E3　表言語の提案

汎用二次元プログラミング言語と その変換アルゴリズムについて

守屋慎次，平松啓二（東京電機大学情報工学研究室）

1. はじめに

decision tableは，種々な条件の組合せから成る問題を解析しそれを文書化するという点で，流れ図に勝るとも劣らない強力な手法である．さらに，記述されたdecision tableそのものが，計算機のプログラムになり得るという利点をも有している．しかしながらいまだに広く利用されてはいない．これは，decision tableが次のような欠点を有していることにも一因があるものと思われる．すなわち，アルゴリズムの中で(1) condition とconditionの間にactionを含む部分と(2)ループを形成する部分には，decision tableがうまく適用できないことである．

筆者らは，decision tableの持つ これらの欠点を完全に除去する方法を先に提案した〔1，2〕．この提案は上記(1)(2)のような性質を持ついわば静的なdecision tableの欠点を除去したことから，ゆうなれば動的な或は活性化されたdecision tableを構成したことになる．しかしながら，この活性化されたdecision tableともいえるものが，任意のアルゴリズムを非常に簡潔な表形式にプログラムする能力を持つ，一種の二次元プログラム言語であることから，これを表言語（Table Language）と呼ぶこと，さらに表言語で書かれたプログラムを表プログラム（Table Program）と呼ぶことも合わせて提案した〔1，3〕．さらにこの表言語が，decision tableの機能を完全に含むこと，表言語をcomputer programに変換するアルゴリズムが，decision tableの変換アルゴリズムに多少の手続を追加するだけで得られることも示した〔4，5〕．〔4，5〕に与えられているアルゴリズムはマスク法と呼ばれ，そのアルゴリズムの明解さと目的プログラムの所要メモリの少なさなどの利点を持つ反面，ruleの選択を目的プログラムの実行時に行なわなければならない欠点を有している．さらにマスク法は，各ruleが等確率で選択されることを前提としているが，現実には選択の確率が異なるのが通常で，確率が高いrule程判断の回数が少なくなるような目的プログラムの生成方法が望まれる．文献〔6，7〕は，表プログラムの各ruleに与えられた選択の確率を考慮して目的プログラムの平均実行速度を最適化するために，エントロピーの概念を導入した変換アルゴリズムについて述べている．また同時に，表プログラムを構成するdecision table間の包含関係を利用して，目的プログラムの所要メモリを最適化する一方法も示している．さて，これ迄に概観した変換アルゴリズムは，ruleの選択に主眼を置いたもので，表プログラムのcondition entryだけに関するものである．しかしながら，表プログラムのもう一つの主要な構成要素であるaction

entryに関する考慮も重要である．表プログラムのconditionとactionからなる論理をIncident Matrixに表わすと，その表プログラムが持つambiguityの可能性，ruleを減少できる可能性，conditionとactionを含めた最適化の可能性などを意味する顕著な特徴が表われる．文献〔8，9〕ではこの特徴を利用して，conditionとactionを含めた総合的な最適化とambiguityを検出する方法について述べている．

文献〔1，3〕で与えた表言語の一般形には，rule ELSEが1つしか与えられていない．しかしながら，アルゴリズムの表わす論理の意味を考慮すれば，conditionへの異なるentryに対しては，それに対応したrule ELSEが必要である．また文献〔1，2，3〕では，entry pointとなるconditionの行に書かれたブランクがすべて"neglect"になるとして議論を進めたが，"neglect"になってはならない"don't care"も必要なことがある．そこで"neglect"にならない"don't care"という意味を持った新しい記号"don't neglect"を導入する．本論文では，上記のentry毎のELSEと"don't neglect"の二つの新しい概念を導入した表言語と，そのcomputer programへの変換アルゴリズムを中心として述べる．

2. 表プログラムの一般形

表プログラムは，いくつかのdecision tableから構成されている．表プログラムの構成要素としてのdecision tableは，表プログラムのconditionにつけられた文番号によって分離されるから，表言語には，最大conditionへの異なるentryの数だけのELSEが論理上必要といえる．したがって，文献〔1，3〕から，表プログラムの一般形は図1のようになる．

entry Head ⇓

	name		R_1	…	R_j	…	R_n	E_1	…	E_s	…	E_v
	LC_1	C_1	C_1R_1	…	C_1R_j	…	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_mR_j	…	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_uR_j	…	A_uR_n	A_uE_1	…	A_uE_s	…	A_uE_v

図1　表プログラムの一般形

Fig.1の表プログラム（TPと略記する）を構成する各要素の意味は，およそ次の通りである．

name：TPの名称

R_j　：TPのrule番号（$j=1,\cdots,n$）．rule数をnとする．

C_i　：TPのcondition（$i=1,\cdots,m$）．condition数をmとする．

A_k　：TPのactionの集合（$k=1,\cdots,u$）．action数をuとする．

$C_i R_j$：C_iに対するR_jのdecisionを表わす．

$A_k R_j$：R_jに対する実行の記述を表わす．

E_s　：R_j（$j=1,\cdots,n$）に含まれない可能なruleを表現する一種のrule（$s=1,\cdots,v$）．ELSEの数をvとする（$v\leqq m$）．

$C_i E_s$：C_iへのentryに対し，elseとしてE_sが選ばれるとき，$C_i E_s$として＊印が書かれる．

$A_k E_s$：E_sに対する実行の記述を表わす．

entry Head：TPへのentryの1つで，decision tableの持つ唯一のentry pointと同一である．

entry C_i：C_iに制御が移る場合のentry point．

entry A_k：A_kに制御が移る場合のentry point．

3. 表言語の一般則

表言語には次の型がある．すなわち，

(イ) open型表言語（Open Type Table Language：OTTL）

(ロ) closed型表言語（Closed Type Table Language：CTTL）

さらに次の種類を設ける．すなわち，

(a) 制限エントリ表言語（Limited Entry Table Language：LETL）

(b) 拡張エントリ表言語（Extended Entry Table Language：EETL）

(c) 混合エントリ表言語（Mixed Entry Table Language：METL）

このとき，上記(a)(b)(c)の言語レベル上に次の関係がある．すなわち，

$$\mathrm{LETL}\leqq\mathrm{METL}$$

$$\mathrm{EETL}\leqq\mathrm{METL}.$$

これら(イ)(ロ)と(a)(b)(c)の記法と意味は，decision tableで用いられる場合と同様であるから，ここでは述べない．表言語においては使用範囲が拡大されて，(イ)(ロ)と(a)(b)(c)を互に組み合わせて使用できるものとする．

文献〔1，2，3〕では，conditionへのentryの際，そのconditionと同じ行に置かれたcondition entryのブランクがすべて"neglect"となり，そのようなブランクが置かれたruleは選択されないとして議論を進めた．しかしながら，"neglect"になり得ないブランク，すなわち，いかなる場合でも"don't care"であるブランクが必要

な場合がある．そこで，"neglect"してはならない"don't care"として記号 "—" を新たに導入し，これを"don't neglect"と呼ぶことにする．

ここでは，本論文で新たに導入したentry毎のELSEと，"don't neglect"の概念を含めた表言語の一般則について述べる．

表プログラムのcondition entryにおけるブランクを"blank"と書き，文番号（または文名）を"label"と表わす．また，LC_i に文番号が 置かれていることが真であることを"LC_i =label"と書き，置かれていないことが真であることを"LC_i =blank"と表現する．TPの他の構成要素についても同様の記法を用いる．

〔規則1〕 condition entryにおけるblankには"don't care"と"neglect"の2種類の解釈がある． 1つのblankが，異なる時点でこれら2つの解釈を持つことができるが，同時にはいずれか一方の解釈しか持てない．condition entry における記号"—"は"don't neglect"を意味し，決して"neglect"にならない"don't care"を表わす．

以下の規則で，$i=1,\cdots,m$; $j=1,\cdots,n$ とする．

〔規則2〕 $(\forall_i)(\forall_j)(LC_i=\mathrm{blank} \cap C_i R_j=\mathrm{blank} \Rightarrow C_i R_j=\mathrm{don't\ care})$

〔規則3〕 $(\forall_i)(\forall_j)$ $C_i R_j=$don't care のとき，$C_i R_j$ はY(yes)またはN(no)のいずれに解釈してもよい．

entry Headへ制御が移ったことが真であることを"entry=Head"と表現する．同様に"entry=C_i"と"entry=A_k"を，それぞれentry C_i とentry A_k に対して定義する．

〔規則4〕 $\mathrm{entry}=C_1 \Leftrightarrow \mathrm{entry}=\mathrm{Head} \cap LC_1=\mathrm{label}$

〔規則5〕 $(\forall_i)(\forall_j)(\mathrm{entry}=C_i \cap C_i R_j=\mathrm{blank} \Rightarrow C_i R_j=\mathrm{neglect})$

〔規則6〕 $(\forall_i)(\forall_j)$ $C_i R_j=$neglect のとき，そのような$C_i R_j$が置かれたrule R_jは選択されない．

次の規則7で$p=i+1, i+2, \cdots, m$とする．

〔規則7〕 $(\forall_i)(\forall_j)(\forall_p)$ $(\mathrm{entry}=C_i \cap C_i R_j \neq \mathrm{blank} \cap (C_p R_j=\mathrm{blank} \cup C_p R_j=\text{"—"}) \Rightarrow C_p R_j=\mathrm{don't\ care})$

〔規則8〕 $(\forall_i)(\forall_j)(\mathrm{entry}=\mathrm{Head} \cap LC_1=\mathrm{blank} \cap (C_i R_j=\mathrm{blank} \cup C_i R_j=\text{"—"}) \Rightarrow C_i R_j=\mathrm{don't\ care})$

次に，実行のある時点において，論理判断の対象となるconditionが$C_a, C_b, \cdots, C_d$であることが真であるとき，これを"$(C_a, C_b, \cdots, C_d)$=decision"と表現する．このとき，次の規則を設ける．

〔規則9〕 $(\forall_i)(\mathrm{entry}=C_i \Rightarrow (C_i, C_{i+1}, \cdots, C_m)=\mathrm{decision})$，
$\mathrm{entry}=\mathrm{Head} \Rightarrow (C_1, \cdots, C_m)=\mathrm{decision}$

〔規則10〕 $(\forall_i)(C_i, C_{i+1}, \cdots, C_m)=$decisionのとき，condition C_i，

C_{i+1}，…，C_m の論理判断の結果（Y または N）の組合わせと合致する $C_q R_j$（$q=i$，$i+1$，…，m；$j=1$，…，n）の組合わせが，

(i) 唯一の j について存在するとき，そのような **rule** R_j が選択される．

(ii) 複数の j について存在するとき，そのような R_j 間には **ambiguity** が存在する．

(iii) いかなる j についても存在せず，

(iii−i) $C_i E_s \neq$ **blank**（$s=1$，…，v）のような唯一の s が存在するとき **rule** E_s が選択される．

(iii−ii) $C_i E_s \neq$ **blank**（$s=1$，…，v）のような s が複数個存在するとき **error** である．

(iii−iii) $C_i E_s =$ **blank**（$s=1$，…，v），かつ，$C_t E_s =$ **blank**（$t=1$，…，m）のような唯一の s（$s=1$，…，v）が存在するとき，**rule** E_s が選択される．

(iii−iv) $C_i E_s =$ **blank**（$s=1$，…，v），かつ，$C_t E_s =$ **blank**（$t=1$，…，m）のような s（$s=1$，…，v）が複数個存在するとき **error** である．

(iii−v) E_s（$s=1$，…，v）が書かれていないとき **error** である．

ある時点において，実行の対象となる **action** が A_x，A_y，…，A_z の順であることが真であることを "（A_x，A_y，…，A_z）= **execution**" と表現する．

〔規則11〕 $(\overset{\forall}{j})(\overset{\forall}{s})$ R_j（または E_s）が選択されると，$A_k R_j \neq$ **blank**（または $A_k E_s \neq$ **blank**）なる k（$k=1$，…，u）に対し，

（A_1，…，A_k，…，A_u）= **execution** である．

〔規則12〕 $(\overset{\forall}{k})$ **entry** $=$ **A**$_k \cap A_k R_j \neq$ **blank**（または **entry** $= A_k \cap A_k E_s \neq$ **blank**）のような唯一の j（または s）が存在するとき，$A_r R_j \neq$ **blank**（または $A_r E_s \neq$ **blank**）なる r（$r=k$，$k+1$，…，u）に対し，

（A_k，…，A_r，…，A_u）= **execution**

である．

4. 表プログラムの例

本章では，表プログラムの2，3の例について述べる．

〔例1〕 図2の流れ図を表プログラムで表現すると図3が得られる．たとえば $C1$，$C2$，$C3$，$C4$ の判断の結果がそれぞれ Y，Y，N，N のとき，図3では **rule** $R2$ が選択されるから，$R2$ の **action entry** に書かれた×印の **action**，すなわち A_5 が実行されて **exit** となる（規則10，11）．また，GO TO 3によって，**entry** C_3 から判断を始めるときは，C_3 の書かれた **condition entry** の行にあるブランクは "**neglect**" となり（規則5），**rule** $R1$，$R2$，$R3$ のいずれかが選ばれることになる（規則6）．したがって，C_3，C_4 の判断の結果がそれぞれ N，Y のときは **rule** $R1$ が選択され（規則10(i)），A_4，GO TO 1 が実行される（規則11）． このようにして，図2の流れ図の論理が図3の表プログラムに

よって表現される．なお，図2の論理は，図3以外のいくつかの表プログラムで表現することも可能である．

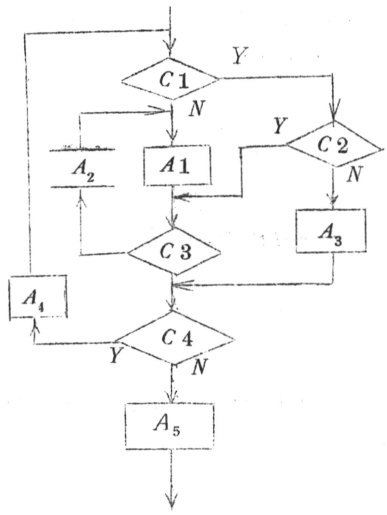


図2.　流れ図の例1

		R1	R2	R3	R4	R5
1	C1	Y	Y	Y	Y	N
	C2	Y	Y	Y	N	
3	C3	N	N	Y		
4	C4	Y	N			
	A_2			×		
	A_1			×		×
	A_3				×	
	A_4	×				
	A_5		×			
	GO TO 1	×				
	GO TO 3			×		×
	GO TO 4				×	

図3.　図2の表プログラム

〔例2〕　図4の流れ図が表わすアルゴリズムは，表プログラムによって図5のように表現することができる．$E1$，$E2$ は2種類のELSEである．どのELSEを選択すべきかは，各

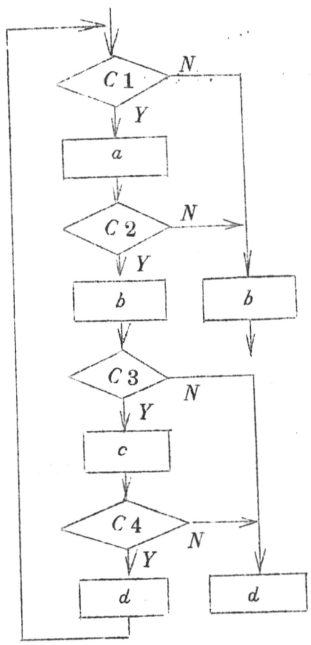


図4.　流れ図の例2

		R1	R2	R3	R4	E1	E2
1	C1	Y				*	
2	C2		Y			*	
3	C3			Y			*
4	C4				Y		*
	a	×					
	GO TO 2	×					
	b		×			×	
	GO TO 3		×				
	c			×			
	GO TO 4			×			
	d				×		×
	GO TO 1				×		

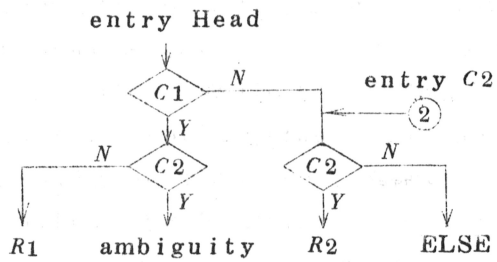
図5.　図4の表プログラム

conditionへのentryに対して書かれた＊印によって決定される（規則10）．

〔例3〕　図6は，同一のcondition stubを有する表プログラムに，文番号と"don't neglect"記号を使用した場合としない場合に，どのように論理が変化するかを図示したものである．図6aには"don't neglect"記号はないが，図6bのように使用すると，entry C2の位置が異なってくる．また，図6c，dのように，"don't neglect"を使用したとしないでは，その表わす論理が大きく異なってくる．

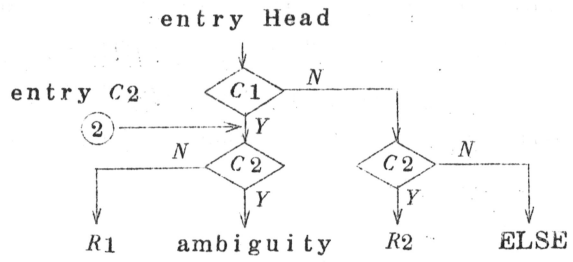
a

		R1	R2	ELSE
	C1	Y		
2	C2		Y	



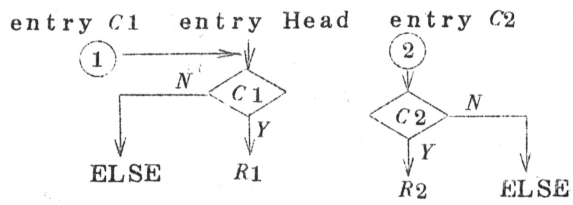
b

		R1	R2	ELSE
	C1	Y		
2	C2	—	Y	



c

		R1	R2	ELSE
1	C1	Y		
2	C2		Y	



d

		R1	R2	ELSE
1	C1	Y	—	
2	C2	—	Y	

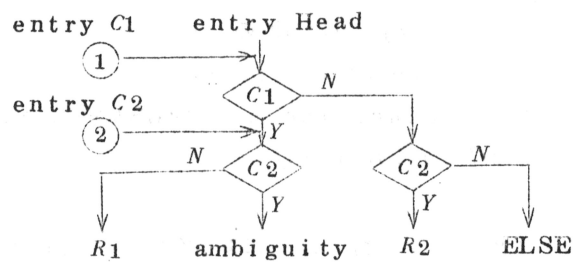


図6　文番号と"don't neglect"記号の効果

5. 表プログラムの変換アルゴリズム

本論文で新に導入した"don't neglect"を持たない表言語を，computer programに変換するアルゴリズムは，筆者らによって先に示されている〔4～9〕．変換アルゴリズムはマスク法〔4,5〕，tree法〔6,7〕そしてincident matrix法〔8,9〕の3つに大別される．大まかにいって，マスク法はアルゴリズムの明解さ，所要メモリの少なさ等に，tree法はエントロピーの概念を導入した実行速度の向上等に，incident matrix法はconditionとactionの両者を含めた総合的な最適化と論理のambiguityの検出等に，それぞれ特長を有している．また各方法の欠点として，マスク法は実行速度の増大，tree法は所要メモリの増加の可能性，incident matrix法はコンパイル時の所要メモリと処理時間の増加の可能性等があげられる．

condition entryに"don't neglect"記号を導入しても，変換アルゴリズムとそれらの特徴には大きな変化はない．したがって，ここではアルゴリズムの明解さ等に特長を持つマスク法を使用して，"don't neglect"記号を導入した表プログラムをcomputer programに変換するアルゴリズムだけについて述べる．なお，本論文で新に取入れた，conditionへのentry毎のELSEに対する処理は，ここで取上げる程の問題ではないので省略する．また，説明の対象としてLETLだけを採用する．METL等の変換アルゴリズムは，ここに述べる方法の考え方の単純な応用で得ることができる．

LETLをcomputer programに変換するアルゴリズムを，コンパイル時と実行時に分けて以下に述べる．

(1) コンパイル時

step 1. 与えられた表プログラムのcondition entryに対し，m行n列の3つのマトリクス$T=[t_i^j]$，$F=[f_i^j]$，$B=[b_i^j]$を次のように作成する．

$$t_i^j=\begin{cases}1 \cdots C_i R_j=(Y \text{ U blank U } -) \text{ のとき}\\ 0 \cdots C_i R_j=N \text{ のとき}\end{cases}$$

$$f_i^j=\begin{cases}0 \cdots C_i R_j=Y \text{ のとき}\\ 1 \cdots C_i R_j=(N \text{ U blank U } -) \text{ のとき}\end{cases}$$

$$b_i^j=\begin{cases}1 \cdots C_i R_j=(Y \text{ U } N \text{ U } -) \text{ のとき}\\ 0 \cdots C_i R_j=\text{blank のとき}\end{cases}$$

$$(i=1,\cdots,m\ ;\ j=1,\cdots,n)$$

ここで，mはcondition数，nはELSEを除いたrule数，t_i^j，f_i^j，b_i^jはそれぞれ，マトリクスT，F，Bのi行j列要素である．

(2) 実行時

entry$=C_i$のとき，規則9によって$(C_i, C_{i+1}, \cdots, C_m)$=decisionとなる．したがって，condition C_i，C_{i+1}，…，C_mが，その時点に与えられた値に対して判断される．

step 2.　$(m-i+1)$行 n列のマトリクス $D=[d_p^j]$ を次のように作成する．すなわち，conditionの判断の結果を考慮して，

$$d_p=\begin{cases} t_p \cdots p\text{番目のcondition}が真のとき． \\ \qquad ここで\ d_p^j=t_p^j\ . \\ f_p \cdots p\text{番目のcondition}が偽のとき． \\ \qquad ここで\ d_p^j=f_p^j\ . \end{cases}$$

$(p=i\ ,\ i+1\ ,\cdots,\ m\ ;\ j=1\ ,\cdots,\ n\ )$

ここに，d_p , t_p , f_p はそれぞれ，マトリクス D , T , F の p 行ベクトルである．

step 3.　n次の行ベクトル $d=[d^j]$ を次のように作成する．すなわち，すべての j に対し，

$$d^j=\bigwedge_{p=i}^{m} d_p^j$$

とする．ここで，演算子∧は，論理積を表わす．

step 4.　n次の行ベクトル $r=[r^j]$ を次のように作成する．すなわち，すべての j に対し，

(イ)　entry=Head ∩ LC_1=blank のとき

$r=d$　　ここに $r^j=d^j$

(ロ)　entry=C_i のとき

$r=d\wedge b_i$　　ここに $r^j=d^j\wedge b_i^j$

step 5.　このstepでruleが選択される．

(イ)　$r^j=1$ のような j が唯一存在するとき，rule R_j が選択される．

(ロ)　$r^j=1$ のような j がいくつか存在するとき，そのような R_j 間にはambiguity が存在する．

(ハ)　$r^j=1$ のような j が存在しないとき，rule ELSEが選択される．（E_1, …, E_v のいずれが選択されるかについては，ここでは述べない．）

〔例〕　図7に示す表プログラムのcondition entryについて，各マトリクスやベクトルを作成すると次のようになる．

$$T=\begin{pmatrix}1&0&1\\1&1&0\\0&0&1\end{pmatrix}$$

$$F=\begin{pmatrix}0&1&1\\1&1&1\\1&1&0\end{pmatrix}$$

$$B=\begin{pmatrix}1&1&0\\0&1&1\\1&1&1\end{pmatrix}$$

		R_1	R_2	R_3	E_1
	C_1	Y	N		
2	C_2		—	N	
3	C_3	N	N	Y	

図7　condition entryの例

ここで,

(1) entry=Head, C_1=真, C_2=偽, C_3=真 のとき, 次のようにruleが選択される.

$$D=\begin{bmatrix} t_1 \\ f_2 \\ t_3 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix},\quad d=[0\ \ 0\ \ 1],\quad r=d=[0\ \ 0\ \ 1]$$

したがってrule R_3が選択される.

(2) entry=C_2, C_2=真, C_3=偽のとき,

$$D=\begin{bmatrix} t_2 \\ f_3 \end{bmatrix}=\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix},\quad d=[1\ \ 1\ \ 0],$$

$$r=d\wedge b_2=[1\ \ 1\ \ 0]\wedge[0\ \ 1\ \ 1]=[0\ \ 1\ \ 0]$$

したがってrule R_2が選択される.

(3) entry=C_3, C_3=偽 のとき,

$$D=[f_3]=[1\ \ 1\ \ 0],\ d=[1\ \ 1\ \ 0],$$

$$r=d\wedge b_3=[1\ \ 1\ \ 0]\wedge[1\ \ 1\ \ 1]=[1\ \ 1\ \ 0]$$

したがって, rule R_1とR_2の間には, ambiguityが存在する.

6. むすび

表言語は, decision tableのconditionとactionに文番号をつけ, condition entryのブランクに"don't care"と"neglect"なる二つの解釈を与えることによって得られた. 本稿では, これら二つの解釈に加えて, "don't neglect"記号を導入し, さらにconditionへのentry毎のELSEという考え方を採入れ, これらを含む表プログラムをcomputer programに変換するアルゴリズムの一例を示した.

表言語を用いて, 任意のアルゴリズムを, 文字通り二次元の表形式に表現することが可能である. 表言語で書かれたプログラムはそれ自身で計算機の原始プログラムともなり得るし, かつまた表形式であることからプログラムの文書性が高いといえよう. しかしながら, conditionが少なくactionの多いアルゴリズムには適しているとはいえない. したがって, 汎用言語の中に埋め込まれた形式で表言語を使用するのが適切であると思われる.

謝辞

日頃御指導いただく青山学院大学 間野浩太郎教授, 御討論いただいた電総研 山本和彦氏, 並に本学情報工学研究室の諸氏に深謝申し上げます.

参考文献

(1) 守屋・平松 「表言語とその処理法について」 情報処理・投稿中

(2) 守屋・平松 「表言語の提案 — 汎用二次元言語」 情報処理学会第12回大会予稿集

(3) 守屋・平松 「表言語とその処理法について — 汎用二次元言語の提案」
信学会研資（1971−9）

(4) 守屋・山本・平松 「制限エントリ表言語の変換アルゴリズム — マスク法」
情報処理学会第12回大会予稿集

(5) 守屋・山本・平松 「混合エントリ表言語の変換アルゴリズム — マスク法」
情報処理学会第12回大会予稿集

(6) 守屋・平松 「表言語のObject Programの最適化について」
信学会研資（1971−11）

(7) 守屋・平松 「表言語の最適変換アルゴリズム — tree法」
情報処理学会第12回大会予稿集

(8) 守屋・斉藤・平松 「表言語の最適変換とambiguityの検出アルゴリズム」
信学会研資（1971−11）

(9) 守屋・斉藤・平松 「表言語の変換にともなう最適化とambiguityの検出 —
Incident Matrix法」 情報処理学会第12回大会予稿集

本 PDF ファイルは 1972 年発行の「第 13 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html