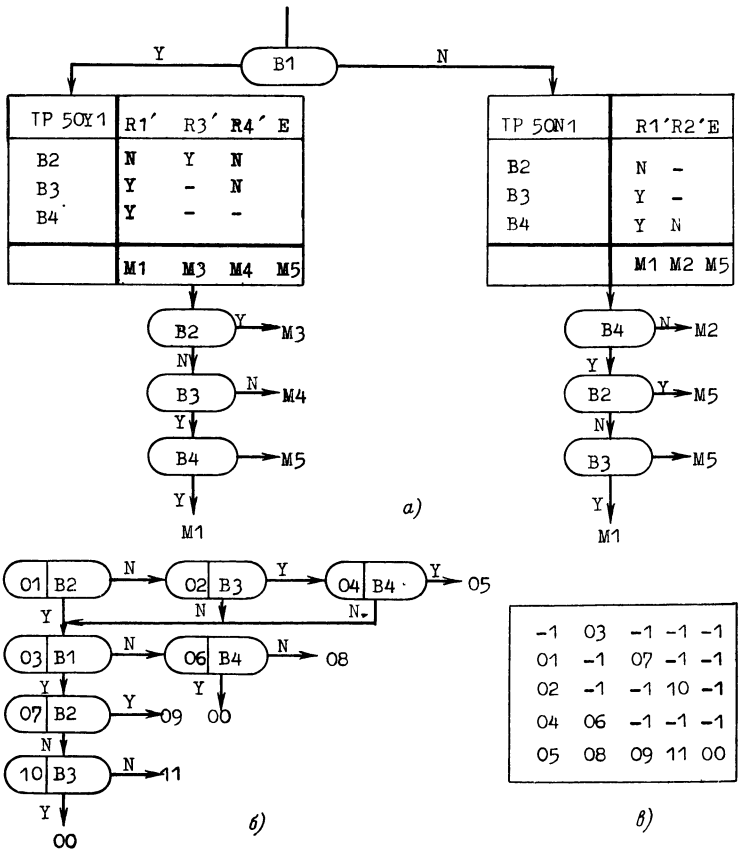


Рис. 63. К решению задачи 11, п. «а».

9. Решение приведено на рис. 61 (ТР64 и ТР65).

10. Локальный алгоритм приводит к блок-схеме, изображенной на рис. 62,*а*. Для контроля на рис. 62,*б* приведена матрица меток. Алгоритм Поллака дает для эквивалентной таблицы по рис. 39 блок-схему, представленную на рис. 62,*в*.

а) Программа, полученная применением локального алгоритма, требует меньшего объема памяти.

б) Для средней длины ветви среднего времени работы (программы) имеем:

$$\bar{w}=(2\cdot 2+2\cdot 2+2\cdot 2+2\cdot 3)/8=2{,}25 \text{ для рис. 62,}а;$$

$$\bar{w}=(2\cdot 2+2\cdot 2+2\cdot 2+2\cdot 3)/8=2{,}25 \text{ для рис. 62,}в.$$

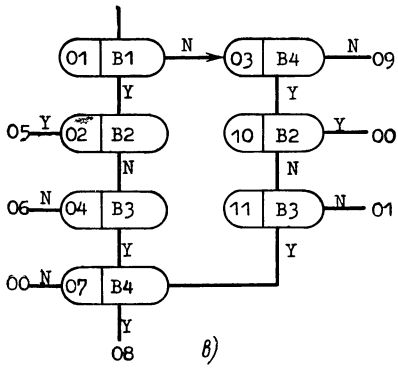
11. а) Решение дано на рис. 63,*а*, *в*. Обе программы имеют одинаковый объем памяти, однако у второй программы среднее время реализации больше.

ТР66	R1	R2	R3	R4	E
B1	Y	Y	N	—	
B2	N	Y	—	N	
B3	N	—	—	Y	
B4	—	—	N	Y	
	M4	M3	M2	M1	M5

а)

01	—1	—1	—1	—1
02	—1	—1	10	—1
04	—1	—1	11	—1
—1	—1	03	07	—1
06	05	09	08	00

б)



в)

Рис. 64. К решению задачи 11 п. «б».

б) Решение (рис. 64) показывает, что локальный алгоритм при таблицах типа Р зависит от порядка правил. В предыдущем случае можно вообще ветвь Y проверки 11/B3 соединить сразу с 08, так как 11/B3 работает только тогда, когда 03/B4 уже выбрала ответ Y (см. пояснение на с. 43).

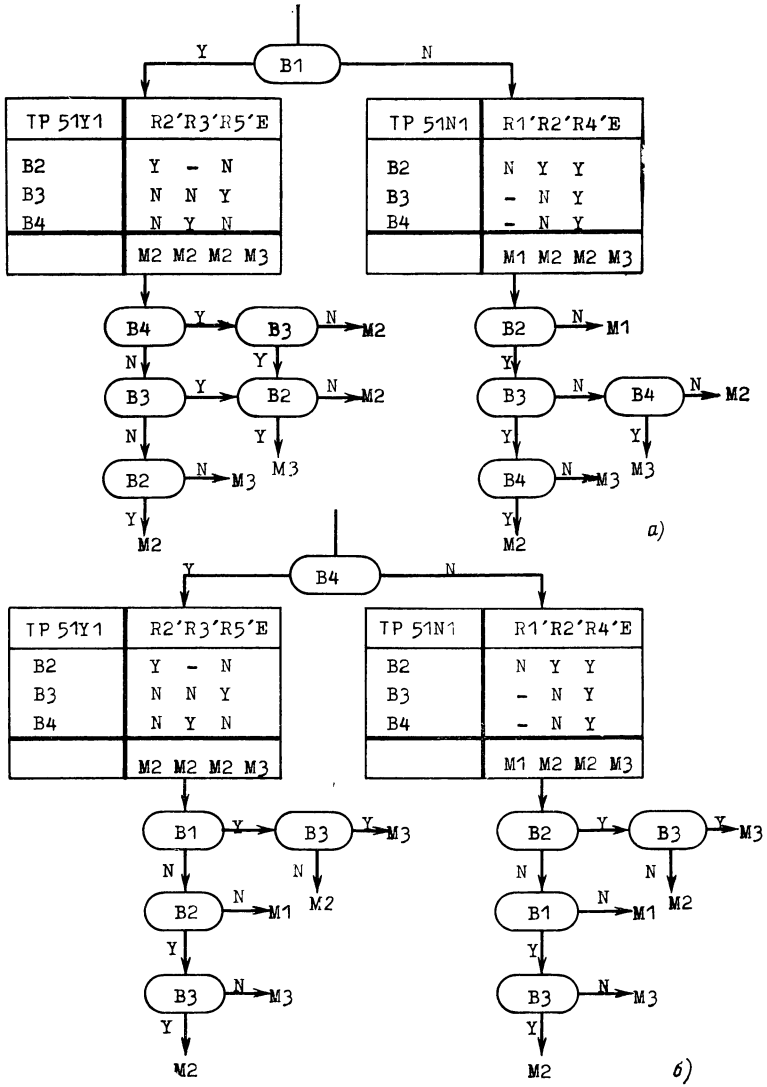


Рис. 65. К решению задачи 12.

12. Число пробелов для первых двух строк равно. Для строчных коэффициентов имеем:

$$\zeta_1 = |2+1-4-1| = 2;$$

$$\zeta_2 = |2+1-4-1| = 2.$$

Начинаем поэтому с проверки первого условия. Обработка получающихся подтаблиц несложна. Получим рис. 65,*а*. Если начать в этой блок-схеме с проверки B4 и использовать алгоритм Поллака, то получится рис. 65,*б*. Средняя длина ветви обеих блок-схем равна 3,25, но вторая блок-схема содержит на одну проверку меньше.

13. Решение изображено на рис. 66.

РОПN	BEZ	SAT	MSP	LGR	APL	KST	TAZ	TSZ	MMB
010	ЧИСТ				09770	518			
020	ФРЕЗ	3	4	5	13123	511	30	80	2
030	ФРЕЗ		7	5	13123	511	30	34	2
040	ФРЕЗ	8		5	13123	511	20	38	2
050	СМАЧ				09892	522			
060	ШЛИФ			5	17079	511	20	98	4

Рис. 66. К решению задачи 13.

УКАЗАТЕЛЬ ОБОЗНАЧЕНИЙ

A — матрица действий
Bi — условие i
b_{ij} — элемент матрицы условий
C — матрица меток при локальном алгоритме
TP — обозначение (имя) произвольной таблицы решений
TPYi — частная таблица TP по i-му условию для ветви Y
TPNi — частная таблица TP по i-му условию для ветви N
e — текущий индекс
g — текущий индекс
i — текущий индекс
j — текущий индекс
k — текущий индекс
l — количество действий
Mi — действие
m — количество правил
N — нет (логическая «ложь»)
n — количество условий
Ri — правило i
Rk′ — правило k в частной таблице
r — текущий индекс
r_k — количество пробелов в k-м столбце
Si — ситуация i
s — текущий индекс
$\underline{w}$ — длина пути (ветви)
w — средняя длина ветви
Y — да (логическая «истина»)
Z — характеристика правил
α_i — относительная частота ситуации
β — пробельное число
ε — элемент вектора ситуаций
σ — коэффициент столбца
ζ — строчный коэффициент

СПИСОК ЛИТЕРАТУРЫ

1. Autorenkollektiv: Automatisierte kundenwunschabhängige Vorbereitung der Produktion (AKV) — Teil I. AUTEVO-Informationsreihe des FLZ AUTEVO Jena im VEB Carl Zeiss JENA.

2. **Chapin N.** Parsing of Decision Tables. — Communications of the ACM **10** (1967) **8**, p. 507—512.

3. **Strunz H.** Eine Methode zur Zergliederung von Entscheidungstabellen. — Angewandte Informatik **13** (1971) 3, S. 117—122.

4. **McDaniel H.** Specific Applications of Decision Tables. — In: Applications of Decision Tables. Princeton: Brandon/Systems Press, Inc 1970, p. 215—226.

5. **Schmidt D. T., Kavanagh T. F.** The Use of Decision Tables.— In: McDaniel H. (Ed.). Applications of Decision Tables. Princeton: Brandon/Systems Press, Inc 1970, p. 1—19.

6. **Reinwald L. T., Soland R. M.** Conversion of Limited-Entry Decisiion Tables to Optimal Computer Programs I: Minimum Average Processing Time. — Journal of the Association for Computing Machinery **13** (1966) 3, p. 339—358.

7. **Reinwald L. T., Soland R. M.** Conversion of Limited-Entry Decision Tables to Optimal Computer Programs II: Minimum Storage Requirement. — Journal ACM **14** (1967), p. 742—756.

8. **Verhelst M.** The Conversion of Limited-Entry Decision Tables to Optimal and Near-Optimal Flowcharts: Two New Algorithms. — Communications of the ACM **15** (1972) 11, p. 974—980.

9. **Авен О. И., Душинский В. А.** Оптимизация машинных программ при помощи таблиц решений. — Экономика и математические методы. 5 (1969), 6, с. 902—908.

10. **Schwayder K.** Conversion of Limited-Entry Decision Tables to Computer Programs — A Proposed Modification to Pollack's — Algorithm. — Communications of the ACM **13** (1971) 2, p. 69—73.

11. **Press L.** Conversion of Decision Tables to Computer Programs. — Communications of the ACM **8** (1965) 6, p. 385—390.

12. **Kirk H. W.** Use of Decision Tables in Computer Programming. — Communications of the ACM **8** (1965) 1, p. 41—43.

13. **King P. J. H.** Conversion of Decision Tables to Computer Programs by Rule Mask Technique. — Communications of the ASM **9** (1966) 11, p. 796—801.

14. **Pollack S. L.** Conversion of Limited — Entry Decision Tables to Computer Programs. — Communications of the ACM **8** (1965) 11, p. 677—682.

15. Entscheidungstabellenumwandler: Programmbeschreibung IBM Form 79 975-0, Code Nummer 360 A-CX-32X.

16. **Veinott C. G.** Programming Decision Tables in FORTRAN, COBOL or ALGOL. — Communications of the ACM **9** (1966) 1, p. 31—35.

17. **Anderson Chr.** ALGOL 60 — eine Sprache für Rechenautomaten. — REIHE AUTOMATISIERUNGSTECHNIK Bd 47. Berlin: VEB Verlag Technik 1968.
18. **Paulin G.** FORTRAN — Datenbeschreibung/Unterprogrammtechnik. — REIHE AUTOMATISIERUNGSTECHNIK Bd 74. Berlin: VEB Verlag Technik.
19. **Schiller W.** Programmiersprache PL/1. Berlin: VEB Verlag Technik 1971.
20. **Knoch W.** Entscheidungstabellen und ihre praktische Anwendung. Rechentechnik/Datenverarbeitung **10** (1973) 12, S. 33—37.
21. ALP/DOS Decision Table Makros. IBM-Programmbeschreibung 360 D 03.7.017.
22. **Hoffmann K.** Automatische Umwandler für Entscheidungstabellen. Bürotechnik bta/bto **21** (1973) 1, S. 28—35.
23. **Hughes M. L., Shank R. M., Stein E. S.** Decision Tables. MDI Publications, New York (1968). p. 147.
24. Autorenkollektiv: FORTRAN — orientierter Entscheidungstabellenübersetzer für ESER-Anlagen — Anwenderhandbuch, Fehlerhandbuch, Bedienerhandbuch — FLZ AUTEVO Jena im VEB Carl Zeiss JENA, 1975.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ

Герд Фрайтаг, Вольфганг Годе, Хорст Якоби, Хейнц Лаутц, Джордж Симон, Ульрих Шпиттель

ВВЕДЕНИЕ В ТЕХНИКУ РАБОТЫ С ТАБЛИЦАМИ РЕШЕНИЙ

Редактор издательства Н. А. Медведева
Обложка художника А. А. Иванова
Технический редактор Н. М. Пушкарева
Корректор З. Б. Драновская
ИБ № 2140

Сдано в набор 19.04.79 Подписано в печать 22.06.79
Формат 84×108¹/₃₂ Бумага типографская № 1 Гарн. шрифта литературная
Печать высокая Усл. печ. л. 4,62 Уч. изд. л. 5,31
Тираж 11 000 экз. Заказ 126 Цена 35 к.

Издательство «Энергия», 113114, Москва, М-114, Шлюзовая наб., 10
Московская типография № 10 Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли, 113114, Москва, М-114, Шлюзовая наб., 10