正則な状態遷移図の全遷移を網羅するテストデータ生成アルゴリズム†

渡辺坦††

プログラム仕様に基づくテストデータ自動生成は，ソフトウェアの信頼性と生産性の向上のために強く望まれている．本論文では，プログラム仕様を連接と場合分け，反復で構成される正則状態遷移図として書き，入出力を列挙型データとし，内部的制約条件を，ある遷移に伴う内部出力が他の遷移を起動する内部入力となるという形で書くならば，その遷移を網羅的にテストする入力系列の集合の自動生成がかなりの成功をおさめることを示す．自動生成の過程では，入口から出口に至る実行可能経路の探索のために，何段階かの副次目標を設定する．自動生成時間は，遷移数が数十個の場合，大型計算機で数秒から数十秒である．

1. まえがき

プログラム中のすべての分岐を網羅的にテスト実行させることは，テスト上の大きい目標の一つとなっている[1]が，実現はむずかしい．十分にテストした場合でも，分岐網羅率は85%程度といわれている[2]．100%の分岐網羅率を達成する一つの方法は，任意の分岐に対してそこを通る実行可能な経路を求め，その経路に合わせたテストデータを作ることである．

この問題に対する一般解を求めることは著しく困難である．記号実行[3]では，分岐の条件式の導出過程をたどり，それを入力で構成される式に帰着させ，それに基づいて，任意の分岐に対してそれを成立させる入力の値域を求めることにより，この問題を解こうとする．この方法は，反復を含む大きいプログラムには適用が困難であり，また，分岐の条件式を解くのも容易でない．

原因結果グラフ法[4),5]では，入力の組合せと出力の対応関係を論理式で表し，与えられた論理式で述べられているすべての関係を確認するテスト項目の最小の集合を求める．これは，入力と出力の関係を論理式として表現できる問題に対しては実用化できる強力な方法であるが，テスト項目からテストデータを作る問題が入手に委ねられている．また，テストの条件設定が互いに独立にできる場合を扱っているので，相互に依存関係のあるテストデータの生成には適さない．

試行錯誤法[6]では，各分岐に対してそれを成立させる入力データを試行錯誤的に求める．これでは，分岐の条件式を線形の等式と不等式に限定し，乱数で試行値を求めるのは独立な変数に限定するなど，試行回数を減らす各種の工夫が行われているが，変数の数が多くなると，試行の成功確率は著しく低くなる．

プログラム仕様を状態遷移図として書き，テストデータを作成する方法はChow[7]が提案しているが，それには入力間の制約条件の解決法は示されていない．状態遷移図に基づくテストデータ自動生成プログラムとしては，ATLAS[8]が開発されている．ATLASは入力間に制約条件を設定でき，テストの実行まで行うシステムであるが，実行可能経路は総当り的に数えあげるので，その数が膨大になる危険性がある．

本論文では，状態遷移図として書かれた仕様に基づいて，そのすべての遷移を網羅できるテストデータのできる限り小さい集合を求めることを目指す．入力間の制約条件は，副次目標を設定して解決する．ただし，任意の状態遷移図を対象とするのではなく，実用性を失わない範囲内で制約条件をつけ，その範囲内でこの問題を解決するアルゴリズムを提示する．すなわち，

（1） プログラムの仕様は状態遷移として書かれており，その入力と出力は列挙型の値をとる．

（2） 状態遷移は，連接と場合分け，反復の3要素で構成される「正則」な状態遷移である．

（3） 入力間の制約条件としては，「ある入力列で生起される遷移に伴う出力が他の遷移を生起させる入力となる」ことが許される．

という制約条件を設け，その条件下で実行可能な経路を求めるアルゴリズムを提示し，求めた経路を全部集めると，対象とする状態遷移の全遷移が網羅されてい

† An Algorithm to Generate Test Data for Covering All Transitions of Regular State-Transition Diagram by TAN WATANABE (2nd Department, Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所第2部

るようにする．入力が列挙型の値の場合，実行可能経路の値が求まると，その個々の経路をたどることにより，テストデータとしての入力列をただちに構成できる．

状態遷移図は，電子交換機の交換機能や，各種のシーケンス制御，プロトコル，マン・マシン会話機能等を扱う多くの分野でプログラムの仕様記述に用いられており，その入力も列挙型の値が多い．正則な状態遷移という条件は，構造化プログラミング技法により，制御の流れを連接と場合分け，反復の3要素で構成するようプログラムを設計すれば，自ずから満たされる．したがって，本方式は，それらの分野におけるプログラムの網羅的テストに適用可能である．

以下，2章では，対象とする状態遷移についての制約条件を明確にする．3章では，遷移網羅のアルゴリズム記述に必要な経路式，ならびに，状態と遷移に対する座標という概念を導入する．4章では遷移網羅用のテストデータ生成アルゴリズムの概要を述べ，5章でそのプログラム化の結果を述べる．アルゴリズムの詳細は付録に示す．

2. 条件設定

2.1 正則な状態遷移

遷移の流れの錯綜した状態遷移では，任意の遷移に対してそこを通る実現可能な順路の検出は困難である．本論文で扱うのは，以下のようにして構成される**正則な状態遷移**とする．

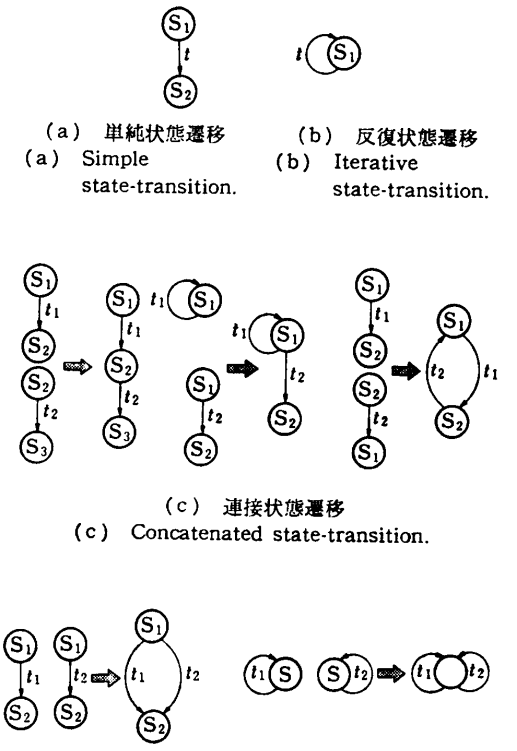
（1） 一つの遷移のみで構成される状態遷移は正則である（**図1** a，b）．

（2） 二つの正則な状態遷移 St_1 と St_2 において，St_1 の終状態と St_2 の始状態が一致するならば，St_1 の終状態を St_2 の始状態として連接した状態遷移は正則である（図1 c）．

（3） 二つの正則な状態遷移 St_1 と St_2 において，両者の始状態が一致し，両者の終状態も一致し，かつ，両者の始状態で許容される入力に重複がなければ，St_1 と St_2 の始状態と終状態をそれぞれ一つの状態にまとめた状態遷移も正則である（図1 d）．

一つの遷移のみから成り，始状態と終状態の異なる状態遷移を単純状態遷移という．始状態と終状態の一致した状態遷移を反復状態遷移という．上記(3)により合成された状態遷移は，St_1 と St_2 に場合分けされているという．

構造化プログラミングにより制御の流れを単純化することの効果は広く認められるようになってきた．正則な状態遷移は構造化プログラミングに従った仕様記述であり，これは無理な要請というより好ましい要請といえる．



（a） 単純状態遷移　（b） 反復状態遷移
（a） Simple state-transition.　（b） Iterative state-transition.

（c） 連接状態遷移
（c） Concatenated state-transition.

（d） 場合分け状態遷移
（d） Case-wise state-transition.

図 1 正則状態遷移
Fig. 1 Regular state-transition diagram.

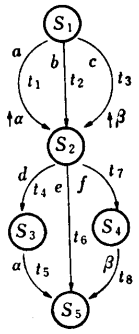
2.2 列挙型の入出力

許容される入力と出力は，数え挙げのできる離散値であり，そのおのおのには固有の名前をつけることができるものとする．これは，入力列と出力列の対応関係を単純化し，入力列ならびにその結果の予測値の自動生成を可能にするための条件である．

スイッチのオン，オフや，電話番号の各桁の数値，ACK/NAK 信号等，多くのデータは列挙型である．連続データ等，列挙型でないデータも，プログラムの制御の流れからみて区別なく扱えるデータを一つのグループにまとめてゆくと，そのグループ名は列挙型データとして扱える場合が多い．

2.3 純粋の状態遷移

状態遷移においては，「状態 S_1 に入力 I が与えられると遷移 t が生起されて状態 S_2 に移り，それに伴って出力 O が出る．」という形で，遷移元状態と，遷



S_1,S_2,S_3,S_4,S_5: 状態 (state)
$t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8$: 遷移 (transition)
a,b,c,d,e,f: 外部入力
$\uparrow\alpha$: 内部出力 (internal output) α
$\uparrow\beta$: 内部出力 β
α,β: 内部入力 (internal input)

図 2 内部入力と内部出力を持つ状態遷移図
Fig. 2 State-transition with internal input and internal output.

移を生起する入力，遷移先状態，ならびに遷移に伴う出力の関係のみが与えられているものとする．したがって，状態の内部記述や，遷移に伴う内部処理記述は不要とみる．遷移に伴う出力は必須としない．一つの状態から無条件に次の状態に移る場合は空入力 ε を与えるものとする．

2.4 内部データの授受による関係づけ

ある遷移を生起させる入力は他の遷移に伴う出力として得られたものであるとするのは許されるが，入力間にはそれ以外の相互関係はないものとする．上記の場合の入力と出力を，それぞれ，内部入力，内部出力と呼ぶ．内部入力でない入力を外部入力と呼ぶ．**図 2** にその例を示す．

2.5 実行可能性

内部出力と内部入力の受渡し関係において，遷移 t_1 の生起には遷移 t_2 で出力される O_2 が必要で，t_2 の生起には t_1 で出力される O_1 が必要というようなデッドロックに至る矛盾は含まないものとする．また，入口から到達不可能な状態や出口へ到達不可能な状態はなく，すべての遷移は実現可能なものとする．

3. 諸概念の導入

3.1 経 路 式

プログラム上の一つの実行経路は，その仕様記述としての状態遷移において，入口状態から出口状態に至る遷移の列として表される．本論文では，遷移列の集合を**経路式** (route expression) という式で表現する．経路式は次のようにして構成される．

（1） 一つの遷移 t を表す経路式は，

t

と表わす．その始点は t の遷移元状態であり，終点は t の遷移先状態である．

（2） 二つの経路式 r_1 と r_2 において，r_1 の終点と r_2 の始点が一致するならば，r_1 の経路のうしろに r_2 の経路を連接した経路は，

$r_1 : r_2$

と表す．その始点は r_1 の始点であり，終点は r_2 の終点である．

（3） 経路式 r_1, r_2 において，おのおのが共通の始点 S_1 と共通の終点 S_2 をもつならば，r_1 の経路と r_2 の経路を合わせた場合分け経路を

$r_1|r_2$

と表す．その始点は S_1 であり，終点は S_2 である．

（4） 経路式 r の始点と終点が一致するとき，r の経路を n 回通る反復経路は，

r^n

と表す．r の経路を 0 回以上任意回通る経路は

r^*

と表す．

（5） r が経路式のとき，それをかっこでくくった

(r)

も経路式であり，それは r と同一の経路を表す．

連接と場合分けの組合せに対しては，

$r_1 : r_2|r_3 : r_4=(r_1 : r_2)|(r_3 : r_4)$

と，連接の結合強度が高いものとする．以下，経路式 r の表す経路を，たんに，経路 r と呼ぶ．

前後に共通の部分経路をもつ場合分け経路式

$r_h : r_1 : r_t|r_h : r_2 : r_t$

は，

$r_h : (r_1|r_2) : r_t$

と同じである．ここで，r_1 と r_2 の一方がない

$r_h : r_1 : r_t|r_h : r_t$

のような場合，空遷移 null を導入すれば

$r_h : (r_1|\mathrm{null}) : r_t$

と同じである．r_h と r_t の一方がない

$r_h : r_1|r_h : r_2,\ r_1 : r_t|r_2 : r_t$

は，それぞれ

$r_h : (r_1|r_2),\ (r_1|r_2) : r_t$

と同じである．

経路式 r において，そのなかに含まれるどの場合分

け経路式についてみても，前または後に共通の経路が重複していなければ，r を**正規経路式** (normal route expression) と呼ぶ．経路式 r をそれと等価な正規経路式に変えることを q の正規化と呼ぶ．正規化は，経路式が長大になるのを防ぐのにきわめて有効である．

3.2 状態と遷移の座標づけ

ある状態から他の状態へ至る経路を探すとき，途中で通る必要のある状態はどれか，また，ある遷移 t がその経路に含まれる可能性が有るか否か，等がわかると探しやすい．そこで，正則な状態遷移に対して，おのおのの状態や遷移の相互の位置づけがただちにわかるようにするため，状態と遷移のおのおのに座標をつける．

正則状態遷移においては，入口状態から出口状態に至るどの経路をとっても必ず通過しなければならない状態としての**必須通過点**が何点かある．その第 j 番目の必須通過点としての状態に S_j という座標をつける．入口状態の座標は S_1 である．必須通過点 S_j から S_{j+1} に至る経路の先頭遷移には T_j という座標をつける．S_j から S_{j+1} に至る経路が m 本に場合分けされているならば，その各経路の先頭遷移にそれぞれ，$T_jC_1, T_jC_2, \cdots, T_jC_m$ という座標をつける．S_{j+1} は，これらの場合分けされた経路の共通の合流点である．状態 S_j から出て S_j に戻る反復経路に対しては，その先頭遷移に L_j という座標をつける．その反復経路が m 本あるならば，そのおのおのの先頭遷移にそれぞれ $L_jC_1, L_jC_2, \cdots, L_jC_m$ という座標をつける．

遷移の座標の一般形は，

$t, t_1 \cdot t_2 \cdot \cdots \cdot t_n$

のいずれかの形をとる．ここに，$t, t_1, t_2, \cdots, t_n$ は，

T_j, T_jC_k, L_j, L_jC_k

のいずれかの形である．状態の座標の一般形は，

$S_j, t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_j$

のいずれかの形をとる．ここに，$t_1, t_2, \cdots, t_n$ は上記の遷移の場合と同じ形である．これらのうち，結合用の点を含まない形についてはすでに説明したので，以下，点で結ばれた表記形式をとる座標について，その定め方を述べる．

状態

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_j$, $n \geqq 0$, $j \geqq 1$

において，この状態から出て $t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_{j+1}$ に至る経路の先頭遷移が一つしかないときは，その先頭遷移に

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_j$

という座標をつける．先頭遷移が m 個あるときは，そのおのおのに

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_1, t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_2,$

$\cdots, t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_m$

という座標をつける．

遷移

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_k$, $1 \leqq k \leqq m$

から始まり，状態 $t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_{j+1}$ に至る経路上に，m 本の経路のどれから出発しても必ず通らなければならない必須通過点が，$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_{j+1}$ を除いて，p 個あるとき，その各必須通過点の状態に，先頭から順番に

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_k \cdot S_1,$

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_k \cdot S_2, \cdots,$

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot T_jC_k \cdot S_p$

という座標をつける．

状態

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_j$, $n \geqq 0$, $j \geqq 1$

を始点と終点とする反復経路において，その経路の先頭遷移が一つしかないときはそれに

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_j$

という座標をつけ，m 個あるときはそのおのおのに

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_jC_1, t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_jC_2,$

$\cdots, t_1 \cdot t_2 \cdots t_n \cdot L_jC_m$

という座標をつける．移遷

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_jC_k$, $1 \leqq k \leqq m$

から始まり状態 $t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_j$ に戻る反復経路上に，その反復経路の必須通過点が $t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot S_j$ を除いて p 個あるとき，その各必須通過点の状態には，先頭か

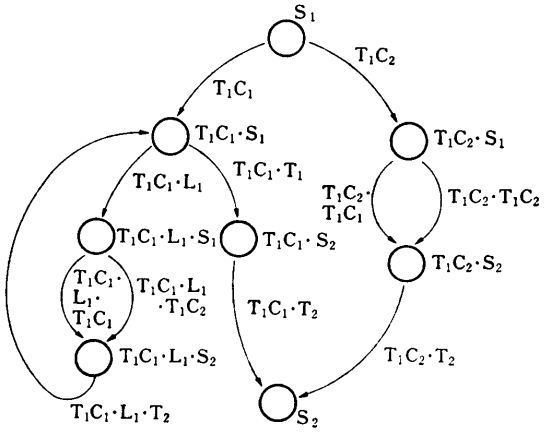


図 3 正則状態遷移図の座標づけ
Fig. 3 Coordinates of regular state-transition.

ら順番に

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_j C_k \cdot S_1$,

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_j C_k \cdot S_2, \cdots$,

$t_1 \cdot t_2 \cdot \cdots \cdot t_n \cdot L_j C_k \cdot S_p$

という座標をつける.

以上の座標づけでは，枝分れの細部に入るにつれて点による結合の数が多くなる．**図3**に，座標づけの例を示す．座標づけは，計算機で処理することもできるが，人手でも容易に行える．この座標を利用すると，任意の二つの状態や遷移に対して，それらの前後関係や並列的配置関係，ある状態から他の状態へ至るまでの必須通過点などを容易に読みとることができる.

4. 遷移網羅用のテストデータ生成アルゴリズム

正則状態遷移図のすべての遷移を網羅できるテストデータ集合を求めるには，まず，その全遷移を網羅する実行可能な経路の集合を求め，それを経路式として表現する．次に，その経路集合からの代表経路の選出を未被覆遷移がなくなるまで続け，各代表経路に対する入力列を求める.

全遷移を網羅する実行可能な経路集合を求める過程は，次の3段階に分かれる.

第1の網羅的訪問の段階では，まず，入口状態で入力を一つ与えることによって実現できるすべての遷移を求める処理を行い，ついで，前のサイクルで到達した状態において入力を一つ与えることにより実現できるすべての未訪問遷移を求めることを繰り返す．すなわち，第 i 回目の入力によって到達した状態において入力を一つ与えることにより実現できる未訪問遷移を求める処理を，$i=1,2,3,\cdots$ と変えてゆきながら繰り返す．ここで，内部入力を必要とする遷移は，それに対応する内部出力が得られるまで棚上げにし，それが得られた次のサイクルで実現する．この処理を既訪問の遷移が増えなくなるまで繰り返す.

上記過程は，状態遷移網の入口で発生した波が1サイクルに一つの遷移の割合で進み，出口に達すると消滅する現象になぞらえられる．その波頭は，未訪問の遷移を選んで進み，既訪問の遷移ならびに未実現の内部入力を必要とする遷移のところでは，消滅しないで停滞している．この第1段階で，すべての遷移が出口に達した波により被覆されていれば，プログラムの実行は終了する.

第2の出口探索の段階では，出口に至らずに停滞している波頭を，通りやすい経路を選んで，早く出口へ導こうとする．そのため，停滞している波頭に対し，外部入力を一つ与えることにより生起される遷移を選ばせて進める．内部入力を必要とする遷移ばかりの状態に達した波頭に対しては，その内部入力を出力する遷移を通って当状態に至る経路を求めることを試みる．状態遷移の反復に対しては，反復経路上の全遷移を訪問し終わったら，それ以上その反復経路を選ぶことをやめ，無限ループに陥ることを防ぐ．以上の処理を，全遷移が被覆されるか，または各波頭が先へ進めなくなるまで繰り返す．全遷移が被覆されればプログラムは終了する.

遷移 t を通り状態 s へ至る実行可能な経路を探すには，t と s の座標から，途中の必須通過点 $s_1, s_2, \cdots, s_n$ を求め，まず s_1 から s_2 への実行可能経路，ついで s_2 から s_3 への実行可能経路というように，副次目標を設けながら進める．ここに，s_1 は t の遷移先状態，s_n は s とする.

第3の徹底探査の段階では，出口に至らずに停滞しており，かつ未被覆の遷移を経由している波頭に対し，それが内部入力を必要とするものであっても出口へ向けて進めることを試みる．探査の順序は，出やすいものから先に出すという原則に従って，出口までの必須通過点の数の小さいものから先に進める．その際，内部入力が必要であれば，それを出力する遷移を途中で通る経路を探す．この処理を一巡してもまだ未被覆遷移が残っていれば，それを含む反復状態遷移またはその遷移の起動に必要な内部入力を出力する遷移を含む反復状態遷移を途中に挿入し，再度上記処理を繰り返す．この第3段階を，未被覆遷移がなくなるか，またはどの波頭も先へ進めなくなるまで繰り返す．後者の場合は，本アルゴリズムによっては全遷移を網羅できない場合である.

このアルゴリズムを擬似文で書いたものを付録に示す.

5. テストデータ自動生成プログラム

上記アルゴリズムによりテストデータを自動生成するプログラムを開発した.

状態遷移図では，状態数は数百個から千個，遷移数はその数倍で，多くの実際の問題を表現できる．しかし，入口から出口へ至る経路の数は，一般に，状態数に対して指数関数的に増大し，膨大な数となる．本アルゴリズムでは，途中のサイクルにおいて，各状態に

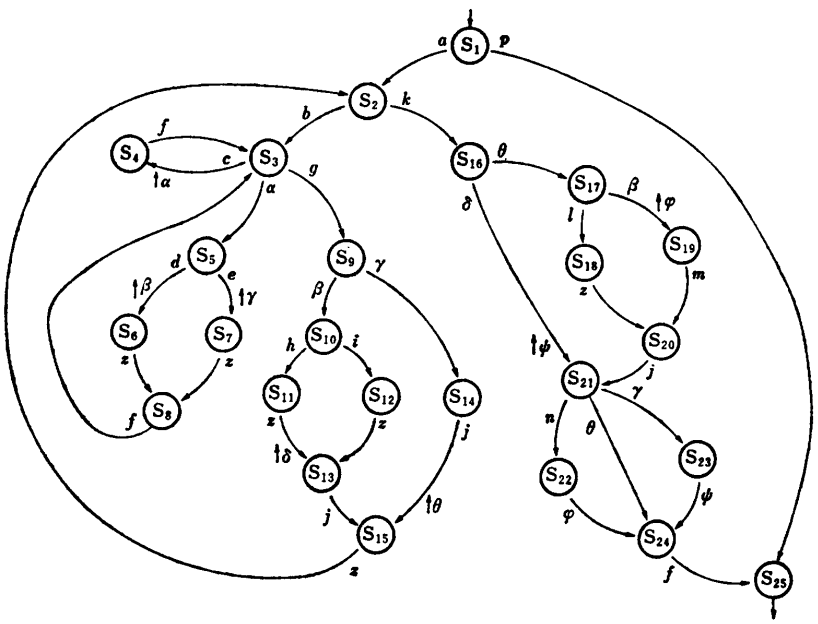
対し，その時点で実現可能なすべての経路を記録するので，それらをどのようにして小さいメモリ量で表現するかがプログラム作成上の大きい課題となる．

経路は，個々別々に記録するのではなく，3.1 節で述べたように，正規経路式として表現する．その物理表現はセグメントと呼ぶもので構成する．セグメントは，連接セグメントと場合分けセグメント，反復セグメントに分かれる．連接セグメントは，遷移またはセグメントを直列に並べたものである．場合分けセグメントは，遷移またはセグメントを並列に列べたものである．反復セグメントは，始点と終点が同じセグメントである．経路式は，セグメントの階層として表現される．一つのセグメントの最上位階層では，連接，場合分け，反復のうちの1通りの結合しか許されない．それを構成する下位セグメントは，上位のセグメントと異なる結合形式のセグメントとなる．一つの経路式全体も一つのセグメントである．

一つのセグメントが，いくつかのセグメントに共通の下位セグメントであってもよい．経路式の算出過程で，新しいセグメントが作られ，古いセグメントへのポインタが消される．セグメントを入れるメモリ領域が不足すると，ガーベジコレクションを行い，どこか

表 1 テストデータ生成例
Table 1 Examples of test data generation.

ケース	状態数	遷移数	内部入力参照数	ループ数	ループの最大多重度	所要サイクル数	所要時間(秒) (HITAC M 180 CPU)	生成入力列数
A	5	9	0	4	3	9	0.53	1
B	15	25	8	3	2	18	1.03	4
C	25	35	9	3	2	32	2.28	4
D	23	49	5	6	4	25	3.51	15
E	23	51	9	19	5	23	4.83	6
F	31	58	6	9	7	29	19.58	7
G	29	60	0	3	2	31	5.05	18
H	34	83	14	14	4	25	17.40	14



英小文字：外部入力
ギリシア文字：内部入力
$\uparrow\alpha,\cdots,\uparrow\psi$：内部出力 $\alpha,\cdots$，内部出力 ψ

図 4 対象とする状態遷移図の例
Fig. 4 Example of state-transition diagram for test-data generation.

```
SEQ =  1
INPUT  = A      . B      . C      . F      . D      . Z      .
         F      . G      . I      . Z      . J      . Z      .
         B      . C      . F      . D      . Z      . F      .
         E      . Z      . F      . G      . J      . Z      .
         K      . N      . J      . F
OUTPUT = S1S2   . S2S3   . S3S4   . S4S3   . S3S5   . S5S6   .
         S6S8   . S8S3   . S3S9   . S9S10  . S10S12 . S12S13 .
         S13S15 . S15S2  . S2S3   . S3S4   . S4S3   . S3S5   .
         S5S6   . S6S8   . S8S3   . S3S5   . S5S7   . S7S8   .
         S8S3   . S3S9   . S9S14  . S14S15 . S15S2  . S2S16  .
         S16S17 . S17S19 . S19S20 . S20S21 . S21S23 . S23S24 .
         S24S25
SEQ =  2
INPUT  = A      . B      . C      . F      . D      . Z      .
         F      . G      . H      . Z      . J      . Z      .
         K      . N      . F
OUTPUT = S1S2   . S2S3   . S3S4   . S4S3   . S3S5   . S5S6   .
         S6S8   . S8S3   . S3S9   . S9S10  . S10S11 . S11S13 .
         S13S15 . S15S2  . S2S16  . S16S21 . S21S22 . S22S24 .
         S24S25
SEQ =  3
INPUT  = A      . B      . C      . F      . E      . Z      .
         F      . G      . J      . Z      . K      . L      .
         Z      . J      . O      . F
OUTPUT = S1S2   . S2S3   . S3S4   . S4S3   . S3S5   . S5S7   .
         S7S8   . S8S3   . S3S9   . S9S14  . S14S15 . S15S2  .
         S2S16  . S16S17 . S17S18 . S18S20 . S20S21 . S21S24 .
         S24S25
SEQ =  4
INPUT  = P
OUTPUT = S1S25
```

図 5 テストデータ生成例（図 4 に対応）
Fig. 5 Example of generated test data. (Corresponds to Fig. 4.)

らも参照されていないセグメントの占有場所を空き領域に編入する．

経路の被覆する遷移や出力等は，ビットベクトルとして表現する．入力や出力，状態，遷移等は，記号名称で入力し，内部処理においては番号として扱う．各外部入力に対しては，テストデータ生成の際に形を整えるための入力生成ルーチンを対応させる．

本プログラムは，正則でない状態遷移も扱えるように拡張してある．正則でない場合，出口への経路探索において，途中の必須通過点という副次目標の設定を行わない．

表 1 に，本プログラムによるテストデータ生成例を示す．詳細については，一例として，ケース C の状態遷移図を**図 4** にあげて説明する．その状態数は 25，遷移数は 35 であり，内部入力 $\alpha, \beta, \gamma, \delta, \theta, \varphi, \psi$ が 9 カ所で参照されている．図では，外部入力は a から n までの英字で示されている．本プログラムによって求めた入力系列と，それに伴う出力系列を**図 5** に示す．入力系列に対する経路をわかりやすく示すために，本例題では，状態 S_i から S_j への遷移に伴って，出力 S_iS_j が得られるものとしている．この例では，4 本の入力系列により，100% の網羅率が達成されている．計算時間は，HITAC M 180 の CPU 時間で 2.3 秒である．

6. あとがき

正則な状態遷移図に対しては，入力間の制約条件が内部出力と内部入力の受渡し関係として表され，入力は列挙型データに限るという制約条件をつけると，網羅的テストのためのテストデータ生成が自動化できることを示した．その計算時間は，状態数や遷移数が数十個であれば HITAC M 180 で数秒から数十秒であり，実用上問題ない範囲におさまる．ここに示したアルゴリズムは，必ずしも常時 100% の網羅率を達成できるものではないが，これを用いると，自動生成での未被覆部分に対してのみ人手でテストデータを作成すればよいので，テストデータ作成が容易になるばかりでなく，テスト漏れの防止に大きい効果がある．

本方式が適用可能な分野としては，電子交換機や通信，シーケンス制御のソフトウェアが挙げられよう．制限条件を緩和すれば適用範囲はさらに広まる．

謝辞 本アルゴリズムのプログラム化の半分，ならびに，反復遷移等についてのアルゴリズム改良は，元日立製作所システム開発研究所大沢恒春氏に行っていただいた．また，同社戸塚工場黒崎徹主任技師と山本利明主任技師，ならびに緒方秀夫副技師長には，有益な助言とご支援をいただいた．ここに深く感謝の意を表する．

参考文献

1) Miller, E. F., Jr.: Program Testing: Art Meets Theory, *IEEE Comput.*, Vol. 10, No. 7, pp. 42-51 (1977).
2) Howden, W. E.: Validation of Scientific Programs, *ACM Comput. Surv.*, Vol. 14, No. 2, pp. 191-227 (1982).
3) Clarke, L. A.: A System to Generate Test Data and Symbolically Execute Programs, *IEEE Trans. Softw. Eng.*, Vol. SE-2, No. 3, pp. 215-222 (1976).
4) Furukawa, Z. et al.: AGENT: Automatic Generation of Test Cases with Cause-Effect Graphs: Poster Session in 6 th International Conf. on Software Engineering, pp. 23-24 (Sep. 1982).
5) 古川, 他3名：ソフトウェアテスト項目作成支援システム AGENT-II の開発と評価, 情報処理学会ソフトウェア工学研究会資料, 28-8(1983. 2).
6) 稲村, 中野, 中西：テスト・データ自動生成の一方法, 情報処理学会ソフトウェア工学研究会資料, 25-2 (1982. 7).
7) Chow, T. S.: Testing Software Design Modeled by Finite-State Machines, *IEEE Trans. Softw. Eng.*, Vol. SE-4, No. 3, pp. 178-187 (1978).
8) Jessop, W. H. et al.: ATLAS—An Automated Software Testing System, Proc. of 2nd International Conf. on Software Engineering, pp. 629-635 (Oct. 1976).

(昭和59年2月14日受付)
(昭和59年5月15日採録)

付　録：テストデータ生成アルゴリズム（1）

```
------------------------------------------------------------------------
--                                                                    --
--  TEST DATA GENERATION FOR REGULAR STATE TRANSITION DIAGRAM.        --
--                                                                    --
------------------------------------------------------------------------
-- ALGORITHM TO GET THE SET OF INPUT SEQUENCES
--   FOR COVERING ALL TRANSITIONS OF A REGULAR STATE TRANSITION DIAGRAM
--     : --
 BEGIN
   GET A STATE TRANSITION DIAGRAM STD FROM INPUT FILE ;
   GET THE SET OF FEASIBLE ROUTES FOR COVERING ALL TRANSITIONS
     OF A STATE TRANSITION DIAGRAM STD ;
   GET THE SET OF INPUT SEQUENCES ALONG THE FEASIBLE ROUTES
     FOR COVERING ALL TRANSITIONS ;
 END ;
------------------------------------------------------------------------
-- GET THE SET OF FEASIBLE ROUTES FOR COVERING ALL TRANSITIONS
--   OF A STATE TRANSITION DIAGRAM STD : --
 BEGIN
   INITIALIZE TRANSITION COVERING LOOP
     BY SETTING THE FIRST WAVE CREST AT THE ENTRY STATE OF STD ;
   PROPAGATE THE WAVE CREST FROM ENTRY_STATE TO EXIT_STATE
     BY TRYING TO VISIT NONVISITED TRANSITIONS ;
   RAISE WAVE_CREST_FLAG ON THOSE STATES WHICH ARE THE TARGET OF
     ROUTE EXPRESSIONS HAVING UNCOVERED TRANSITIONS ;
   IF UNCOVERED TRANSITION IS LEFT  THEN
     PROPAGATE THE WAVE CRESTS FROM ENTRY_STATE TO EXIT_STATE
       BY SEARCHING FOR ROUTES WHICH REQUIRE LEAST ADDITIONAL
       INTERNAL INPUT ;
     WHILE (UNCOVERED TRANSITION IS LEFT) AND
           (COVERAGE COUNTER INCREASE) LOOP
       PROPAGATE THE WAVE CRESTS EXHAUSTIVELY FROM ENTRY_STATE TO
         EXIT_STATE IN THE ORDER OF NEAREST_TO_TAIL FIRST ;
       IF UNCOVERED TRANSITION IS LEFT THEN
         INSERT THE LOOP EXPRESSION WHICH INCLUDE SOME UNCOVERED
           TRANSITIONS OR SOME INTERNAL OUTPUT REQUIRED TO INVOKE
           UNCOVERED TRANSITIONS ;
       END IF ;
     END LOOP ;
   END IF ;
 END ;
------------------------------------------------------------------------
-- PROPAGATE THE WAVE CRESTS FROM HEAD10 TO TAIL10
--   BY TRYING TO VISIT NONVISITED TRANSITIONS:--
 BEGIN
   CHANGE_COUNT := 1 ;
   WHILE (CHANGE_COUNT > 0) AND
         (UNCOVERED TRANSITION IS LEFT BETWEEN HEAD10 AND TAIL10)  LOOP
     CHANGE_COUNT := 0 ;
     LET ICYCLE := ICYCLE + 1 ;
     TRY TO INVOKE TRANSITIONS BETWEEN HEAD10 AND TAIL10
         REQUIRING INTERNAL OUTPUT PRODUCED IN THE PREVIOUS CYCLE ;
     FOR EVERY NONVISITED TRANSITION T BETWEEN HEAD10 AND TAIL10
         HAVING WAVE CREST FLAG  LOOP
       TRY TO INVOKE THE TRANSITION T RETURNING SUCCESS_FLAG ;
       IF SUCCESS_FLAG = ON  THEN
         CHANGE_COUNT := CHANGE_COUNT + 1 ;
       END IF ;
     END LOOP ;
   END LOOP ;

 END ;
------------------------------------------------------------------------
-- TRY TO INVOKE THE TRANSITION T RETURNING SUCCESS_FLAG : --
 BEGIN
   IF TRANSITION T IS INVOKED BY AN EXTERNAL INPUT  THEN
     LET ROUTE_INF OF T := CONCATTRANS(ROUTE_INF(ORG(T)), T) ;
     SET INFORMATIONS OF THE INVOKED TRANSITION T ;
     SUCCESS_FLAG := ON ;
   ELSE   -- INVOKED BY SOME INTERNAL INPUT.
     TRY TO INVOKE THE TRANSITION T WHICH REQUIRE INTERNAL OUTPUT OI
       RETURNING SUCCESS_FLAG1 ;
     IF SUCCESS_FLAG1 = OFF THEN
       LINK T TO THE WAITING_TRANSITION_LIST OF OI WHICH INVOKES T ;
     END IF ;
     SUCCESS_FLAG := SUCCESS_FLAG1 ;
   END IF ;
 END ;
```

付　録：テストデータ生成アルゴリズム（2）

```
-----------------------------------------------------------------------
-- PROPAGETE THE WAVE CRESTS FROM HEAD30 TO TAIL30 BY SEARCHING FOR
--   ROUTES WHICH REQUIRE LEAST ADDITIONAL INTERNAL INPUT : --
 BEGIN
   CHANGE_COUNT2 := 1 ;
   WHILE (CHANGE_COUNT2 > 0) AND
         (UNCOVERED TRANSITION IS LEFT BETWEEN HEAD30 AND TAIL30)  LOOP
     CHANGE_COUNT2 := 0 ;
     LET ICYCLE := ICYCLE + 1 ;
     TRY TO INVOKE TRANSITIONS REQUIRING INTERNAL OUTPUT
       PRODUCED IN THE PREVIOUS CYCLE ;
     FOR EVERY STATE S BETWEEN HEAD30 AND TAIL30
                       HAVING WAVE_CREST_FLAG  LOOP
       FOR EVERY TRANSITION T OUTGOING FROM S
           AND HAVING WAVE_CREST_COUNT GREATER THAN 0  LOOP
         IF T IS INVOKED BY AN EXTERNAL INPUT  THEN
           LET ROUTE_INF OF T := CONCATTRANS(ROUTE_INF(ORG(T)), T) ;
           SET INFORMATIONS OF THE INVOKED TRANSITION T ;
           LET CHANGE_COUNT2 := CHANGE_COUNT2 + 1 ;
         ELSIF (PRODUCED_OUTPUT_LIST(S) CONTAINS THE INTERNAL OUTPUT OI
                WHICH INVOKES THE TRANSITION T)  OR
               (S HAS NO TRANSITION INVOKED BY EXTERNAL INPUT)  THEN
           TRY TO INVOKE THE TRANSITION T WHICH REQUIRE
             INTERNAL OUTPUT OI RETURNING SUCCESS_FLAG2 ;
           IF SUCCESS_FLAG2 = ON  THEN
             LET CHANGE_COUNT2 := CHANGE_COUNT2 + 1 ;
           ELSE
             LINK T TO THE WAITING_TRANSITION_LIST OF OI
                    WHICH INVOKES T ;
           END IF ;
         END IF ;
     END LOOP ;
   END LOOP ;
 END ;
-----------------------------------------------------------------------
 -- PROPAGATE THE WAVE CRESTS EXHAUSTIVELY FROM HEAD40 TO TAIL40
 --   IN THE ORDER OF NEAREST_TO_TAIL FIRST : --
  BEGIN
    BUILD THE EXAMINATION_STATE_LIST BY EXTRACTING SUCH STATES AS
      HAVING WAVE_CREST_FLAG AND RESIDING IN BETWEEN HEAD40 AND TAIL40 ;
    LET CHANGE_COUNT3 := 1 ;
    WHILE (CHANGE_COUNT3 > 0) AND (UNCOVERED TRANSITION IS LEFT
             BETWEEN HEAD40 AND TAIL40)  LOOP

      LET CHANGE_COUNT3 := 0 ;
      LET ICYCLE := ICYCLE + 1 ;
      REORDER THE EXAMINATION_STATE_LIST IN THE ORDER OF
        SMALLEST_NUMBER_OF_INEVITABLE_STATES_TO_TAIL FIRST ;
      RESET NEXT_EXAMINATION_STATE_LIST ;
      TRY TO INVOKE TRANSITIONS BETWEEN HEAD40 AND TAIL40
          REQUIRING INTERNAL OUTPUT PRODUCED IN THE PREVIOUS CYCLE ;
      FOR EVERY STATE S IN THE EXAMINATION_STATE_LIST  LOOP
      FOR EVERY TRANSITION T OUTGOING FROM S  LOOP
        IF T IS INVOKED BY AN EXTERNAL INPUT  THEN
          LET ROUTE_INF OF T := CONCATTRANS(ROUTE_INF(ORG(T)), T) ;
          SET INFORMATIONS OF THE INVOKED TRANSITION T ;
          LET CHANGE_COUNT3 := CHANGE_COUNT3 + 1 ;
        END IF ;
      END LOOP ;
      FOR EVERY TRANSITION T OUTGOING FROM S  LOOP
        IF T IS INVOKED BY AN INTERNAL OUTPUT  THEN
          EXHAUSTIVELY TRY TO INVOKE THE TRANSITION T WHICH REQUIRE
            AN INTERNAL OUTPUT OI RETURNING SUCCESS_FLAG3 ;
          IF SUCCESS_FLAG3 = ON  THEN
            CHANGE_COUNT3 := CHANGE_COUNT3 + 1 ;
          ELSE
            LINK T TO THE WAITING_TRANSITION_LIST OF OI WHICH INVOKES T;
          END IF ;
        END IF ;
      END LOOP ;
      FOR EVERY UNCOVERED TRANSITION T ON WAVE CRESTS  LOOP
        SEARCH FOR ROUTES FROM TRANSITION T TO STATE EXIT_STATE ;
      END LOOP ;
      LET EXAMINATION_STATE_LIST := NEXT_EXAMINATION_STATE_LIST ;
      FOR EVERY STATES IN NEXT_EXAMINATION_STATE_LIST  LOOP
        IF PRODUCED_OUTPUT_VECTOR(S)
                        = PREVIOUSLY_PRODUCED_OUTPUT_LIST(S)  THEN
        ERASE ELEMENT S OF EXAMINATION_STATE_LIST ;
        END IF ;
      END LOOP ;
    END LOOP ;
  END ;
-----------------------------------------------------------------------
```