Decision Table*

吉村鉄太郎**

1. Decision Table

電子計算機のプログラムはいくつかの論理判断を含んでおり，その判断結果に従ってそれぞれ異なる処理過程を実行するようになっている．また，これらの処理過程でさらに何回かの判断を行なうのがふつうである．プログラムを書くためにはまず対象とする問題の分析からはじめ，プログラムがみたすべき性能をまとめて仕様書ともいうべきものを作り，ブロック・チャートまたはフロー・チャートを経てコーディングに至るのだが，問題が比較的簡単でこれらの仕事を全部1人でやる場合は必ずしもこれらを全部行なう必要はない．しかし規模の大きなプログラムで，アナリスト，プログラマおよびコーダがそれぞれ別の人であり，さらにプログラマやコーダも何人かが1組になるような場合には多少の問題が生ずる．

広い意味での情報処理組織を，アナリストからオペレータに至るチームと電子計算機をいっしょにしたものと考えて，このシステム全体の能率，正確さを問題にするならば，そのシステムの構成分子である人間対人間および人間対電子計算機の間でどのように能率よく情報を伝えるかが大きなポイントになってくるといえるだろう．

ふつうアナリスト，プログラマ，コーダたちの間の連絡用には仕様書，ブロック・チャート，フロー・チャートを使い，これに基いて問題向き言語または計算機向き言語を使って実際のプログラムを書いている．つまり人間どうしで使う言語と人間が計算機に対して使う言語とがはっきり分かれているので，多くの場合アナリストが直接プログラムを読み書きすることができない．このために：

- ○プログラムができ上がるまでに時間がかかる．
- ○論理的な誤りの発見と訂正がやりにくい．
- ○完成したプログラムを別人が読んで望みの変更を加えることが困難である．
- ○プログラム・ドキュメンテーション（そのプログラムに関するすべての情報を統一したフォームでまとめ保存して，後日他人が見てもよくわかるようにしておくためのもの）を作るのに手間がかかる．

などの欠点があげられる．

もちろん近年発達した FORTRAN, ALGOL または COBOL などの問題向き言語によって上にあげた諸点は大きく改善された．言語プロセッサ自体も計算機の性能向上に伴って高度の発達をとげ，コンパイル時間が実用上役に立つ程度に向上し，各種の文法的誤りを親切に指摘するようになり，つくり出されるオブジェクト・プログラムもそう能率の悪いものではなくなってきている．そこで最近発表される計算機は，これらのシステムを標準ソフトウエアとしてもっている．したがって現在では，ほとんどすべてのプロダクション・プログラムが，このような言語で書かれるようになってきたし，さらに人間どうしの通信にも使われる場合がふえてきた．たとえば，Communications of the ACM に寄稿するアルゴリズムは，ALGOL 60 で書かれていなければならないし，「情報処理」のプログラムのページについても同様である．また事務計算用言語 COBOL の開発目的の一つは，人間どうしの通信にも役立つようなプログラミング言語をつくることにあった．

ところで，このような問題向き言語の間には一つの共通した特徴がある．つまり，これらを使ってプログラムを書く場合，各ステートメントの順序と，計算機が指定された手続を実行する時間的な順序とは大体対応している．もちろん例外としてはサブルーチンと主ルーチンとの関係や，ループなどがあり，さらに計算とバッファリングなど入出力関係の動作との関係をあげることができる．

しかし論理判断を複雑に組み合わせたようなタイプの問題になると，このような一次元的な記述方式が最適とは必ずしもいえないことがある．そこで，このタイプの過程をもっと見通しよく表現する方法を必要と

---

* Decision Tables, by Tetsutaro Yoshimura (Data Processing Section, Tokyo Shibaura Electric Co.)

** 東京芝浦電気（株）総務部機械計算課

するわけであるが，近年アメリカで話題となっている decision table がその一つにあげられる．

Decision table は判断とその結果行なうべき処理過程を表の形で（二次元的に）書いたものである．そしてフロー・チャートに代わるものとしてアナリストやプログラマの間の通信に使うばかりでなく，そのフォームや使う字母に制限を加えて計算機が読めるようにすれば，ほとんどそのまま入力言語として使えるようになっている．そのため人間と計算機からなる組織の大部分の通信にこの decision table を使うことができるので，前述のドキュメンテーションに伴う問題を改善するばかりでなく，アナリストと計算機との距離をちぢめるものとして注目されている．

Decision table の簡単な例を示そう．

例 1 ある生命保険の掛金率と最高契約額は，被保険者の年令，健康状態および性別によって決まるものとする．この計算手続を示した decision table は**第1図**のようなものになる．

		規則 1	規則 2	規則 3		規則 30
年 令	≧	25	25	25		65
年 令	＜	35	35	35		
健康状態	＝	優	優	良		可
性 別	＝	女	男	女		男
契約額100円当たりの掛金(円)	＝	1.57	1.72	1.95		5.9
最高契約額(万円)	＝	200	200	150		20

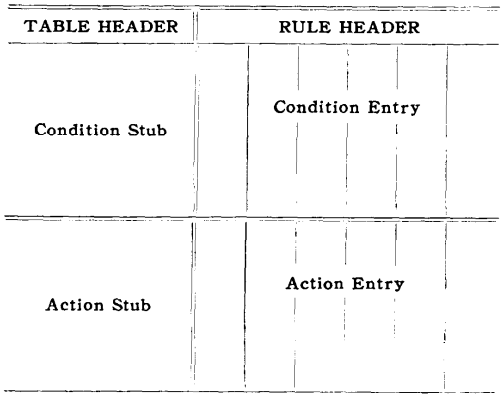
第 1 図

たとえば規則 1 は「25 才以上 35 才未満で健康状態が優の女性ならば，契約額 100 円当たり 1.57 円の掛金を毎月払って，最高 200 万円までの契約をすることができる」という意味をもっている．

Decision table の構造を一般的に書いたものが**第2図**である．

表全体は 2 本の二重線によって 4 分割されている．左上の部分（condition stub）に比較すべき変数の名と，比較に使う operator（＝，＞，≦ など）を記入するが，条件式全体を記入することもある．

右上の部分（condition entry）には比較の対象になる基準値を入れる．これらは必ずしも数値でなくてもよく，英数字，カナ文字などでもよい．つまり，COBOL の literal や ALGOL の string に対応するものが使える（ただし，この表の他の部分についてもそうであるが，decision table を直接の入力言語とする場合は字母などの関係から，それ相応の制限を加えられるのはいうまでもない）．



第 2 図

次に左下の部分（action stub）には，表の上半部で指定した論理関係が満たされた場合に何の値を決めるか，あるいは何をするかがはいる．たとえば従属変数の名前と等号がはいる．ここで使う等号は表の上半部にでてくる等号とは意味が違い，左辺の変数に右辺に示した値を代入するという意味であって，FORTRAN にでてくる等号と同じである．

TABLE HEADER にはその table のもつ名前などを入れる．実際の問題はとても 1 枚の表に書き切れないものが多い．つまり，いくつかの判断の結果，さらに別の判断過程を行なうために，それらを記述した別の表に移ることがあるので，行先が他の表から指定できるよう各 table は名前をもたなければならない．

例 2 信用販売を受付けるかどうかは，

1） 最高制限額を越えないか
2） 過去における支払成績の良否
3） 特別な受注許可命令の有無

の 3 条件によって決まるものとすると，この判断過程は次のような decision table で表現することができる．

例 1 の decision table と違って各 stub には条件式または命令が完全な形で記入されているが，各 entry には次のようなものしかはいらない．すなわち，condition entry には N, Y またはブランクしかはいっ

表の名称：信用販売承認手続	規則 1	規則 2	規則 3	規則 4
信用販売額≦最高制限額	Y	N	N	N
過去の支払成績＝良好		Y	N	N
特別受注命令＝有			Y	N
受註手続を行なえ	X	X	X	
注文差し戻し手続を行なえ				Y

第 3 図

ておらず，action entry には X またはブランクのいずれかしかない．Yは yes，Nは no の頭文字で，それぞれ対応する stub の条件式が成立するかどうかを示すものであり，もし，ブランクなら，そのテストを無視することを示す．また action entry における X は，対応する action stub で指定した命令を実行する (execute) ことを指定するもので，そこがブランクなら実行しないことになる．

例 1 のような decision table を extended entry と呼び，例 2 のようなものを limited entry といっている．どちらを使うかはその問題によって決まることであるが，いずれにせよ，ここにあげた例のように判断操作の多いタイプの問題ではフロー・チャートや narrative form の言語と同等，場合によってはそれ以上の見通しを得ることができる．

たとえば，例 1 の規則 1 に対応する COBOL の PROCEDURE は次のようになる．

```
IF (AGE NOT LESS THAN 25 AND LESS
THAN 35) AND HEALTH = 'GOOD' AND
FEMALE,
THEN COMPUTE RATE-PER − 100 = 1.57,
COMPUTE POLICY-LIMIT = 2000000.
```

このような statement をいくつか書いてやっと 1 枚の decision table に対応することになる．

また**第 4 図**のような limited entry decision table をフロー・チャートにすると，**第 5 図**のようになる．

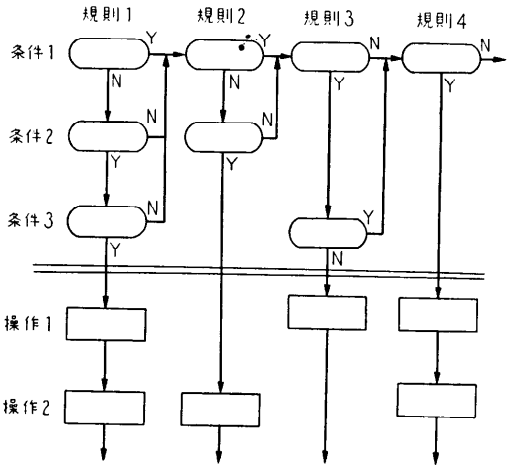
このように問題によっては decision table で書いたほうがずっとコンパクトに書けるものがある．

2. DETAB-X

これまでの説明では decision table の概要，特に人間どうしの通信の道具としての有用性に重点を置いていたため，具体的な notation とか使用可能な文字

	規則 1	規則 2	規則 3	規則 4
条件 1	N	N	Y	Y
条件 2	Y	Y		
条件 3	Y		N	
操作 1	X		X	X
操作 2	X	X		X

第 4 図



第 5 図

のセットなどにはいっさい触れなかった．しかし，decision table を直接の入力言語とするためには，そのフォーマットに種々の具体的な規約を設けて machine-readable な形にしなくてはならない．このような具体的仕様については各所で開発を進めており，すでに実用段階にはいっているものには General Electric 225 のための TABSOL とか，Rand Corporation が開発した FORTAB (FORTRAN を object language とする一種の preprocessor) などがある．ここで紹介する DETAB-X は米国国防省の支持によって活動している CODASYL (Conference on Data Systems Language) が作った decision table 言語の実験仕様であって，特定の計算機を対象としたものではないが，COBOL 61 への追加機能とも見ることができるものである．

1960 年および 1961 年に CODASYL で COBOL 言語 (Common Business Oriented Language) の仕様を発表してから，大多数の計算機メーカがこの仕様に基くコンパイラを書いたので，COBOL は現在最も

広く使われている事務計算用プログラミング言語になった.

さて DETAB-X とは Decision Table, Experimental の略であって, CODASYL Systems Group 内の Data Description and Transformation Logic Task Force が作ったものである. そして 1962 年 9 月ニューヨークで開かれた Decision Table シンポジウムで詳しく紹介された.

DETAB-X は COBOL との共通点を多く持っている. というよりは DETAB-X で書いたプログラムをできるだけ簡単な preprocessor で COBOL プログラムに変換できるようにその仕様を定めたというのが本当の所なのである. そうした理由は decision table が本当に入力言語として有用かどうかを, できるだけ早く試験するために, processor 開発に要する手間を最小にとどめるためであった.

DETAB-X による source program の構成は次に示すように COBOL のそれと正確に対応している.

IDENTIFICATION DIVISION
ENVIRONMENT DIVISION
DATA DIVISION
PROCEDURE DIVISION

このうちはじめの 2 DIVISION は, COBOL の対応する DIVISION の仕様をそのまま採用している.

まず IDENTIFICATION DIVISION にはそのプログラム全体に関する一般的な識別情報, すなわちプログラムの名称, プログラムの名前, 年月日などを記入する. この部分はプログラムの論理的な内容とは全く無関係で, 単に見出し程度の意味しかない.

次に ENVIRONMENT DIVISION は source program の中で金物に関係のあることをまとめて記述するためにある. たとえば source program を object program に直すための計算機と, object program を入れて実際に計算を行なうための計算機の両方について, その名称, 記憶装置の大きさ, 磁気テープ装置の数などを指定する. また PROCEDURE DIVISION では金物のスィッチやインジケータなどを直接扱うことができないので, それらに適当な (PROCEDURE DIVISION で使えるような) 名前をわりあてることもここで行なう. さらにファイルに関する情報のうち金物に直接関係のある項目はここに書く. たとえば, ファイルと特定の入出力装置の対応を指定したり, バッファー区域の大きさや特別な入出力処理方式の使用などを指定したりする. この DIVISION の内容が計算機によって大幅に変わることは記述の性質上やむをえないことである.

DATA DIVISION はファイルに関する指定する部分と, WORKING STORAGE および CONSTANT を指定する部分とに分かれている. ここではプログラムで処理 (すなわち PROCEDURE DIVISION で参照) するファイル, レコード, レコードを構成するグループ項目および基本項目, 作業用区域および定数について, その長さ, フォーマット, 初期値, 用途などを指定するが, 特定の計算機に対する依存の度合は, ENVIRONMENT DIVISION ほど大きくはない. しかし, 個々の計算機の特性をよくわきまえて, この DIVISION を注意深く書くかどうかで, オブジェクト・プログラムの速度が 10～100 倍違ってくることも決してめずらしいことではない. その意味では計算機の特性が implicit に反映しているといえよう. なお DETAB-X の DATA DIVISION に含まれる情報は COBOL のそれとほとんど同じであるが, 前者はある決まった形式の表によって表わし, COBOL におけるような narrative form は使わない.

PROCEDURE DIVISION はプログラマが計算機にやらせようとすることを書く部分である. COBOL では英文 (命令形) によって指定したが, DETAB-X ではここに decision table を採用した. この部分がいわゆるプログラムに対応するが, COBOL と同様この部分は必要な情報の一部にすぎず, ENVIRONMENT DIVISION や DATA DIVISION で指定した情報と合わせてはじめて完全なプログラムになる.

以下 DETAB-X の DATA DIVISION と PROCEDURE DIVISION の仕様のあらましを紹介しよう.

2・1 DATA DIVISION

前述のように PROCEDURE DIVISION で参照するデータの特性をここで明確に定義する. それらのうちには次のようなものが含まれている.

データの階層構造の指定: ファイルとそのファイルに属するレコードとの関係, 各レコードとその中のグループ項目との関係, グループ項目とそれを構成する下位レベルのグループ項目あるいは基本項目との関係などである.

基本項目に関する記述: 大きさ, 主用途 (内部計算のためか, 外部標示のためか), 符号の有無, 小数点位置, 編集されたときのフォーム (ゼロ抑制, ドル記号) などの指定.

第1表 DETAB-X DATA DIVISION FORM

PROCEDURE _____ SYSTEM _____ ANALYST _____ DATE _____ PAGE NO. _____

LINE NO.	LTYINPEE	LEVEL NOOF	NAME	ABBR	USE	CODE	DTYESPCE	PICTORIAL REFERENCE OR VALUE	REPETITIONS MIN	REPETITIONS MAX	SCYONDCE	RANGES OR REFERENCES FROM	RANGES OR REFERENCES TO

第2表 DETAB-X PROCEDURE DIVISION FORM

SYSTEM _____ SECTION _____ PREPARED BY _____ DATE _____ PAGE NO. _____

TABLE I.D.	BOW NO	LINE	CONT	TABLE NAME	FORM	PARAMETERS Cond Rows	PARAMETERS Action Rows	PARAMETERS Rules	NORMAL EXIT	ELSE DO	ELSE GO TO
					E	004	002	030	NEXT-TABLE	ALT-TABLE	ERR-TABLE
TAB	000	A		SAMPLE-TABLE							
				Rule Numbers							
TAB	000	Z				001			002		030
TAB	001			AGE GE		25			25		65
TAB	002			AGE LR		35			35		
TAB	003			HEALTH EQ		"EXCELLENT"			"EXCELLENT"		"POOR"
TAB	004			SECTIØN EQ		"EAST"			"WEST"		"WEST"
TAB	005			SET RATE-PER-1000 EQ		1.57			1.72		5.92
TAB	006			SET POLICY-LIMIT EQ		2000000			2000000		100000

WORKING STORAGE および CONSTANT についても上のような指定を行なう．

COBOL と違って DETAB-X では，これらを全部特別のフォームに書き込んで指定する（**第1表**参照）．

第1欄 LINE NO. 3桁の数字を記入する．PAGE NO とともにその行を一意に定める．

第2欄 LINE TYPE 空白，R, C または N がはいる．

空白 新しい行が始まることを意味する．

R COBOL の REDEFINE clause と同じ意味

［例］

LINE TYPE	LEVEL NO	NAME
	2	BLOCK-A……data-name-1
	3	ABLE
	3	BAKER
R	2	BLOCK-B……data-name-2
	3	ABLE
	3	BAKER

第6図

ここでは BLOCK-A というグループ項目と BLOCK-B というグループ項目が同一の記憶区域を使用（共有）することを示している．Data-name-1 と data-name-2 の間の関係は COBOL の規則に従う．

C（Continuation） この行が一つ前の行の続きであることを示す．

N（Note） この行は単なるコメントであることを示す．

D（Delete） この行を無視することを示す．

第3欄 LEVEL NUMBER データ項目の階層構造を示すために，各 data-name にレベル番号をつける．番号 01 をもつ項目はいちばん大きなグループで，たとえば record がこれに対応する．下位のグループになるほどレベル番号は大きくなる．番号は 01 から 49 までを使うことができ，その間は必ずしも連続していなくてもよい．COBOL の DATA DIVISION だとレベルの違う項目ごとに行の始まりを少しずつずらして書く，いわゆる indentation ができるが，この DETAB-X ではレベル番号を記入する欄が決まっているのでそれができない（ただし NAME の欄では行なうことができる）．

なお特別のレベル番号 77，88 および 66 があるが，これらの意味は COBOL におけるものと同様である．

第4欄 NAME Data-name を記入する．0～9，A～Z および —(hyphen) の自由な組み合わせから成る長さ 30 字までの name を使うことができるが，その中の少なくとも一字はアルファベットでなくてはならない．大別して次の3種類がある．

DATA NAME

CONDITIONAL VARIABLE

CONDITION NAME

このうち CONDITIONAL VARIABLE はあらかじめ決められたいくつかの値またはある範囲の値をもつことのできる変数に対してつけられた名前であり，CONDITION NAME はそれらの個々の値そのものを呼ぶためにつける名前である．

［例］ もし data-name TITLE が値1から7までをとりうるとし，値 1, 2, 3 に ANALYST という名前をつけ，値 5, 6 と値7にそれぞれ PROGRAMMER と CODER という名前をつけたとする．この場合 TITLE は CONDITIONAL VARIABLE で，CODER, PROGRAMMER および ANALYST はそれぞれ値7；値5, 6；および値1, 2, 3をもつ CONDITION NAME ということになる．

COBOL と同様 data-name を一意に定めるために QUALIFICATION（修飾）をすることができる．たとえば**第7図**のようなデータの構造があるときに，グループ項目 MASTER 中の基本項目 DAY を PROCEDURE DIVISION で呼ぶ場合は，単に DAY とするだけでは不十分であってそれより上位のレベルの data-name と結びつけて呼ばなければならない．すなわち，

LEVEL NO	NAME	LEVEL NO	NAME
01	MASTER	01	NEW-MASTER
02	CURRENT-DATE	02	CURRENT-DATE
03	MONTH	03	MONTH
03	DAY	03	DAY
03	YEAR	03	YEAR
02		02	
⋮		⋮	

第7図

DAY OF CURRENT-DATE IN MASTER または，DAY, CURRENT-DATE, MASTER

としなければならない．後者の書き方は DETAB-X でのみ許される notation であって，COBOL では使えない．

さらに data-name に添字（subscript）をつけて呼ぶこともできる．そのためには DATA DIVISION

で array を宣言しておく必要がある（第9欄参照）.

第5欄　ABBREVIATION　長い data-name を省略形で呼べるように，適当な略号を定義するための欄である．略号は長さ8字以内であることを除いて第4欄に関する制限がそのままあてはまる．この機能は COBOL にはないが，前述の DETAB-X から COBOL への preprocessor はこの欄を見て適当な RENAMES clause をつくり出すことができる.

第6欄　USE CODE　この欄に C (Computational) があれば対応する data-name の主用途は計算用であることを示し，Dなら標示用であることを示す．たとえば内部2進法の計算機ならCのあるものは2進形で記憶しているから，直接計算に使える．しかしD (Display) と指定されているものは BCD コードなどに変換した型式で記憶しているので標示用には便利だが，演算に使うときはそのつど2進法に変換しなければならない.

第7欄　DESCRIPTION TYPE　この欄より右の各欄がどんな情報をもっているかを1字の略号で指定する.

P (Pictoral)　このデータ項目の PICTURE を第8欄に書く.

F (Fixed)　このデータ項目のとる値が第8欄に記入されていることを示す．これは COBOL DATA DIVISION の VALUE clause に対応している.

X (Rename)　この場合レベル番号は 66 でなければならない．いくつかの基本項目をまとめて，それに別の data-name をつけるときに使う．その data-name は第4欄に記入する.

C (Copy)　同じ DATA DIVISION 内の別の所にある，グループ項目または基本項目に関する指定をそのままここに写すことを指定する．ただし写されるときにレベル番号はこの行のもっているレベル番号に合わせて相対的に調節される.

L (Library copy)　Cが同一プログラムからの copy であるのに対し，Lは DETAB-X システムに付属している標準ライブラリからの copy である.

LEVEL	NAME	DESCRIPTION TYPE	RANGE REFERENCE FROM	RANGE REFERENCE TO
03	TITLE	P		
88	ANALYST	K	1	3
88	PROGRAMMER	K	5	6
88	CODER	K	7	7
88	OPERATOR	K	8	9

第8図

K (Condition value)　CONDITION NAME のもつ値を右の各欄を使って指定することを示す．この行はレベル番号88をつけなければならない.

［例］（**第8図** 参照）

第8欄　PICTORIAL, VALUE, REFERENCE　基本項目の PICTURE, VALUE などを記入する.

PICTORIAL　COBOL の PICTURE と同じものを記入する．その詳細な規則は COBOL Reference manual を参照されたい.

VALUE　Data-name の初期値や，CONDITION NAME のもつ値を記入する.

［例］ Data-name のもつ値を指定する場合(DESC. TYPE F)

NAME	DESCRIPTION TYPE	VALUE
PI	F	3.14159
TEST	F	"COMPLETED"

第9図

［例］ Condition name のとる値を指定する場合 (DESC. TYPE K)

NAME	DESCRIPTION TYPE	VALUES
FM-TUNER	K	1041, 0141
PORTABLE-RADIO	K	4407, 0407
COLOR-TV	K	4907, 0907

第 10 図

REFERENCE　COPY の対象となる data-name で，DATA DIVISION または LIBRARY 中にすでに存在していることが必要である.

第 9, 10 欄　NUMBER OF REPETITIONS　Subscript をつけて参照する data-name であれば，その subscript がとりうる値の範囲をここで指定する．COBOL の DATA DIVISION 中の OCCURS clause に対応する.

第 11 欄　SYNCHRONIZED CODE　固定語長の計算機の場合にデータの packing を行なうかどうかを指定するためのもので，もし空白なら packing を行なう．Rなら packing は行なわず，データを語の右側によせて入れ，Lなら左側によせて入れる.

COBOL の SYNCHRONIZED clause に相当する．

第12, 13欄 RANGE (S) OR REFERENCES (FROM-TO)

第8欄の DESCRIPTION TYPE が P の場合には，この欄にその基本項目がとる値の最大値と最小値を記入する．またKの場合には condition-name がもつ値の範囲（最大値と最小値）を記入する．また DESCRIPTION TYPE がXなら RENAME の対象となる一連のデータ項目の最初のものと最後のものをそれぞれ記入する．

［例］

LEVEL NO	NAME	DESC TYPE	REFERENCES	
			FRON	TO
01	PAYROLL-RECORD			
02	EMPLOYEE-NO			
03	DEPT-NO			
03	PAYROLL-NO			
02	EMPLOYEE-DATA			
03	NAME-OF-EMPL			
03	ST-ADDRESS			
03	.....			
03	.....			
66	EMPLOYEE-INFO	X	PAYROLL-NO	ST-ADDRESS

第 11 図

2・2 PROCEDURE DIVISION

DETAB-X の PROCEDURE DIVISION は何枚かの decision table だけからできており，その形式は第2表のようである．この表の各項目について説明しよう．

FORM HEADER

system name, 作者，日付を入れる．

TABLE HEADER

各 table は2行の header line を持っている．

最初の行はその table 全体に関する情報を含み，

第2行は Rule Number を記入する．Identification Number を変えることによって1頁に複数個の table を書くこともできる．また逆に何頁かを使って1枚の decision table にすることもできる．この場合は rule, condition または action の数などが1頁に入り切れなくなったためで，後続する頁の ROW 000, LINE A の行の CONT 欄にCを記入する．TABLE HEADER 内の各欄は次のようなものが含まれる．

TABLE ID ある decision table に属する row はみな同一の TABLE ID をもたなければならない．

ROW NO. Table header の row no. は常に 000 である．

LINE Table header のときはAを記入する．

CONT ふつう空白だが，この頁が，ある decision table の第2頁以降ならCを記入する．

TABLE NAME ACTION STUB で参照するために，COBOL の paragraph name に相当するものと記入する．GO TO または DO verb で参照する．

FORM L, E またはMを記入し，それぞれこの table が limited entry か，extended entry か，あるいは mixed entry かを指定する．

Limited Entry なら condition entry は一欄の幅が3字分となり，各欄に記入できる文字は Y, N または空白（もしくは－）の3種類に限られる．

Extended entry の場合1欄の幅は 12 字である．

PARAMETERS Condition Row の数，action row の数および rule の総数を記入する．前述した decision table の一般形において，タテ，ヨコの二重線に相当するものがここで指定する情報である．

NORMAL EXIT Rule のうち一つでも GO TO によって行先を明示していないものがあると，そのような rule の最後はここに記入した table への GO TO 命令となる．

ELSE DO Table 中のどの rule も満足しないときは，ここで指定した table を closed subroutine として実行する．

ELSE GO TO ELSE DO から帰ってくると，ここで指定した table にコントロールを移す．

RULE HEADER

ROW 000
LINE Z
CONT 空白

RULE NUMBERS 3桁の数字から成る．テストの順序とこの数字の大小とは無関係である．

CONDITION STUB, AREA

ROW 3桁の数字（自然数の順序）
LINE 空白
CONT 空白

ACTION STUB, AREA

ほとんど上と同様の規則に従う．

実際の記入の仕方は**第2表**のとおりである．ACTION STUB に書く動詞は COBOL で使うものによく対応しており，その文体的な規則も大体同じである．すなわち，まず入出力関係では

ACCEPT, DISPLAY, OPEN CLOSE, READ, WRITE, ADVANCE

四則演算用には

ADD, SUB, MUL, DIV

実行順序制御用には

DO (COBOL の PERFORM に相当), GO TO, STOP

データの操作のためには

MOVE, SET (COBOL の COMPUTE に相当), EXAMINE

などが定義されている．

3. その他

このほか技術計算のために RAND CORPORATION が開発した FORTAB システムもある．これは FORTRAN プログラムを object code とする preprocessor であるから，DETAB-X 同様ふつうのコンパイラより pass が 1～2 回多くなる欠点がある．しかし，RAND CORP. での実験結果によれば，FORTAB で書いたほうが FORTRAN で書いたときより debug 回数が少なくてすみ，FORTAB が余分の処理時間を必要とするという欠点は完全に打ち消されているとのことである．

なお，decision table はアプリケーション・プログラムのみならず，モニタなどいくつかのシステム・プログラムの設計にも役立つと思われる．

最後に DETAB-X の項では，CODASYL のマニュアル〔2〕を基礎としたことを付記しておく．

参考文献

1) COBOL-61 Extended: US Gov. Printing Office 1962.
2) DETAB-X Specificatons: CODASYL 1962.
3) Proceeding of a Symposium on Decision Tables: Sept. 20, 1962.
4) 和訳 COBOL: 情報処理学会 1962.
5) Logic Structured Tables: Cantrell et al, CACM vol. 4 No. 6

（昭和39年7月18日受付）