Table Program and its Conversion to Computer Program —An Extension of Decision Table—

SHINJI MORIYA* AND KEIJI HIRAMATSU*

Introduction

Decision tables are a fairly good method for analyzing and documenting systems, but it is pointed out that we cannot satisfactorily describe program segments with loops or with actions between conditions by decision tables. In this paper these defects of decision tables are perfectly improved only by adding the following two functions; one is to allow the program loops by labeling statement labels to both conditions and actions of decision tables. The other is to define two interpretations, 'don't care' and 'neglect', for blanks in condition entry to avoid the ambiguity brought into by labeling the statement labels. These have important meanings that the improved decision table becomes a programming language which describes an algorithm by very simple tabular form. The name 'Table Program' will be suitable for the improved decision table.

General Format of Table Program

General format of table program is illustrated in Fig. 1. Each field in Fig. 1 has the following meanings.

Entry 'Head'

	name		R_1	⋯	R_j	⋯	R_n	E_1	⋯	E_s	⋯	E_v
	LC_1	C_1	C_1R_1	⋯	C_1Rj	⋯	C_1R_n	C_1E_1	⋯	C_1E_s	⋯	C_1E_v
	⋮	⋮	⋮		⋮		⋮	⋮		⋮		⋮
entry C_i ⟹	LC_i	C_i	C_iR_1	⋯	C_iR_j	⋯	C_iR_n	C_iE_1	⋯	C_iE_s	⋯	C_iE_v
	⋮	⋮	⋮		⋮		⋮	⋮		⋮		⋮
	LC_m	C_m	C_mR_1	⋯	C_mRj	⋯	C_mR_n	C_mE_1	⋯	C_mE_s	⋯	C_mE_v
	LA_1	A_1	A_1R_1	⋯	A_1R_j	⋯	A_1R_n	A_1E_1	⋯	A_1E_s	⋯	A_1E_v
	⋮	⋮	⋮		⋮		⋮	⋮		⋮		⋮
entry A_k ⟹	LA_k	A_k	A_kR_1	⋯	A_kR_j	⋯	A_kR_n	A_kE_1	⋯	A_kE_s	⋯	A_kE_v
	⋮	⋮	⋮		⋮		⋮	⋮		⋮		⋮
	LA_u	A_u	A_uR_1	⋯	A_uRj	⋯	A_uR_n	A_uE_1	⋯	A_uE_s	⋯	A_uE_v

Fig. 1. General format of table program

This paper first appeared in Japanese in Joho-Shori (Journal of the Information Processing Society of Japan), Vol. 13, No. 1 (1972), pp. 13~21.

* Information Science Laboratory, Tokyo Electrical Engineering College

name: name of table program
R_j: rule number ($j=1, \cdots, n$), n is a number of rules.
C_i: condition ($i=1, \cdots, m$), m is a number of conditions.
A_k: action ($k=1, \cdots, u$), u is a number of actions.
C_iR_j: description of condition test.
A_kR_j, A_kE_s: description of execution.
E_s: else rule ($s=1, \cdots, v$), v is a number of else rule.
LC_i: statement label on condition C_i.
LA_k: statement label on action A_k.
entry Head: one of entry points to table program, which is the same as unique entry point of decision table.
entry C_i: entry point to the ith condition C_i.
entry A_k: entry point to the kth action A_k.

1. *General Rules of Table Program*

Let a string of blanks and a statement label be 'blank' and 'label' respectively. 'LC_i=label' means a statement label is placed in LC_i. Similarly, 'LC_i=blank' means no statement label is placed in LC_i. We will use similar notations for another elements of table program.

RULE 1: Give two interpretations 'don't care' or 'neglect' for blanks in condition entry. A blank may have both of these interpretations at distinct instance but only one interpretation at a time.

RULE 2: $(\forall_i)(\forall_j)(LC_i=\text{blank}\Rightarrow C_iR_j=\text{don't care})$

RULE 3: If C_iR_j=don't care, blank C_iR_j may be either Y (Yes) or N (No).

We denote 'entry=Head' to describe that a control is sent to entry Head. 'entry=C_i' and 'entry=A_k' are similarly defined.

RULE 4: entry=C_1 is equivalent to entry=Head and LC_1=label.

RULE 5: $(\forall_i)(\forall_j)(\text{entry}=C_i \text{ and } C_iR_j=\text{blank}\Rightarrow C_iR_j=\text{neglect})$

RULE 6: If C_iR_j=neglect, rule R_js are never selected.

In the following RULEs, we assume $p=i+1, i+2, \cdots, m$.

RULE 7: $(\forall_i)(\forall_j)(\forall_p)(\text{entry}=C_i \ \&\ C_iR_j\neq\text{blank} \ \&\ C_pR_j=\text{blank}\Rightarrow C_pR_j=\text{don't care})$.

RULE 8: $(\forall_i)(\forall_j)(\text{entry}=\text{Head} \ \&\ LC_1=\text{blank} \ \&\ C_iR_j=\text{blank}\Rightarrow C_iR_j=\text{don't care})$.

The notation '$(C_a, C_b, \cdots, C_z)$=decision' is used in the case when $C_a, C_b, \cdots, C_z$ are the conditions to be tested in this order during execution.

RULE 9: $(\forall_i)(\text{entry}=C_i\Rightarrow(C_i, C_{i+1}, \cdots, C_m)=\text{decision})$.
entry=Head$\Rightarrow(C_1, C_2, \cdots, C_m)$=decision.

RULE 10: For all j, if the combination of Y (Yes) or N (No) which is decided by condition test

(1) is identical with unique combination of $C_qR_j(q=i, i+1, \cdots, m)$ then the rule R_j is selected.

(2) is identical with more than one combinations of $C_qR_j(q=i, i+1, \cdots, m)$ then

there are ambiguities between these combinations.

(3) does not identical with any combination of $C_qR_j(q=i, i+1, \cdots, m)$ then such else rule E_s as C_iE_s=mark 'X' is selected.

'$(A_a, A_b, \cdots, A_z)$=execution' is used to express the case when these $A_a, A_b, \cdots, A_z$ are actions to be executed in that order during execution.

RULE 11: If for all j, rule R_j(or E_s) is selected then $(\cdots, A_k, \cdots)$=execution for all k such that $A_kR_j \neq$ blank (or $A_kE_s \neq$ blank).

RULE 12: If for all k, there exists unique j such that entry$=A_k$ & $A_kR_j \neq$ blank (or entry$=A_k$ & $A_kE_s \neq$ blank) then $(A_k, \cdots, A_r, \cdots)$=execution for such r as $A_rR_j \neq$ blank (or $A_rE_s \neq$ blank).

2. *An Example of Table Program*

In this section, we will compare the capability of table program with that of decision table. An algorithm represented by the flowchart shown in Fig. 2 is identically expressed by three decision tables $D1$, $D2$ and $D3$ as in Fig. 3. Segments $S1$, $S2$ and $S3$ of the flowchart correspond to $D1$, $D2$ and $D3$ respectively. To express these algorithms with loops or with actions between conditions only by means of decision tables, open and/or closed decision tables must be linked as shown in this example.

Now, we show a table program TP1 in Fig. 4 which expresses the algorithm of the flowchart in Fig. 2. Let us apply the RULEs described in the last section to TP1.

(a) Upon entry to Head or entry to C_1

By RULE 7, all of blanks of TP1 become 'don't care'. Then, if C_1=FALSE, rule

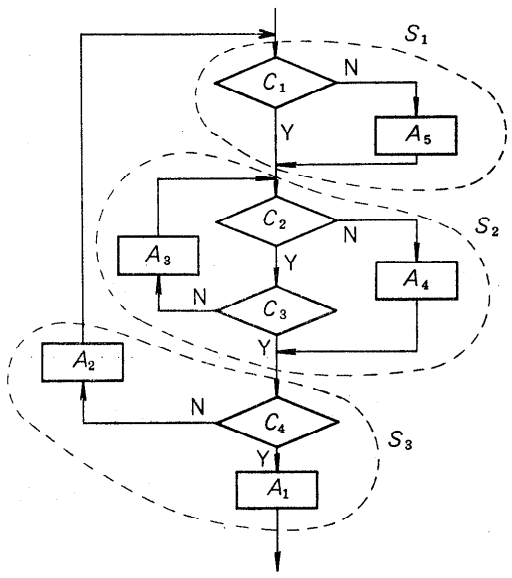

Fig. 2 An example of flowchart

R_5 will be selected (by RULE 9 and RULE 10), and actions A_5 and GO TO 2 will be executed.

(b) Upon entry to C_2

By RULE 9, conditions C_2, C_3 and C_4 will be tested. If we apply usual rules of decision tables to this case, blank C_2R_5 will become don't care and therefore rules of R_1 through R_5 will be ambiguous. If we apply RULE 5 to this case, blank C_2R_5 will be interpreted 'neglect' and rule R_5 will never be selected (by RULE 6).

By applying other RULEs to table program TP1, we will find the algorithm expressed by TP1 is identical with that of the flowchart shown in Fig. 3.

D_1	R_1	R_2
C_1	Y	N
A_5		X
Go To D_2	X	X

D_2	R_1	R_2	R_3
C_2	Y	Y	N
C_3	Y	N	
A_3		X	
A_4			X
Go To D_2		X	
Go To D_3	X		X

D_3	R_1	R_2
C_4	Y	N
A_1	X	
A_2		X
Go To D_1		X

Fig. 3 Table program and its conversion to computer program

TP 1		R_1	R_2	R_3	R_4	R_5
1	C_1	Y	Y	Y	Y	N
2	C_2	Y	Y	Y	N	
	C_3	Y	Y	N		
4	C_4	Y	N			
	A_1	X				
	A_2		X			
	A_3			X		
	A_4				X	
	A_5					X
	Go To 1		X			
	Go To 2			X		X
	Go To 4				X	

Fig. 4 Table program which algorithm is identical with those of Fig. 2 and Fig. 3

3. *Table Program and Decision Table*

The rules of decision tables can be described by RULE 2, RULE 3, in the case of $i=1$ of RULE 7, RULE 8, in the case of entry=Head of RULE 9, in the case of $i=1$ of RULE 10 and RULE 11 shown in the preceeding section in this paper. The format of table program is the same as that of decision table with

statement labels placed on both conditions and actions. Therefore we can conclude the relation

Table Program $\supseteq$ Decision Table

holds.

4. *Conversion of Table Programs to Computer Programs*

In this paper, an algorithm converting limited entry table program (LETP) which is an improvement of Muthukrishnan's one [1] is suggested. Algorithm converting mixed entry table program (METP) can be also obtained by similar improvement, but it is not described in this paper.

STEP 1: Generate two $m\times n$ matrices $T=[t_i{}^j]$ and $F=[f_i{}^j]$ by substituting the following codes for the condition entry in the given table program.

$$t_i{}^j=\begin{cases}1\cdots\text{if } C_iR_j=Y \text{ or } C_iR_j=\text{blank}\\ 0\cdots\text{if } C_iR_j=N\end{cases}$$

$$f_i{}^j=\begin{cases}1\cdots\text{if } C_iR_j=N \text{ or } C_iR_j=\text{blank}\\ 0\cdots\text{if } C_iR_j=Y\end{cases}\quad (i=1,\cdots,m;\ j=1,\cdots,n)$$

REMARK. Muthukrishnan [1] generates $2m\times n$ matrix M as following;

$Y\rightarrow_0{}^1$, $N\rightarrow_1{}^0$, blank$\rightarrow_1{}^1$.

Since $T_{m\times n}$ and $F_{m\times n}$ in this paper are the submatrices of $M_{2m\times n}$, there are no intrinsic differences in this step between Muthukrishnan [1] and this paper.

STEP 2: When entry$=C_i$, it becomes $(C_i, C_{i+1}, \cdots, C_m)$=decision by RULE 9. Then the conditions $C_i, C_{i+1}, \cdots, C_m$ are tested for a given record. Here generate a $(m-i+1)\times n$ matrix $D=[d_p{}^j]$ by substituting the following vectors taking the results of the condition tests in account.

$$\boldsymbol{d}_p=\begin{cases}\boldsymbol{t}_p\cdots\text{if the } p\text{th condition is satisfied}\\ \boldsymbol{f}_p\cdots\text{if the } p\text{th condition is not satisfied}\end{cases}$$
$$(p=i, i+1, \cdots, m;\ j=1, \cdots, n),$$

where $\boldsymbol{d}_p$, $\boldsymbol{t}_p$ and $\boldsymbol{f}_p$ are the pth vectors of the matrices D, T and F respectively.

STEP 3: Generate a n-dimensional row vector $\boldsymbol{d}=[d^j]$. The jth element d^j is obtained by $d^j=\bigwedge_{p=i}^{m} d_p{}^j$ where the symbol $\wedge$ is used to express logical AND.

STEP 4: Generate a n-dimensional row vector $\boldsymbol{r}=[r^j]$ as follows.

If entry=Head & LC_1=blank then $\boldsymbol{r}=\boldsymbol{d}$ where $r^j=d^j$.

If entry$=C_i$ then $\boldsymbol{r}=\boldsymbol{d}\wedge(\boldsymbol{t}_i\oplus\boldsymbol{f}_i)$ where $r^j=d^j\wedge(t_i{}^j\oplus f_i{}^j)$.

The symbol $\oplus$ is used to express exclusive OR.

REMARK. In this step, the essential differences between Muthukrishnan [1] and this paper is on the consideration of entry$=C_i$.

STEP 5: (a) If there is unique j such that $r^j=1$ then R_j is the rule selected.

(b) If there are plural js such that $r^j=1$ then there are ambiguities among these R_js.

(c) If there is no j such that $r^j=1$ then rule ELSE is selected.

Example. The above algorithm is explained with reference to the limited entry table program of Fig. 5.

		R1	R2	R3
	C1	Y	N	
2	C2		Y	N
3	C3	N	N	Y

Fig. 5. An example of limited entry table program

Matrices T and F are generated as follows.

$$T=\begin{bmatrix}1&0&1\\1&1&0\\0&0&1\end{bmatrix},\quad F=\begin{bmatrix}0&1&1\\1&0&1\\1&1&0\end{bmatrix}$$

(1) If entry=Head, C1=TRUE, C2=FALSE and C3=TRUE then

$$D=\begin{bmatrix}\boldsymbol{t}_1\\\boldsymbol{f}_2\\\boldsymbol{t}_3\end{bmatrix}=\begin{bmatrix}1&0&1\\1&0&1\\0&0&1\end{bmatrix},\quad \boldsymbol{d}=[0\ 0\ 1],\quad \boldsymbol{r}=\boldsymbol{d}=[0\ 0\ 1],$$

therefore rule R3 is selected.

(2) If entry=C2, C2=TRUE and C3=FALSE then

$$D=\begin{bmatrix}\boldsymbol{t}_2\\\boldsymbol{f}_3\end{bmatrix}=\begin{bmatrix}1&1&0\\1&1&0\end{bmatrix},\quad \boldsymbol{d}=[1\ 1\ 0],\quad \boldsymbol{r}=\boldsymbol{d}\wedge(\boldsymbol{t}_2\oplus\boldsymbol{f}_2)=[0\ 1\ 0],$$

therefore rule R2 is selected.

(3) If entry=C3 and C3=FALSE then

$$T=[\boldsymbol{f}_3]=[1\ 1\ 0],\quad \boldsymbol{d}=[1\ 1\ 0],\quad \boldsymbol{r}=\boldsymbol{d}\wedge(\boldsymbol{t}_3\oplus\boldsymbol{f}_3)=[1\ 1\ 0],$$

therefore this indicates an ambiguity between rule R1 and R2.

5. *Impremention*

An experimental table language preprocessor system 'TFORTRAN' for the FACOM 230-60 computer has been developed by authors for converting mixed entry table programs into computer programs. The preprocessor accepts open and/or closed METP embedded in a FORTRAN IV program and outputs a FORTRAN IV program.

6. *Conclusions*

Table program has been developed by adding the following two functions to decision table; the first, program loops are allowed by labeling statement labels on both conditions and actions of decision table, the second, the interpretations 'don't care' and 'neglect' for the blanks in condition entry are given to avoid the ambiguities brought into by labeling the statement labels. It was shown that an algorithm converting table programs into computer programs

can be obtained only by adding several procedures to that of decision tables, and that the relation "Table program $\supseteq$ Decision table" holds. Thus, we can describe most algorithms more clearly and simply by using table programs than using decision tables. We have become convinced that table program is a preferable method for not only analyzing and documenting systems but also giving a clear concepts of programming language.

Acknowledgement

The authors wish to acknowledge the serious discussions provided by Prof. K. Mano, Aoyama Gakuin University.

Reference

[1] C. R. Muthukrishnan and V. Rajaraman: On the Conversion of Decision Tables to Computer Programs. *Comm. ACM 13*, 6 (June 1970), pp. 347–351.