Vol. 21 No. 3　　情報処理学会論文誌　　May 1980

ショートノート

未決定な条件下におけるプログラム・ジェネレーション方式†

梅本栄治†† 村上隆夫††
奥沢修†† 吉井寛††

特定分野や，特定業務を対象とする応用プログラムの代表的な汎用化手法として，テーブル・ドリブン方式と手続き選択形ジェネレータ方式とがある．両者は互いに補間し合う長所，短所を持っており，最適な手法とはいえない，本稿では，両者の長所を合せ持つ新しいジェネレータの概念と，その実現方法を提案する．

1. まえがき

応用プログラムには，多様なユーザ要求に応えるために，様々な汎用化手法が採られており，おおむね次のように分類できる．

〈タイプ1〉 テーブル・ドリブン方式[1]

多様なユーザ要求を包括的に満足する固定的な手続きを持つタイプである（**図1**参照）．個々のユーザ要求はテーブルに記述され，手続きはこのテーブルを参照しながら処理を進める．

〈タイプ2〉 手続き選択形ジェネレータ方式[2]

ユーザ要求を満足するようにあらかじめ準備された手続き群の中から，ユーザ要求に基づいて手続きを選択することによって，ユーザ要求に適した応用プログラムを作り出すタイプである（**図2**参照）．

〈タイプ1〉，〈タイプ2〉を適用するには，応用プログラムの対象分野，業務を限定しなければならないが，ユーザ要求に適合した処理が行えるため，有用性は高い．本稿では，〈タイプ1〉，〈タイプ2〉の問題点を整理し，両者を補間する新しいジェネレータ方式を提案する．

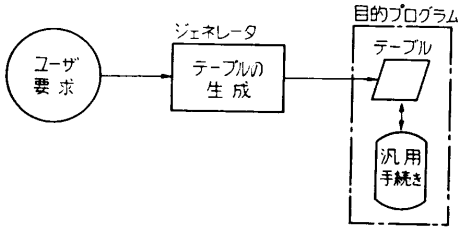

図1 テーブル・ドリブン
Fig. 1 Table driven.

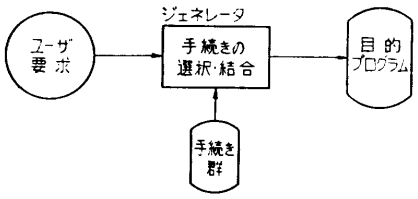

図2 手続き選択形ジェネレータ
Fig. 2 Procedure selecting type.

2. 問題点の整理

〈タイプ1〉，〈タイプ2〉を目的プログラム（ユーザ要求を満たすプログラム）の処理速度，メモリ使用量およびユーザ要求に対する融通性について比較すると**表1**のようになる．融通性は，できる限りユーザ要求の指定を目的プログラムの実行直前に行うことにより向上できる．〈タイプ1〉は，この条件を満足している．〈タイプ2〉も，いくつかの特定のユーザ要求については目的プログラムの実行直前に指定できるように

表1 汎用化方法の比較
Table 1 comparison of generalization methods.

比較項目＼タイプ	タイプ1（テーブル・トリブン）	タイプ2（手続き選択形ジェネレータ）
目的プログラムの処理速度	テーブルを参照し，処理手順を決定するため，タイプ2に比べ遅い	処理手順があらかじめ決定されているため，タイプ1に比べ速い
目的プログラムのメモリ使用量	テーブルおよび，テーブル参照と処理手順の決定が含まれるため，タイプ2に比べ大	処理手順の決定のためのテーブル，手続きが不要なため，タイプ1に比べ小
ユーザ要求の変更に伴う目的プログラムの融通性	テーブルの変更により容易	手続きの変更を要するため困難

† The Program Generation Method on the Indefinite Condition by EIJI UMEMOTO, TAKAO MURAKAMI, OSAMU OKUZAWA and HIROSHI YOSHII (Yokosuka Electrical Communication Laboratories, N. T. T.).

†† 電電公社横須賀電気通信研究所データ通信研究部

することができる．しかし，〈タイプ2〉では目的プログラム生成時に指定するユーザ要求（生成時情報）と，目的プログラム実行時に指定するユーザ要求（実行時情報）とは，ジェネレータ設計段階で区分けされており，自由に区分けできる〈タイプ1〉ほどの融通性はない　したがって生成時情報と実行時情報の区分けを目的プログラム生成時に任意に行うことにより融通性においては〈タイプ1〉に準じ，処理速度とメモリ使用量においては〈タイプ2〉準ずる新しいジェネレータ方式が必要である．（このタイプのジェネレータを不確定（生成時情報と実行時情報の区分けが確定していない）形ジェネレータと呼称することとする．）

3. 不確定形ジェネレータの概念

3.1 ユーザ要求の構造

特定分野や特定業務を対象とする応用プログラムのユーザ要求は，各ユーザ要求を一意に識別可能とする属性 (attribute) 集合 ($\boldsymbol{X}$) の各要素 (element) に属性値 (attribute value) を与えることによって，定義できる．いま，属性集合の要素に属性値を与えた場合の属性集合 ($\boldsymbol{X}$) の状態を $S(\boldsymbol{X})$ と表わすこととする．不確定形ジェネレータでは，$\boldsymbol{X}$ のうち，完全に決定できるもの（後で変更が予想されない属性）のみを生成時に指定することができる．すなわち，式(1)に示すような状態が指定可能である．

$$S(\boldsymbol{X})=\{(x_1=a_{11}\cup a_{12}),\ (x_2=*)\cdots\cdots \cdots\cdots(x_n=a_{n1})\} \tag{1}$$

ここで，x_n は $\boldsymbol{X}$ の要素，a_{ij} は x_i が取り得る属性値であり，$x_1=a_{11}\cup a_{12}$ は，x_1 の値が生成時には a_{11} か a_{12} か分らないことを，また，$x_2=*$は，x_2, の値が生成時には全く不定なことを示す．式(1)で，x_1, x_2 のように，あいまいさのある要素を不確定要素と呼称することとする．これに対して，x_nを確定要素と呼称する．不確定形ジェネレータでは確定要素と不確定要素の区分けは全く任意である．

3.2 不確定形ジェネレータによる手続き選択概念

一般に手続き選択形ジェネレータにおける手続きの集合からの，ひとつの手続き f_k の選択を式(2)のように表わすことができる．

$$f_k=S_{all}(\boldsymbol{X})\cdot\boldsymbol{F} \tag{2}$$

ただし，$S_{all}(\boldsymbol{X})$ は，$\boldsymbol{X}$ の全要素が確定した場合の状態である．これに対して不確定形ジェネレータでは，生成時に不確定状態 $S(\boldsymbol{X})$ によって一旦，$\boldsymbol{F}$ の任意の部分集合 $\boldsymbol{F}'$ を選択し（式(3)参照），実行時に不確定要素を確定する（式(4)参照）ことにより最終的に所要の手続き f_k を選択することができる．

$$\begin{cases}\boldsymbol{F}'=S(\boldsymbol{X})\cdot\boldsymbol{F} & (3)\\ f_k=S_{all}(\boldsymbol{X}')\cdot\boldsymbol{F}' \quad (\boldsymbol{X}'\text{：不確定要素の集合}) & (4)\end{cases}$$

以下に例を示す．

〈例〉

$\boldsymbol{X}=\{x_1, x_2, \cdots, x_n\}$ のうち，x_1 が不確定要素であり，実体として a_{11} または a_{12} をとるものとすると，$\boldsymbol{X}$ の状態 $S(\boldsymbol{X})$ は式(5)のようになる．

$$\begin{aligned}S(\boldsymbol{X})&=\{(x_1=a_{11}\cup a_{12}),(x_2=a_{21}),\cdots,(x_n=a_{n1})\}\\&=\{(x_1=a_{11}),(x_2=a_{21}),\cdots,(x_n=a_{n1})\}\\&\quad\cup\{(x_1=a_{12}),(x_2=a_{21}),\cdots,(x_n=a_{n1})\}\\&=S_{all}'(\boldsymbol{X})\cup S_{all}''(\boldsymbol{X})\end{aligned} \tag{5}$$

ただし，

$$\begin{cases}S_{all}'(\boldsymbol{X})=\{(x_1=a_{11}),(x_2=a_{21}),\cdots,(x_n=a_{n1})\}\\ S_{all}''(\boldsymbol{X})=\{(x_1=a_{12}),(x_2=a_{21}),\cdots,(x_n=a_{n1})\}\end{cases}$$

である．したがって，式(3)は次のようになる．

$$\begin{aligned}\boldsymbol{F}'&=S(\boldsymbol{X})\cdot\boldsymbol{F}=S_{all}'(\boldsymbol{X})\cdot\boldsymbol{F}\cup S_{all}''(\boldsymbol{X})\cdot\boldsymbol{F}\\&=f_k'\cup f_k''\end{aligned} \tag{6}$$

ただし，$f_k'=S_{all}'(\boldsymbol{X})\cdot\boldsymbol{F}$，$f_k''=S_{all}''(\boldsymbol{X})\cdot\boldsymbol{F}$ である．式(6)は，**図3**に示す処理の生成を意味している　不確定要素 (x_1) に対しては，式(4)のように，目的プログラムの実行時に属性値 (a_{11} または a_{12}) を指定することにより，処理手順を最終的に決定することができる．

4. 不確定形ジェネレータ実現上の問題点と実現法の提案

4.1 実現上の問題点

前出の式(3)を「生成時における手続き選択の論理

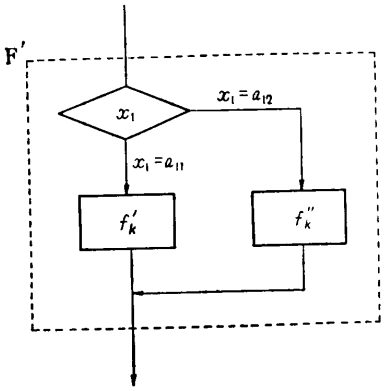

図 3　生成された論理

Fig. 3　Generated logic

(L_1)」，式(4)を「実行時における手続き選択の論理(L_2)」と呼称することとする．また，手続き選択の論理全体を L とすると，三者には式(7)の関係がある．

$$L=L_1 \cup L_2 \qquad (7)$$

すなわち，L_1, L_2 の両論理は，L の部分集合と，その補集合であり，不確定形ジェネレータでは，その境界は任意であるため，あらかじめ両論理を分離しておくことができない．(従来の手続き選択形ジェネレータはあらかじめ分離してしまっていたため，十分な融通性が得られなかった．) したがって，不確定形ジェネレータでは，論理 L から，目的プログラムとして任意の論理 L_2 を残す方法が必要となる．

4.2 実現法の提案

デシジョン・テーブル[3]を使用して上記問題点を解決する方法を提案する．属性集合 X はユーザ要求と手続きの関係（論理 L）を記述した限定形デシジョン・テーブルのコンディションの集合 $C=\{c_k\}$ に置き替えることができる．すなわち，$x_i=a_{ij}$ は $c_k=$ Yes, (c_k: $x_i=a_{ij}$ か？) に対応させることができる．したがって，式(3)，(4)は，式(8)，(9)に置き替えることができる．

$$\begin{cases} F'=S(C)\cdot F & (8) \\ f_k=S_{all}(C')\cdot F' & (9) \end{cases}$$

ただし，C' は不確定要素の集合である．前節で述べた問題点（L からの任意の L_2 の抽出）は式(8)に帰結することができ，これは図4に示す手順により，L を記述したデシジョン・テーブルを変形することによって実現可能である．次に図4の各フェーズを説明する．

〈デシジョン・テーブルの変形規則〉

(1) 不要ルールの削除

コンディション c_j がエントリとして k(k; Y または N) を取ることが確定していれば，$c_j=\bar{k}$(k=Y ならば，$\bar{k}$=N) のエントリを含むルールを削除する．

(2) 不要コンディションの削除

次の規則に従ってコンディションを削除する．

① 全エントリが同記号からなるコンディションを削除する．

② (Y, −) または (N, −) の組合せからなるコンディションを削除する．

この規則により，ⓐ確定コンディション自身，ⓑ確定コンディションと依存関係にあり，不要ルール削除後，無意味になるコンディションが削除される．

(3) 不要なアクションの削除規則

エントリ中に実行を示す「×」の無いアクションを削除する．

以上述べた変形処理の具体例を図5に示す．このような変形処理によって得られたデシジョン・テーブルは「実行の論理 (L_2)」のみを有している．すなわち，図5(d)が，これに対応しており，図5(e)と等価で

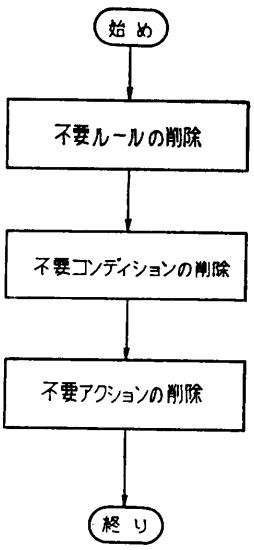

図4 変形の手順
Fig. 4 Process of transformation.

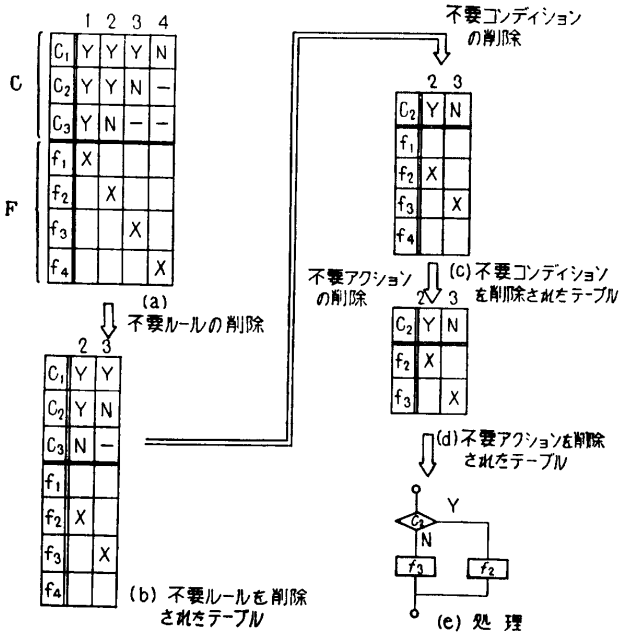

図5 変形の例（確定要素 $C_3=N$）
Fig. 5 Example of transformation.

ある．デシジョン・テーブルを COBOL, FORTRAN 等のソース・プログラムに変換する手法[4]は各種実用化されている．

5. むすび

ここに述べた不確定形ジェネレータは，実際に数値制御用ポストプロセッサを対象とするジェネレータに適用し[5]，処理速度，メモリ使用量および融通性に対して目標を達成することができた．ユーザ要求と処理との間の複雑な関係を整理する上でも，デシジョン・テーブルは有用であり，ソフトウェア生産性の向上にも寄与し得たと考える．

参考文献

1) 吉田ほか：マイクロプロセッサ，ソフトウェア作成支援システムの汎用化手法について，情処学会設計自動化研究会資料，DA 29-1.
2) 高橋，岡本：ポストプロセッサ・ジェネレータ，沖電気研究開発，Vol. 41, No. 2, p. 49～(1975).
3) McDaniel, H. : An Introduction to Decision Logic Tables.
邦訳：岸田孝一：デシジョン・テーブル入門，日本経営出版会 (1970).
4) Pollack, S. L.: Conversion of limited-entry decision table to computer programs, com, of ACM Vol. 8, No. 11, p. 677～682 (1965).
5) 梅本，奥沢：デシジョン・テーブルを用いたジェネレータ，情処学会第 17 回全国大会 (1976).

(昭和 54 年 8 月 31 日受付)
(昭和 54 年 10 月 25 日採録)