論 文

デシジョンテーブルによるプログラムの自動ドキュメンテーション*

守 屋 慎 次** 平 松 啓 二**

Abstract

Many algorithms have been presented for converting decision tables into computer programs. In this paper, the program, called DTG (Decision Table Generator), for converting computer programs into decision tables is proposed and the conversion algorithm is presented. DTG can be used for documenting computer programs and for an aid of program debugging.

1. はじめに

プログラムの自動文書化の手法として自動フローチャーテングがある[1]. これはフローチャートが大変文書性に富むからに他ならない. 一方, フローチャートに劣らず文書として非常に優れた特徴を持つものにデシジョンテーブルがある. ところが, このデシジョンテーブルをプログラムの自動文書化に利用するという試み, 換言すれば, プログラムをデシジョンテーブルを含んだプログラムに変換し, 文書として利用したりデバッグの助けとするというような試みは, 少くも筆者の知る限りにはない***.

本論文の目的は次の二点である. 第一に, デシジョンテーブルをプログラムの自動文書化とデバッグの助けに用いることの提案. 第二に, プログラムをデシジョンテーブルを含んだプログラムへ変換する一つのアルゴリズムを述べることである.

本論文では, デシジョンテーブルとグラフ理論の基本的な用語を説明し, 次いで変換アルゴリズムを述べる. 最後に問題点・特長・今後の課題について議論する.

2. 基本的な用語

デシジョンテーブル (以後, テーブルと略記する) に関する用語は, 文献2),3) を参照されたい. ここではグラフに関する基本的な用語について簡単に述べておく.

有向連結グラフ G を $G=(B,E)$ と表わす. ここで B はグラフの頂点の集合 $B=\{b_1,\cdots,b_n\}$, また E はグラフの弧の集合 $E=\{(b_i,b_j),(b_k,b_l),\cdots\}$ である. 頂点 b_i から b_j に向う弧は順序対 (b_i,b_j) で表わされている. B の元を B の部分集合 (空集合 ϕ も含む) に写像する有向結合関数を $\Gamma_G{}^1$ と書く. すなわち, $\Gamma_G{}^1(b_i)=\{b_j|(b_i,b_j)\in E\}$ である. また, $\Gamma_G{}^{-1}$ なる有向結合関数を同様に, $\Gamma_G{}^{-1}(b_j)=\{b_i|(b_i,b_j)\in E\}$ と定める. $\Gamma_G{}^{-1}(b)$ もまた空であってよい。$G=(B,E)$ の部分グラフは, 有向連結グラフ $G'=(B',E')$ である. ここで, $B'\subset B, E'\subset E, G\cap G'=G', G\cup G'=G$. さらに G' の有向結合関数 $\Gamma_{G'}{}^1$ は, すべての $b_i'\in B'$ に対し, $\Gamma_{G'}{}^1(b_i')=\{b_j'|(b_i',b_j')\in E'\}$ である. 有向グラフ G の経路は, 順序づけられた頂点の列 $(b_1,\cdots,b_n)$ で表わされる部分グラフ P である. こ

---

* An Automatic Program Documentation by Decision Tables Converted from a Computer Program by Shinji MORIYA and Keiji HIRAMATSU (Department of Electrical Communication Engr., Tokyo Electrical Engineering College)

** 東京電機大学電気通信工学科

*** 本論文投稿中, 本論文とほぼ同主旨の論文「J. C. Cavouras *"On the Conversion of Programs to Decision Tables"*, CACM, Vol. 17 (1974), No. 8」が発表されたが, 本論文はこれとは独立である. 上記論文の変換の目的はプログラムのデバッグと最適化であり, 変換例 (BASIC 言語) や問題点・解決策と共に, 変換アルゴリズムを DT で記述している. 本論文との目的上の相違 (文書化と最適化) が主として次の違いを生んでいる. 本論文は文の位置や論理的関係などを重視して複雑多岐な部分だけを DT 化しているのに対しは, 上記論文それらにはほとんど無関心で比較的単調にすべての文を DT 化している. 一方, 上記論文では最適化 (あまり詳しくは述べられていない) を扱っているが, 本論文では全く言及していない.

こで，$b_{i+1}\in\Gamma_P{}^1(b_i),(b_i,b_{i+1})\in E$．$b_1=b_n$ なる経路はサイクルと呼ばれ，$b_1,\cdots,b_n$ がすべて異なる頂点の場合サイクルは単純であるといわれる．ある経路 $P=(b_1,\cdots,b_n)$ が存在し，かつ $b_1=p, b_n=q, n>1$ であるとき，q は p の子孫であるという．有向連結グラフ G における任意の頂点 p の子孫から成る集合を p の子孫集合と呼び，$\Gamma_G{}^*(p)$ と表わす．すなわち，$\Gamma_G{}^*(p)=\{q\mid q$ は p の子孫$\}$ である．$\Gamma_G{}^*(p)$ は空であってもよい．また任意の有限集合の元の個数を $|A|$ で表わす．

3. 変換アルゴリズム

ここでは，プログラムを，テーブルを含んだプログラムに変換するアルゴリズムについて述べる．アルゴリズムに対する基本方針として次の二つがある．

（**方針1**） 論理的に連続している条件群で，一つのテーブルの条件部を構成する．ただし例外がある．

（**方針2**） そのような条件群と論理的関係が強いと判断される実行文により，そのテーブルのアクション部を構成する．

変換アルゴリズムは三つの Phase から成る．説明の都合上，変換の対象言語として FORTRAN IV (JIS 水準 7000) を選んでいるが，アルゴリズムの大部分は他のコンパイラ言語にも適用可能である．以下の三つの Phase は，プログラム単位ごとに実行される．なお，原始プログラムに対し，後述の定義3の制限を設ける．

Phase 1. 原始プログラムのグラフへのモデル化

この Phase では，原始プログラムをブロック化しながらグラフ・モデルへ変換する．原始プログラム内の算術 IF 文と，真側に単純 GOTO 文以外の文が書かれた論理 IF 法は次のように変換される．すなわち，それと等価な論理を表わし，真側に単純 GOTO 文が置かれた形式の論理 IF 文の集まりに変換する．このようにして得られたプログラムを改めて原始プログラムと呼ぶ．原始プログラム内の先頭から i 番目 ($i=1,\cdots,z$) に書かれた文を s_i とすれば，原始プログラム Z は，$Z=s_1,s_2,\cdots s_z$ なる有限列で表現できる．ここに z は文の総数である．

〔**定義1**〕 原始プログラム Z 内の連続する部分列 $b=s_p,s_{p+1},\cdots,s_{p+m}$ を，文列という．ただし，s_p は次の α_1～α_8 のいずれかである．

α_1. 論理 IF 文 ($m=0$).

α_2. STOP 文，RETURN 文，END 文 ($m=0$).

α_3. 計算形 GOTO 文，割当形 GOTO 文 ($m=0$).

α_4. DO の端末文 ($m=0$).

α_5. 文番号が書かれた文 ($m\geqq1$)．ただし α_1～α_4，α_6～α_8 および FORMAT 文を除く．

α_6. DO 文 ($m\geqq1$).

α_7. プログラム単位の先頭の文 ($m\geqq1$).

α_8. 直前の文 (s_{p-1}) が論理 IF 文または DO の端末文である文 ($m\geqq1$).

α_1～α_4 の場合 $b=s_p$．α_5～α_8 の場合の s_{p+1}～s_{p+m} は，α_5～α_8 なる s_p に続き α_1～α_6 のいずれかが初めて出現する直前までの文である．なお，α_1 の文列を条件列，α_2～α_8 の場合の文列をアクション列と呼ぶ．

〔**定義2**〕 ブロック化された原始プログラム[4] Z' は，$Z'=b_1,b_2,\cdots,b_{z'}$ なる文列 b_i ($i=1,\cdots,z'$) から成る有限列である．ただし，$b_1,\cdots,b_{z'}$ の順序は原始プログラム Z の文の順序に対応するものとする．なお，Z' の添数集合を，$I_{Z'}=\{1,2,\cdots,z'\}$ によって表わす．

〔**定義3**〕 流れグラフ G とは，$G=(B,E)$ で表わされる有向連結グラフである．ここで，頂点の集合 B は，ブロック化された原始プログラム Z' を構成するすべての文列から成る集合，弧の集合 E は，Z' における文列の制御の流れを表わしている．また有向結合関数 $\Gamma_G{}^1,\Gamma_G{}^{-1}$ を **2.** と同様に定める．ただし，END 文，STOP 文，RETURN 文の各々から成る文列をそれぞれ b_E,b_S,b_R とすれば，$\Gamma_G{}^1(b_S)=\Gamma_G{}^1(b_R)=\{b_E\}$ であるとする．また，流れグラフ G は，アクション列だけから成るサイクルを部分グラフとして持つことはないものとする．なお，$\Gamma_G{}^{-1}(b)=\phi$ なる b は定義1の α_7 のみ，$\Gamma_G{}^1(b')=\phi$ なる b' は END 文のみとする．

〔例1〕 **Fig. 1**(次頁参照) の場合，$Z=s_1,\cdots,s_{52}$，ただし，s_1 は SUBROUTINE 文，s_2 は COMMON 文，s_{52} は END 文である．また $Z'=b_1,\cdots,b_{39}$．ここで，$b_1=s_1s_2s_3, b_2=s_4,\cdots,b_{39}=s_{52}$ である．

Phase 2. 頂点の集合 B の分割

変換された目的プログラム上の各テーブルは，可能な限り論理的なまとまりを持っていることが望ましい．したがって集合 B をできる限り論理的なまとまりを持つ部分集合に分割しなければならないが，それを行うための確固とした方法が存在しない。本論文では，以下に述べる六個の Stage で分割を行う．

Stage 1. DO ループの範囲を求める，

FORTRAN IV プログラム内における DO ループ

```
      SUBROUTINE CPCACP(II,IJ,IK,IE,V,IERR)
      COMMON IN(26),ID(3),IND(3),ICH(47),IB
      DATA IV,IF,IZ/1,0,0/
  100 IB=IB+1
      IF(IV.EQ.IB).AND.ICH(44))GOTO 20
      IF(IN(IB).EQ.ICH(15))GOTO 30
      IF(IN(IB).EQ.ICH(43))GOTO 20
      IF(IN(IB).EQ.ICH(37))GOTO 40
      IF(IN(IB).EQ.ICH(38))GOTO 50
      IF(IN(IB).EQ.ICH(45))GOTO 60
  200 IERR=1
      RETURN
   20 IF(I.NE.1)GOTO 200
      I=2
      GOTO 100
   30 IF(I.EQ.3)GOTO 200
      IF(I.EQ.1)GOTO 65
      GOTO 70
   40 IK=1
      GOTO 300
   65 IF(IND(1).EQ.0)GOTO 200
   70 I=3
      GOTO 100
   50 IK=-1
  300 IF(I.EQ.1)GOTO 80
      IF(I.EQ.3)GOTO 90
      GOTO 200
   80 IF(IK.NE.0)GOTO 200
      IF(IND(1).NE.0)GOTO 200
      IR=IK
      GOTO 100
   90 IF(IE.NE.0)GOTO 200
      IF(IND(3).NE.0)GOTO 200
      IE=IK
      GOTO 100
   60 IJ=1
  400 IF(I.EQ.1)GOTO 15
      IF(IND(1).NE.0)GOTO 55
      IF(IND(2).NE.0)GOTO 25
      GOTO 200
   15 IF(IR.EQ.0)GOTO 35
      II=IV(1)*IR
      IK=1
   35 RETURN
   23 IF(IR.EQ.0)GOTO 45
   55 IF(IE.EQ.0)GOTO 35
   45 V=10.**ND(2)
      N=ID(3)*IE
      IERR=0
      IK=2
      GOTO 35
      END
```

Fig. 1 A blocked FORTRAN program.

は，強く連結された論理的なかたまりと考えられる．

〔**定義 4**〕 ブロック化された原始プログラムを $Z'=b_1,\cdots,b_s,\cdots,b_e,\cdots,b_{s'}$ と置く．このとき，流れグラフ G の経路 $P=(b_s,\cdots,b_t,\cdots,b_e)$ $(s\leq i\leq e)$ 上の文列から成る集合 $D=\{b_s,\cdots,b_t,\cdots,b_e\}$ を G の DO ループと呼ぶ．ただし，b_s は任意の DO 文から成る文列，b_e は b_s に対応する端末文である．

〔**定義 5**〕 流れグラフ G の DO ループ D は，G 内に $D\cap D'=D$ なる他の DO ループ D' が存在しないとき極大であるという．

〔**定義 6**〕 流れグラフ G の DO ループの入れ子とは，次のような半順序集合 $\mathcal{D}_i=\{D_{i1},\cdots,D_{in}\}$ である．また，互に素な DO ループの入れ子 $\mathcal{D}_i(i=1,\cdots,m)$ を元とする集合を $\mathcal{D}=\{\mathcal{D}_1,\cdots,\mathcal{D}_m\}$ と表わす．ここに，$D_{ij}(j=1,\cdots,n)$ は DO ループで，任意の $j,k(j<k)$ に対し，$D_{ij}\cap D_{ik}=\phi$ または $D_{ij}\cap D_{ik}=D_{ij}$ のいずれかである．また，D_{in} は極大で，すべての j に対し $D_{ij}\cap D_{in}=D_{ij}$ である．

DO ループと入れ子を実際に求めるアルゴリズムについては，文献 4) を参照されたい．

〔アルゴリズム A〕 DO ループの範囲 R_{ij}.

DO ループ D_{ij} $(i=1,\cdots,|\mathcal{D}|)$ に対応させて集合 R_{ij} を次のように定める．R_{ij} を DO ループの範囲と呼ぶ．

(1) すべての i について(2)(3)をくり返す．得られた結果を $\boldsymbol{R}_i=\{R_{i1},\cdots,R_{in}\}$, $\boldsymbol{R}=\{\boldsymbol{R}_1,\cdots,\boldsymbol{R}_m\}$ と表わす．ただし，$n=|\mathcal{D}_i|$, $m=|\mathcal{D}|$.

(2) $j=1$ のとき，

(2.1) すべての $b\in D_{ij}$ を R_{ij} の元とする．

(2.2) 任意の $b\in R_{ij}$ に対し，$\Gamma_G^{-1}(b')\subseteq R_{ij}$ なるすべての $b'\in\Gamma_G^{1}(b)$ を R_{ij} の元に加える．

(2.3) すべての $b\in R_{ij}$ について(2.2)をくり返す．

(3) (3.1) $j\leftarrow j+1$, $j>n$ のとき(1)へ戻る．それ以外のとき，(2.1)～(2.3)により R_{ij} を求める．

(3.2) すべての $k(<j)$ について $D_{ik}\cap D_{ij}=\phi$ のとき，(3.1)へ戻る．$D_{ik}\cap D_{ij}=D_{ik}$ なる k $(<j)$ が存在するとき，そのようなすべての k に対する $R_{ij}-R_{ik}$ を改めて R_{ij} とする．

〔記法〕 $I=\{1,2,\cdots,|\mathcal{D}|\}$ と表わす．また，すべての $i\in I$ に対し，$J_i=\{1,2,\cdots,|\mathcal{D}_i|\}$ と表現する．

〔補題 1〕 すべての $i\in I$ に対し次の関係が成立する．すなわち，任意の $j,k\in J_i$ $(j\neq k)$ について，$R_{ij}\cap R_{ik}=\phi$. (証明略)．

〔補題 2〕 任意の $i,j\in I$ $(i\neq j)$ について，$\boldsymbol{R}_i\cap\boldsymbol{R}_j=\phi$. (証明略)．

〔**定義 7**〕 どの DO ループにも属さないすべての頂点 $b\in B$ を元とする集合を，$R_{01}=B-\bigcup_{i\in I}\bigcup_{j\in J_i}R_{ij}$ と置き，以後 R_{01} をも DO ループの範囲と呼ぶ．また，$\boldsymbol{R}_0=\{R_{01}\}$, $\boldsymbol{R}'=\boldsymbol{R}\cup\{\boldsymbol{R}_0\}$ とする．

〔記法〕 $I_0=I\cup\{0\}=\{0,1,2,\cdots,|\mathcal{D}|\}$, $J_0=\{1\}$ と表わす．

集合 R_{ij} $(i\in I_0, j\in J_i)$ は，論理的なまとまりを持った，頂点の集合 B の一つの分割である．したがって，原始プログラムをテーブル化する際，テーブル内の文列同志が異なる DO ループの範囲にまたがらないように変換すべきと考えられる．

Stage 2. テーブルの先頭の条件列を求める.

この Stage では，テーブル化する際に条件スタブの一番上の行に書かれるべき条件列を求める．この条件列を先頭の条件列と呼ぶ。

〔**定義 8**〕 $C=\{b\mid b\in B$ は条件列$\}$, $A=\{b\mid b\in B$ はアクション列$\}$ と定める．

$b\in C$ が先頭の条件列であるためには何が要求されるかを考えてみる．もし，$b'\in\Gamma_G^{-1}(b)\cap C$ なる b' が

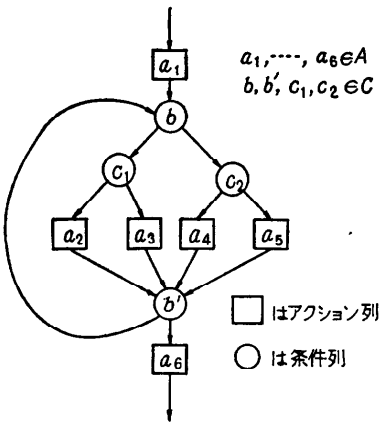


Fig. 2 A subgraph with simple cycles

存在すれば，b より b' の方が，先頭の条件列としての資格を備えている．換言すれば，$\Gamma_G^{-1}(b) \subseteq A$ なる b は先頭の条件列である．しかし，上記のような b' が存在しても b を先頭の条件列にすべき場合がある．場合の第一は，そのような b' を含む DO ループの範囲が b を含むそれと，すべての b' について異なるとき．場合の第二は，**Fig. 2** に例示するように，b と b' があるサイクル上の頂点になっているとき．場合の第三は $\Gamma_G^{-1}(b) \cap A \neq \phi$ のときである．場合はこの三つだけではない．残された「場合」は後の過程(Stage 4) で検出される．

〔アルゴリズムB〕 先頭の条件列 $h_{ij}{}^k$.

$p, q \in I_{Z^+}$ とする．

(1) すべての $i \in I_0, j \in J_i$ について(2)をくり返す．

(2) 任意の $b_p \in R_{ij} \cap C$ について次のような集合 X, Y を求める。$X = \{b_q \mid b_q \in C \cap \Gamma_G^{-1}(b_p),\ p < q,\ b_p$ と b_q は流れグラフ G におけるある単純サイクル上の二頂点$\}$. $Y = \{b \mid b \in C \cap \Gamma_G^{-1}(b_p),\ b \notin R_{ij}\}$.

(2.1) このとき，$\Gamma_G^{-1}(b_p) - X - Y \subseteq A$, または $\Gamma_G^{-1}(b_p) \cap R_{ij} \cap A \neq \phi$ かつ $\Gamma_G^{-1}(b_p) \neq \phi$, または $\Gamma_G^{-1}(b_p) = \phi$ のとき，b_p を集合 H_{ij} の元とする．

(2.2) すべての $b_p \in R_{ij} \cap C$ について(2.1), (2.2)をくり返す．得られた集合 H_{ij} を，$H_{ij} = \{h_{ij}{}^1, h_{ij}{}^2, \cdots\}$ と表わす．

アルゴリズムBの(2)における集合 X を求めるアルゴリズムは省略する．単純サイクルを求めるアルゴリズムについては文献5) を参照されたい．

〔記法〕 すべての $i \in I_0, j \in J_i$ に対し，$K_{ij} = \{1, \cdots, |H_{ij}|\}$ と表わす．H_{ij} は先頭の条件列の集合．

Stage 3. テーブルの最大範囲を求める．

$h \in H_{ij}$ を先頭の条件列とするテーブルの，条件部とアクション部に含める可能性のあるすべての文列から成る集合をテーブル h の最大範囲と呼び，$T_1[h]$ と表わすことにする．また $C_1[h] = T_1[h] \cap C$ を条件列の最大範囲，$A_1[h] = T_1[h] \cap A$ をアクション列の最大範囲と呼ぶ．本章の最初に述べた(方針1)(方針2)により，この Stage において各 $h \in H_{ij}$ の $C_1[h]$ と $A_1[h]$ を求めることにする．

〔アルゴリズムC〕 条件列の最大範囲 $C_1[h_{ij}{}^k]$,

(1) すべての $i \in I_0, j \in J_i, k \in K_{ij}$ について(2), (3), (4)をくり返す．

(2) 先頭の条件列 $h_{ij}{}^k \in H_{ij}$ を $C_1[h_{ij}{}^k]$ の元とする．

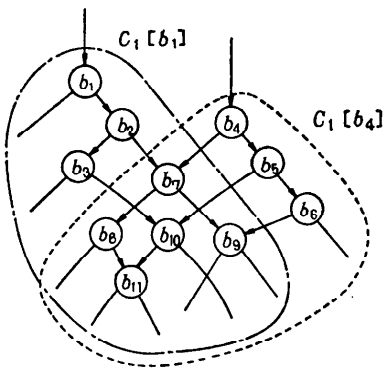
(3) 任意の $b \in C_1[h_{ij}{}^k]$ に対して，$b' \in \Gamma_G^{1}(b) \cap R_{ij} \cap C$ かつ $b' \notin C_1[h_{ij}{}^k] \cap H_{ij}$ なる b' が存在するとき，そのようなすべての b' を $C_1[h_{ij}{}^k]$ の元として，(4)へ行く．存在しないときはそのまま(4)へ．

(4) すべての $b \in C_1[h_{ij}{}^k]$ について(3)をくり返す．

[例2] **Fig. 3** で b_1, b_4 が先頭の条件列の場合，$C_1[b_1]$ と $C_1[b_4]$ は，図上に破線と一点鎖線で囲む範囲である．

〔補題3〕 (1) $C = \bigcup_{i \in I_0} \bigcup_{j \in J_i} \bigcup_{k \in K_{ij}} C_1[h_{ij}{}^k]$.

(2) 任意の $i \in I_0, j \in J_i$ に対し，$C' = C_1[h_{ij}{}^k] \cap C_1$



b_1, b_4 が先頭の条件列の場合
$b_1, \cdots, b_{11} \in C$

Fig. 3 An example of conditional statements' ranges with intersection.

$[h_{ij}']\neq\phi$なる$k,l\in K_{ij}(k\neq l)$が存在し得る.(証明略)

〔アルゴリズムD〕 アクション列の最大範囲 $A_1\cdot[h_{ij}^k]$.

(1) すべての$i\in I_0, j\in J_i, k\in K_{ij}$について(2)～(4)をくり返す。

(2) 定義1のα_4から成る集合を$\{\alpha_4\}$と表わすことにする.このとき,任意の$b\in C_1[h_{ij}^k]$に対して,$b'\in\Gamma_G^1(b)\cap R_{ij}\cap(A-\{\alpha_4\})\neq\phi$なる$b'$が存在するとき,そのようなすべての$b$に対するすべての$b'$を集合$A_1[h_{ij}^k]$の元として(3)へ行く.存在しないとき,$A_1[h_{ij}^k]$は空のまま(3)へ行く.

(3) 定義1におけるすべてのα_2,α_3から成る集合を$\{\alpha_2,\alpha_3\}$と表わす.また,原始プログラムZ'の添数集合$I_{Z'}$の二元をp,qと表わし,$b_p=h_{ij}^k\in H_{ij}$だとする.このとき,(*)のような文列bが存在するとき,b_qを$A_1[h_{ij}^k]$の元として(4)へ行く.存在しないときはそのまま(4)へ.

(*) $b\in A_1[b_p]\cap(A-\{\alpha_2,\alpha_3\})$かつ$b_q\in\Gamma_G^1\cdot(b)\cap R_{ij}\cap(A-\{\alpha_4\})\neq\phi$かつ$q>p$.

(4) 上記(*)を満足するすべてのbについて(3)をくり返す.

Stage 4. 条件列の範囲を求める.

補題3が示すように互に素でない条件列の最大範囲が存在する.補題3の記法を用いて,共通部分C'の処置について考えてみる.デシジョンテーブルでは条件部への飛込み[2]は許されないので,(1)$C_1[h_{ij}^k]$, $C_1[h_{ij}']$の両者にC'を属させ,二つのテーブルに変換するか,(2)$C_1[h_{ij}^k]-C'$, $C_1[h_{ij}']-C'$から構成される二つのテーブルと,C'から新たに造られるいくつか(例2参照)のテーブルに分ける,という二つの方法が考えられる.(1)の場合を採用すると,条件列のみならずその条件列に続くアクション列までをも重複して出力することになり,冗長で好ましくない.そこで(2)を採用する.

次のアルゴリズムでは,条件列の最大範囲から共通部分C'を削除し,アクション列の最大範囲からC'の子孫のアクション列を取り去る.次いで,C'内で新たに先頭の条件列を拾い出す.

アルゴリズムEの実行に先立ち,すべての$i\in I_0$, $j\in J_i$に対し$K_{ij}''=K_{ij}$と置く.

〔アルゴリズムE〕 共通部分の削除,先頭の条件列の追加。

(1) すべての$i\in I_0, j\in J_i$について(2),(3)をくり返す.ただし,二つの集合K_{ij}'とH_{ij}'を$K_{ij}'=H_{ij}'=\phi$と置く.

(2) すべての$k\in K_{ij}''$について$C_2[h_{ij}^k]=C_1\cdot[h_{ij}^k]$とする.

(3) すべての$k,l\in K_{ij}''$ $(k\neq l)$について(4)をくり返す.(4)をくり返して新たに得られた先頭の条件列の集合H_{ij}'と添数集合K_{ij}'を,$H_{ij}'=\{h_{ij}^{m+1}, h_{ij}^{m+2},\cdots\}$, $m=|H_{ij}|$, $K_{ij}'=\{m+1,m+2,\cdots,m+|H_{ij}'|\}$と表わす.また,$H_{ij}\cup H_{ij}'$を改めて$H_{ij}$, $K_{ij}\cup K_{ij}'$を改めてK_{ij}と表現する.

(4) $C'=C_1[h_{ij}^k]\cap C_1[h_{ij}']$と置く.このとき,$C'=\phi$ならば(3)へ戻る.$C'\neq\phi$のときは,$C_2[h_{ij}^k]-C'$を改めて$C_2[h_{ij}^k]$, $C_2[h_{ij}']-C'$を改めて$C_2[h_{ij}']$と表現する.また,$A'=\{b\mid b\in A\cap\Gamma_G^*(b'), b'\in C'\}$と置くとき,$A_2\cdot[h_{ij}^k]=A_1[h_{ij}^k]-A'$, $A_2[h_{ij}']=A_1[h_{ij}']-A'$と表現する.次いで,$C'$内で次の条件を満たす条件列を,新たに先頭の条件列とする.すなわち,$b'\in C'$かつ$\Gamma_G^{-1}(b')\cap(C_2[h_{ij}^k]\cup C_2[h_{ij}'])\neq\phi$かつ$b'\notin H_{ij}'$なる$b'$が存在するとき,$b'$を集合$H_{ij}'$の元とする.

[例3] Fig. 3の共通部分$C'=C_1[b_1]\cap C_1[b_4]$からアルゴリズムEによって新たに得られた先頭の条件列は,$H_{ij}'=\{b_7,b_9,b_{10}\}$である.

アルゴリズムEで新たに得られた先頭の条件列$h_{ij}^k\in H_{ij}'$ $(i\in I_0, j\in J_i, k\in K_{ij}')$に対し,再び条件列の最大範囲を求めなければならない。求められた新しい最大範囲の間には,さらに共通部分が存在するかも知れない.この過程は,共通部分がなくなるまで続けられる.

〔アルゴリズムF〕 共通部分の処理.

(1) アルゴリズムC,DにおけるK_{ij}とH_{ij}を,アルゴリズムEで得られたそれぞれK_{ij}'とH_{ij}'で置換えて,アルゴリズムCとDを実行する.新たに得られた条件列とアクション列の最大範囲を,それぞれ$C_1[h_{ij}^k]$, $A_1[h_{ij}^k]$ $(i\in I_0, j\in J_i, k\in K_{ij}', h_{ij}^k\in H_{ij}')$と表わす.

(2) アルゴリズムEにおけるK_{ij}''を,(1)で用いたK_{ij}'で置換えてアルゴリズムEを実行する。

(3) 次の条件を満たすまで(1)(2)をくり返す。すなわち,(2)でアルゴリズムEを実行する

際，すべての $k,l\in K_{ij}''$ $(k\neq l)$ について $C'=\phi$ となるまで．

アルゴリズムFを終了すると，H_{ij} は追加されたすべての先頭の条件列を含み，添数集合 K_{ij} の元は H_{ij} の元と1対1に対応している．さらに次の性質がある．

〔性質1〕 $i\in I_0, j\in J_i, k\in K_{ij}$ とする．このとき，$C=\bigcup_i\bigcup_j\bigcup_k C_2[h_{ij}^k]$ であり，しかも，任意の $k,l\in K_{ij}$ に対し，$C_2[h_{ij}^k]\cap C_2[h_{ij}^l]=\phi$ である．（証明略）

〔補題4〕 任意の $i\in I_0, j\in J_i$ に対し，$A'=A_2[h_{ij}^k]\cap A_2[h_{ij}^l]\neq\phi$ なる $k,l\in K_{ij}$ $(k\neq l)$ が存在し得る．（証明略）

Stage 5. アクション列の範囲を求める．

補題4が示すように，互に素でないアクション列の最大範囲が存在し，共通部分をどのように扱うかが問題となる．また，互に素であっても，最大範囲をそのままテーブルのアクション部とすることにも問題がある．便宜上，前者を問題1，後者を問題2と呼ぶことにする．この Stage では，両問題に一つの解を与え，生成すべき各テーブルのアクション部の構成要素（これをアクション列の範囲と呼ぶ）を定める．

まず問題2について考察してみよう．**Fig. 4**(a)のアクション列の最大範囲 $A_1[b_1]$ $(=A_2[b_1])$ をそのままアクション部としてテーブルを構成すると，Fig. 4 (b)が得られよう．b_8 のアクションエントリには，ルール R_1〜R_4 のすべてに×印が書かれている．すなわち，b_8 の実行は，条件列 b_1, b_2, b_3 とは無関係である．したがって，文書性の見地からも，b_8 はテーブルに含めない方が自然である．あるアクション列が，この b_8 のようなアクション列であるか否かは，そのアクション列のアクションエントリに書かれるはずの×印の数で判定できる．この数をアクション列のルール数と呼ぶ．同様に，条件列のルール数とは，その条件列の条件エントリに書かれるはずの文字YおよびNの数であると定める。アクション列と条件列のルール数を厳密に定義することはもちろん可能であるが，紙数の都合上，ここでは記法と例を上げるに止める．

〔記法〕 任意の $i\in I_0, j\in J_i, k\in K_{ij}$ における先頭の条件列を h_{ij}^k とする．このとき，文列 $b\in A_2[h_{ij}^k]\cup C_2[h_{ij}^k]$ のルール数を $N[b,h_{ij}^k]$ と表わす．

〔例7〕 Fig. 4 (a) の場合，先頭の条件列は b_1 である．このとき，$N[b_1,b_1]=4$, $N[b_2,b_1]=N[b_3,b_1]=2$, $N[b_4,b_1]=N[b_5,b_1]=N[b_6,b_1]=1$, $N[b_7,b_1]=3$, $N[b_8,b_1]=4$.

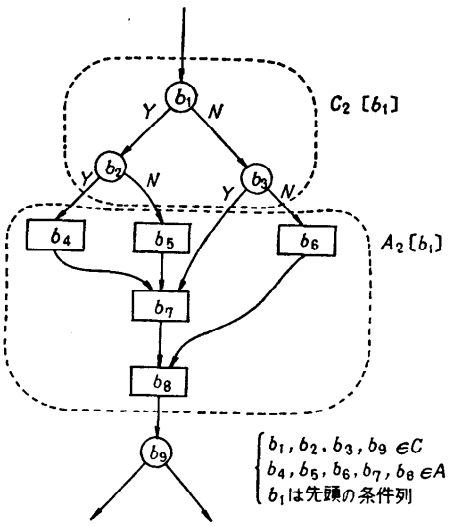


Fig. 4 (a) Typical example of control flow graph

	R_1	R_2	R_3	R_4
b_1	Y	Y	N	N
b_2	Y	N	–	–
b_3	–	–	Y	N
b_4	×	–	–	–
b_5	–	×	–	–
b_6	–	–	–	×
b_7	×	×	×	–
b_8	×	×	×	×

Fig. 4 (b) A decision table for Fig. 4 (a).

Fig. 4 の b_8 のように，アクションエントリがすべて×印で埋まるアクション列のルール数は，先頭の条件列のルール数に等しいことは明らかである．

〔**定義9**〕 任意の $i\in I_0, j\in J_i, k\in K_{ij}$ において，アクション列 b が，$N[b,h_{ij}^k]=N[h_{ij}^k,h_{ij}^k]$ となる $A_3[h_{ij}^k]$ 内の最初（Γ_G で定められる順序の上で）の元であるとき，b を収束点と呼ぶ．ただし，h_{ij}^k は先頭の条件列である．

〔アルゴリズムF〕 $A_2[h_{ij}^k]$ から収束点とその子孫を除く．

（1） すべての $i\in I_0, j\in J_i, k\in K_{ij}$ について(2)をくり返す．

（2） $A_2[h_{ij}^k]$ $(h_{ij}^k\in H_{ij})$ が収束点 b を持つとき，$A_3[h_{ij}^k]=A_2[h_{ij}^k]-\{b\}-(\Gamma_G^*(b)\cap A_2[h_{ij}^k])$．持たないとき，$A_3[h_{ij}^k]=A_2[h_{ij}^k]$．

次に，この Stage の最初に述べた「問題1」について考える．補題4の記法を用いて，共通部分 A' の処置

を考えてみる．デシジョンテーブルではアクション部への飛込み[2]も許されないので，$A_3[h_{ij}{}^k]$ と $A_3[h_{ij}{}^l]$ の(1)両者に A' を属させる，(2)いずれか一方に属させる，(3)どちらにも属させない，の三種類が考えられる．(2)の場合，一つのアクション列が一方ではテーブル内に，他方ではテーブル外に出力されるといった不均衡と冗長性があって好ましくない．(1)も冗長である．そこで(3)を採用する．

〔アルゴリズムG〕 アクション列の範囲 $A_4[h_{ij}{}^k]$.

(1) すべての $i\in I_0, j\in J_i$ について(2)をくり返す．

(2) すべての $k,l\in K_{ij}(k\neq l)$ について，$A'=A_3[h_{ij}{}^k]\cap A_3[h_{ij}{}^l]$ と置くとき，$A_4[h_{ij}{}^k]=A_3[h_{ij}{}^k]-A'$，$A_4[h_{ij}{}^l]=A_3[h_{ij}{}^l]-A'$ と定める．

Stage 6. テーブルの範囲を求める．

〔**定義 10**〕 すべての $i\in I_0, j\in J_i, k\in K_{ij}$ に対し，テーブルの範囲 $T_{ij}{}^k$ を次のように定める。すなわち，$T_{ij}{}^k=C_2[h_{ij}{}^k]\cup A_4[h_{ij}{}^k]$. ただし，$T_{01}{}^1=A-\bigcup_i\bigcup_j\bigcup_k A_4[h_{ij}{}^k]$.

〔性質2〕 $i\in I_0, j\in J_i, k\in K_{ij}$ とする．このとき，$B=A\cup C=\bigcup_i\bigcup_j\bigcup_k T_{ij}{}^k$ であり，しかも任意の $k,l\in K_{ij}(k\neq l)$ に対し，$T_{ij}{}^k\cap T_{ij}{}^l=\phi$ である．(証明略)

Phase 3. 目的プログラムの生成

この Phase で，テーブルを含んだ目的プログラムを生成する．本論文では，生成の際の主要な注意点を列挙するに止める．(1)目的プログラム中でテーブルに含めた文列と含めない文列との論理的な連結を行う．(2)各テーブルまたは各文列を出力する場所の決定（可能な限り $Z'=b_1,\cdots,bz'$ の順に出力する）．(3) $T_{01}{}^1$ に含まれる文列は原始プログラム上の文列をそのまま印刷．$T_{01}{}^1$ を除く $T_{ij}{}^k$ のうち，$|C_2[h_{ij}{}^k]|=1$ のものは，テーブル化するより IF 文のままの方が見やすいであろう．$|C_2[h_{ij}{}^k]|\geqq 2$ の $T_{ij}{}^k$ はテーブルに変換して出力する．

4. アルゴリズムの実装

およそ以上の方法により，Fig. 1 のプログラムは **Fig. 5** のように出力される．この変換プログラムを DTG/FORTRAN (DTG は Decision Table Generator の略) と呼んでいる．DTG 自身も FORTRAN で書かれている（宣言文を除いて約 1100 文，約 13 k バイト）．入力のプログラム単位中の文列数を b，ラベル数を l とし，また，生成されるテーブル数を t，1テーブル当りの条件行とアクション行とルール(列)の数をそれぞれ c,a,r で表わせば，DTG のデータ構造として次が必要になる．文の種類，文の真偽方向のポインタ，文がどのテーブルに属すかなどの，長さ b の1次元配列が 13 個．1テーブルのエントリ部の記憶のために，$(c+a)r$ の2次元配列が1個．それに作業用のスタッフ s，入出力エリア io 等を含め，最低 $m=13b+(c+a)(r+1)+t+2l+s+io$ の主記憶を要し，それに原始プログラムの記憶にシーケンシャルファイルを必要とする．$b=300$, $l=100$, $t=30$, $c=20$, $a=20$, $r=20$, $s=100$, $io=160$ とすれば $m=5230$．各2バイトを要する場合 10,460 バイト．プログラムも含めて所要主記憶は約 23.5 k バイトである．変換に要す CPU 時間は，Fig. 1 から5への場合で約 12 秒 (FACOM 230-48) である．

本論文では原始プログラムとして FORTRAN を選

```
      SUBROUTINE CHECKCH(I1,IJ,IK,N,Y,IERR)
      COMMON IN(44),IV(3),ND(3),ICH(47),IB
      DATA IN/15*0,29*0/
  100 IB=IB+1
      ------------------------------------------
      IN(13).EQ.ICH(44)   N N N N N N N N N N Y Y
      IN(14).EQ.ICH(15)   N N N N N N Y Y Y Y . .
      IN(15).EQ.ICH(43)   N N N N Y Y . . . . . .
      IN(17).EQ.ICH(37)   N N N Y . . . . . . . .
      IN(18).EQ.ICH(38)   N N Y . . . . . . . . .
      IN(19).EQ.ICH(45)   N Y . . . . . . . . . .
      I.NE.1              . . . . N Y . . . . N Y
      I.EQ.3              . . . . . . N N N Y . .
      I.EQ.1              . . . . . . N Y Y . . .
      ND(1).EQ.0          . . . . . . . N Y . . .
      ------------------------------------------
      IJ=1                . X . . . . . . . . . .
      GOTO 400            . X . . . . . . . . . .
      IK=-1               . . X . . . . . . . . .
      IK=1                . . . X . . . . . . . .
      GOTO 300            . . X X . . . . . . . .
      I=2                 . . . . X . . . . . X .
      I=3                 . . . . . . . . . . . .
      GOTO 100            . . . . X . X X . . X .
      ------------------------------------------
  200 IERR=1
      RETURN
      ---------------------------
  300 I.EQ.1          N N N N Y Y Y
      I.EQ.3          N Y Y Y . . .
      IE.NE.0         . N N Y . . .
      ND(3).NE.0      . N Y . . . .
      IK.NE.0         . . . . N N Y
      ND(1).NE.0      . . . . N Y .
      ---------------------------
      IE=IK           . X . . . . .
      IR=IK           . . . . X . .
      GOTO 100        . X . . . . .
      GOTO 200        X . X X . X X
      ---------------------------
  400 I.EQ.1          N N N N N Y Y
      ND(1).NE.0      N N N N Y Y .
      ND(2).NE.0      N Y Y Y . . .
      IR.EQ.0         . N N Y . . .
      IE.EQ.0         . N Y . N Y .
      IR.EQ.0         . . . . . . Y
      ------------------------------
      GOTO 200        X . . . . . .
      Y=10.**ND(2)    . X . X X . . .
      N=ID(3)*IF      . X . X X . . .
      IERR=0          . X . X X . . .
      IK=2            . X . X X . . .
      II=ID(1)*IR     . . . . . . X .
      IK=1            . . . . . . . X
      RETURN          . X X X X X X X
      ------------------------------
      END
```

Fig. 5 An object program of Decision Table Generator. The source program is shown in Fig. 1.

んでアルゴリズムを述べた．他のコンパイラ言語に対してほとんどそのまま適用可能な部分は，Phase 2 の Stage 2～6 と，Phase 3 の前半（データ構造上でテーブルを構成する）である．Phase 1 と Phase 3 の後半（目的プログラムの出力）が言語に依存するのは当然である．Phase 2 の Stage 1 は，DO ループとその拡張範囲を求める部分であるが，例えば COBOL の場合には，PERFORM 文による実行範囲を求めればよい．

5. むすび

デシジョンテーブルをプログラムの自動ドキュメンテーションとデバッグの助けに利用することを提案しその変換アルゴリズムを示した．変換アルゴリズムでは，デシジョンテーブルに含めるべき文列の範囲の決定方法を詳述した．本論文に示したアルゴリズムは，むろん絶対的なものではなく，変換の際に必要となるいくつかの基本的な考え方を示すものである．殊に，条件列とアクション列のそれぞれの最大範囲間における共通部分の処理方法には，いくつかの他の方法も考えられる．たとえば，デシジョンテーブルの条件部とアクション部への飛込みを許した表プログラム[2]への変換にすれば，この問題をもっとスマートに扱えるが，紙数の都合上ここでは述べなかった．なお，次の五点には検討の余地が残されている．すなわち，①注釈行やラベルの処理を消極的に扱っているが，これを積極的に利用する，②制限エントリテーブルだけでなく，混合エントリへの変換も考慮する，③open 型テーブルに加え，closed 型テーブルへの変換も含める，④デシジョンテーブル並びに表プログラムへの変換，⑤他の言語，特にアセンブリ語の場合のアルゴリズムの開発，である．

プログラムの自動ドキュメンテーションの立場から，自動フローチャーティングと本論文の DTG とを比較してみると，およそ次のことがいえよう．DTG には特記すべき程の短所は見当らない．一方，自動フローチャーティングに比した DTG の長所として，①出力のスペースが小さくてすみ，見通しが良い，②出力機器(プロッタなど)からくる制限が少ない，③論理が複雑多岐にわたる部分だけをテーブル化している。したがって，原始プログラム自身が持つ情報が比較的よく保存される，等があげられる．

謝辞．計算機利用等で種々御援助いただいた情報処理開発センター研究課長甲賀氏に深謝申上げます．

参考文献

1) P. H. Sherman: Flow Trace: A Computer Program for Flowcharting Program, C. ACM, Vol. 9 (1966), No. 12. など．
2) 守屋，平松：表言語とその処理法について，情報処理，Vol. 13, No. 1, pp. 13～21 (1972).
3) 守屋，斉藤，平松：制限エントリのデシジョンテーブルと表プログラムのプリコンパイラ，情報処理，Vol. 15, No. 1, pp. 77～79 (1974).
4) J. L. Baer et al.: Segmentation and Optimization of Programs from Cyclic Structure Analysis SJCC, pp. 23～35, 1972 など．
5) H. Weinblatt: A New Search Algorithm for Finding the Simple Cycles of a Finite Directed Graph, J. ACM, Vol. 19, No. 1, pp. 43～56 (1972) など．

(昭和 49 年 5 月 1 日受付)
(昭和 51 年 2 月 25 日再受付)