AGENT：機能テストのためのテスト項目作成の一手法†

古川善吾†† 野木兼六†† 徳永健司†††

ソフトウェアの機能テストのためのテスト項目を系統的に作成するAGENT技法を提案した．AGENT技法は，機能図式（Function Diagram）という形式的な記法によってソフトウェアの機能仕様を表現した後，機械的にテスト項目を作成する技法である．機能図式は，入力や出力の順序に依存した対応関係を表す状態遷移（状態遷移図で記述する）と，状態遷移の各状態での入力データと出力データや遷移先状態との対応関係を表す論理関係（原因結果グラフあるいは決定表で記述する）とから成っている．AGENT技法では，この機能図式から，通過すべき状態の列と各状態での入出力データの条件の組合せとして，以下の条件を満たすテスト項目を機械的に作成する．①各状態での入出力データの条件を確認するのに十分である．②状態遷移を構造化した構造化状態遷移の各遷移を少なくとも1回は辿る．③構造化状態遷移の繰返しは0回と1回の2通りを実現する．このテスト項目作成を自動的に行うためにAGENTプログラムを開発した．本論文では，テスト項目作成の考え方，AGENTプログラムの概要について述べた．

1. はじめに

本研究の目的は，ソフトウェアの機能テストのためのテスト項目を系統的に作成する技法を与えることである．われわれは，機能仕様を形式的に記述するための機能図式記法[11]と自動的にテスト項目を作成するプログラムAGENT (Automated Generation System of Test-cases) を開発した[12]．

機能図式は，状態遷移と論理関係を組み合わせた記法である．入力を行う順序や変換の順序などによって出力データが変わることを状態遷移で表し，ある状態での入力データについての条件と出力データとの対応関係を論理関係で表す．この機能図式から，AGENTプログラムは通るべき状態の列と各状態での入出力データが満たすべき条件の組合せとしてテスト項目を作成する．

本論文では次章以下で，機能図式，テスト項目作成方式，AGENTプログラムについて論じる．本章では以下，テスト法について述べる．

プログラムのテストの際に用いるデータ（テストデータ）をいかに選択するかは，テストにおける重要な課題であり，数多くの研究がなされてきた[4),6),9)]．すなわち，テストデータ選択の基準が信頼でき確かであるなら，テストによってプログラムの正しさを示すことができる[4)]．しかし，一般的には実用的な基準はなく[6),9)]，発見できる誤りを限定するか[6)]基準に対する条件を緩めることが必要である[9)]．

一方，上記の研究とは別に，ソフトウェアのテストを効率化し系統立てるために，テストデータを機械的に選択する研究も進められてきた[1)-3)]．Elmendorf[3)]は，入力データに対する条件と出力データの対応を論理関係で結びつけた原因結果グラフを用いて記述した機能仕様からテストデータが満たすべき条件の組合せと確認すべき出力データをテスト項目として作成した．Chow[1)]は，有限状態機械で制御の流れをモデル化した後テストデータを作成した．また，Duncanら[2)]は，属性文法をもとにして，属性を利用して自動的にテストデータを作成する技法を提案した．

これらの方法に対し，われわれの方法は以下のような特徴がある．

（1）状態遷移と各状態における入力と出力および遷移先状態の間の論理関係を用いているので，有限状態機械や生成規則によって機能を記述するより多様なソフトウェアの機能が記述できる．

（2）入力データや変換の順序に依存した出力データの変化を状態遷移で記述できるので，原因結果グラフに比較しソフトウェアの機能を容易に記述できる．

（3）状態の列と入出力データの条件とが結合されたテスト項目を機械的に作成できる．

2. 機能図式

ソフトウェアの機能仕様，すなわち，入力データと出力データの対応関係，の記述のための研究が数多く

† AGENT: A Test Cases Generation Method for Functional Testing by ZENGO FURUKAWA, KENROKU NOGI (Systems Development Laboratory, Hitachi, Ltd.) and KENJI TOKUNAGA (Software Works, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所
††† (株)日立製作所ソフトウェア工場

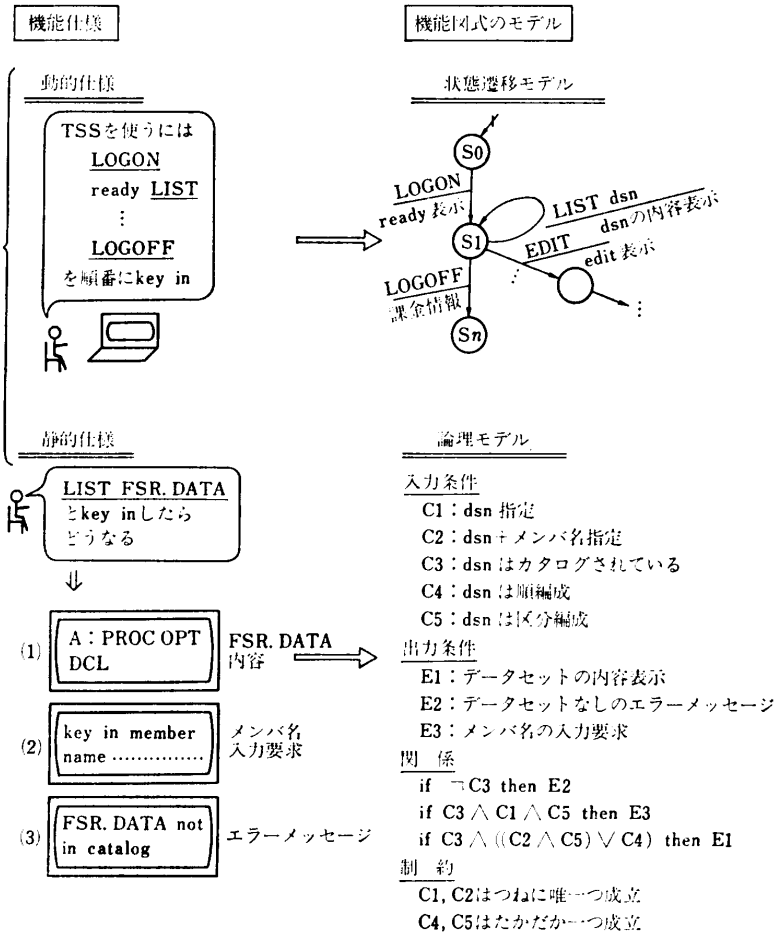


図 1 機能仕様のモデル化
Fig. 1 Model of a functional specification.

なされてきている．原因結果グラフは[3),7)]，入力データの条件と出力データの条件の対応を論理関係で，また入力データの間の依存関係を制約条件で記述したものである．

ソフトウェアの機能仕様を記述するとき，**図1**に示すように動的な仕様（入力データの入力順序，変換の順序などによって出力データが決まる）と静的な仕様（入力データに対する条件の組合せによって出力データが決まる）を分けて表現することが多い．静的な仕様だけでは，場合の数が多くなりすぎて書ききれないため，時系列的な考えを導入して記述を簡略化する．動的な仕様を表すには，入力データとそのときの状態によって出力データと状態が決まる状態遷移モデルが適している．一方静的な仕様を表すには，入力データの条件の組合せによって出力データを決める論理モデルが適している．

機能図式では，入力データの入力順序をもとにした状態遷移モデルと，各状態における入力データと出力データおよび遷移先状態の対応を表すために論理モデルを用いてソフトウェアの機能仕様を表す．

2.1 機能図式記法

機能図式を記述するために**図2**に示す記法を用いる．

（1）状態遷移部

状態遷移を記述するために，状態と遷移を用いる．

（a）状　態

状態はデータの入力が行われることを示している．初期状態はそこからプログラムが動作を始めることを示し，終了状態は，そこでプログラムの動作が停止する可能性のあることを示す．

（b）遷　移

遷移は，状態の移行を示すものである．論理遷移は，論理関係Tによって入力データと出力データおよび遷移先状態が決まることを示す．論理遷移で，互いに排他的でどれかは必ず成立する入力データの条件 IC_i によって出力データ O と遷移先が決まるときは，条件遷移で記述することができる．さらに，状態からの遷移が一つでその状態に達したときは必ずその遷移が起こることを無条件遷移で記述する．

（2）論理関係部

論理関係を記述するために，入力データと出力データおよび遷移先状態の対応関係を表す，原因結果グラフあるいは決定表を用いる．また，入力データの条件の間の依存関係を表すために制約条件を用いる．

2.2 機能図式例

図3は，次のような機能仕様をもつ「現金自動支払い機」の機能図式である．

現金自動支払い機は，つねにカードが入力されるのを待っており，カードが入力されると‘暗証番号入力’というメッセージを出す．暗証番号が入力されると，登録されているものと一致するか調べ一致していれば，‘金額入力’というメッセー

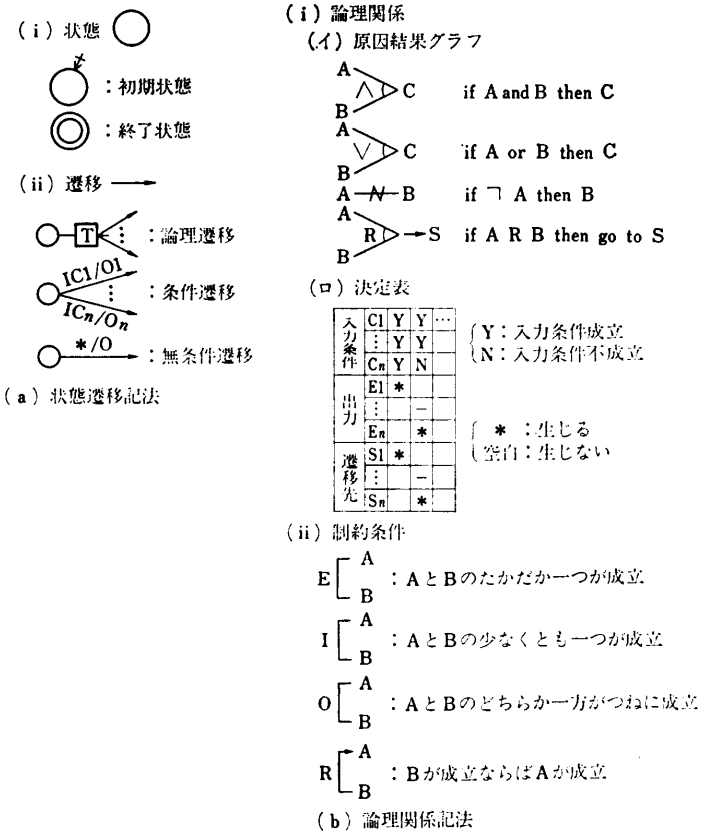


図 2 機能図式記法
Fig. 2 Functional Diagram notation.

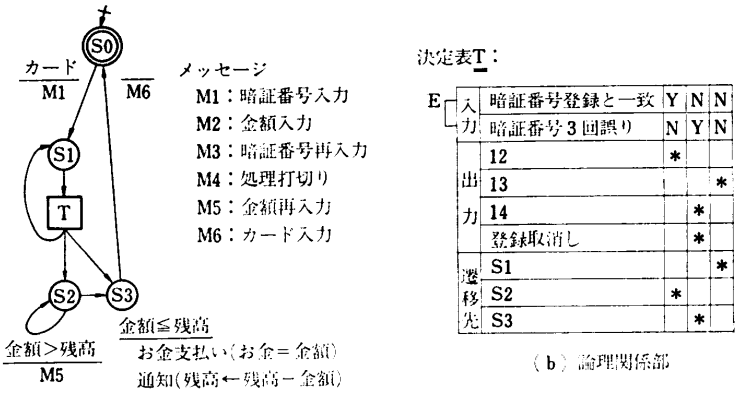


図 3 機能図式の例（現金自動支払い機）
Fig. 3 An example of FD (ATM).

ジを出す．もし，一致していなければ，不一致が3回目かどうかを調べ，3回目であれば‘処理打切り’というメッセージを出し，カードの登録を取り消し，さらに，‘カード入力’というメッセージを出し，次のカードが入力されるのを待つ．不一致が，1回目，2回目のとき，‘暗証番号再入力’というメッセージを出して，暗証番号が入力されるのを待つ．金額が入力されたとき，入力された金額と残高を比較し，金額のほうが大きいときは，‘金額再入力’というメッセージを出して再度金額が入力されるのを待つ．金額が残高より大きくないときは，金額で指定されたお金と残高を金額分減らした通知を出し，‘カード入力’というメッセージを出し次のカードが入力されるのを待つ．

図3の機能図式では，状態S3からの遷移が無条件遷移になっているが，これは，暗証番号を3回誤ったときとお金が支払われたとき，どちらも入力はないが出力および遷移先状態が同じであるために，‘カード入力’というメッセージの出力を1回だけ書けばすむように設けた状態である．このように，無条件遷移は記述量を減らすために用いることができる．

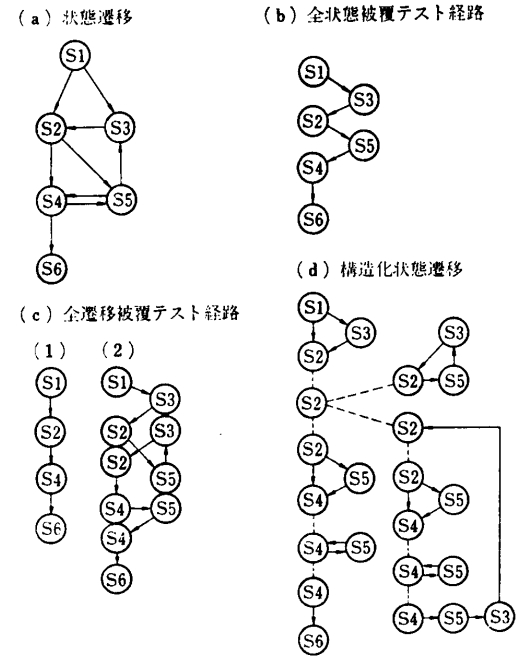
3. テスト項目作成の方式

機能図式からテスト項目を作成するに当たっては，テスト項目の数が実用的であること，作成されたテスト項目がどういう基準を満たしているかが明確であることが重要である．機能図式は状態遷移と論理関係からなっている．論理関係については，論理回路のテストパターン作成アルゴリズム[8]を用いたElmendorf[3]の原因結果グラフ法が知られており，われわれは先にその方法に基づくプログラムを開発した[10]．状態遷移については，プログラムの制御の流れに基づいたテストの基準が知られており[5]，状態遷移の状態を節点，遷移を分岐として同様のテスト基準が考えられる．

状態遷移のテスト項目と論理関係のテスト項目を合成して機能図式のテスト項目とすることにより，実用的で基準が明確なテスト項目を作成した．

3.1 テスト項目作成の基準

機能図式の論理関係部に対してElmendorfの考え



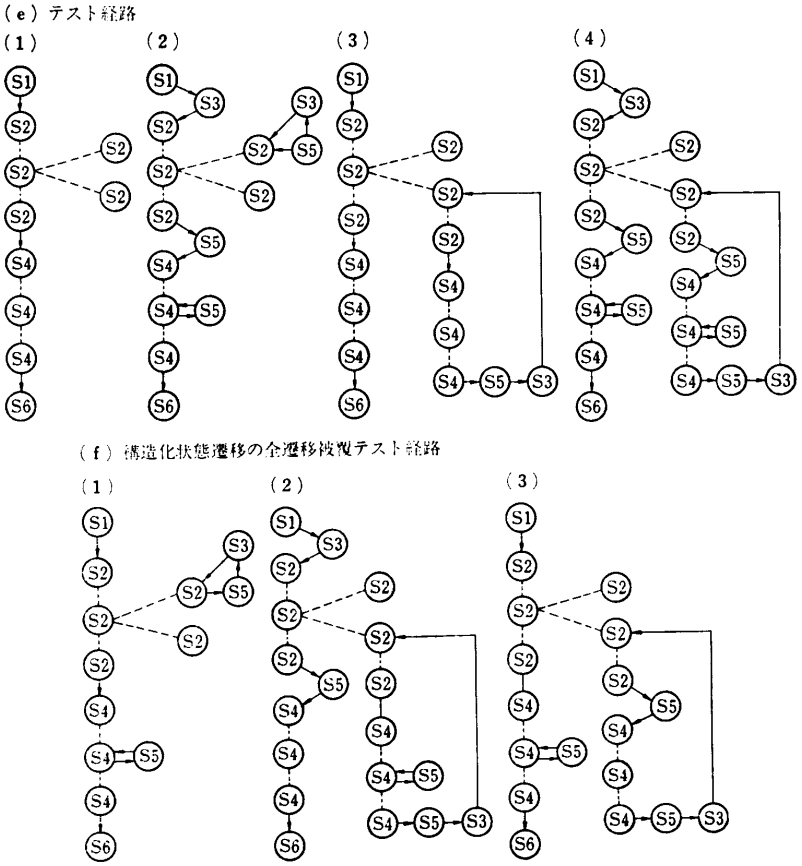
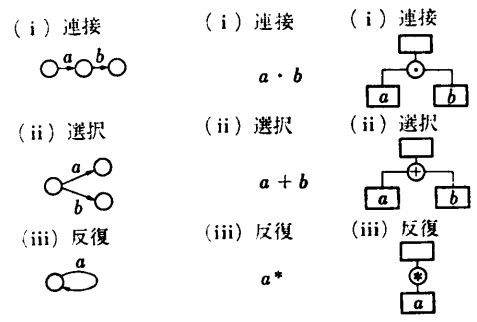


図 4 状態遷移のテスト経路の例
Fig. 4 An example of testing paths.



(a) 状態遷移表現 (b) 正規表現 (c) 木表現

図 5 構造化状態遷移の表記法
Fig. 5 Notation for a structured state transition.

方に従ってテスト項目を作成したとき，テスト項目は入出力条件の成立・不成立を確認するのに十分で，最小数に近いものである．テスト項目の数は入出力条件の数に比例することが知られている．

一方，状態遷移部に対しては，各種の基準があるが，全状態通過や全遷移通過がよく知られている．たとえば図4の(a)の状態遷移図に対して，全状態を通過するための状態の列（これを‘テスト経路’と呼ぶ）の例として図4(b)が考えられ，全遷移を通過するためのテスト経路の例として(c)の(1)(2)が考えられる．プログラムでよく経験する誤りとして，ループを通らなかった場合を考え落としたり，ループの繰返しの終了条件を誤るなどがある．このことから，状態遷移でもループを1回も通らないテスト経路と1回以上繰り返した後終了するテスト経路があることが望ましい．しかし，図4の(c)では (S2, S4, S5, S3, S2) というループが1回も繰り返されておらず，全遷移通過という基準を満たしていてもループについて0回と1回以上の繰返しを実現しない場合があることがわかる．

一般の状態遷移図でループを認識することは困難な問題である．一方，図5の(a)に示す連接，選択，反復の三つの組合せだけで構成した構造化状態遷移であれば，

ループは反復で生じるだけであり容易に認識できる．たとえば，図4(a)のの状態遷移の構造化の一例として(d)に示す構造化状態遷移が考えられる．ただし，図4の(d)の点線は，選択と反復の入口と出口を明示するために状態を複写したことを示している．

「構造化状態遷移の全遷移を少なくとも1回は通り，ループについて0回と1回の2通りを実現する」をテスト経路作成の基準として設定した．図4の(e)に，この基準を満たすテスト経路の一例を示す．テスト経路作成の基準のループについての部分がないと，図4の(f)に示すテスト経路によっても基準が満たされてしまい，S2のループを1回も通らない場合のテスト経路が実現されなくてもよくなってしまう．

3.2 テスト項目作成手順

機能図式から以下の手順で機能図式のテスト項目を作成する．

(1) 部分テスト項目の作成

各状態ごとに，入力データについての条件を原因，出力データおよび遷移先状態を結果として，論理関係が決定表で与えられている場合は等価な原因結果グラフに変換した後，論理関係のテスト項目（これを‘部分テスト項目’と呼ぶ）を，Elmendorf の方法を用いて作成する．部分テスト項目は，原因の成立・不成立の組合せと，それに対する結果の成立・不成立の組合せから成る．部分テスト項目で遷移先状態に対応した結果は，つねに唯一つ成立していなければならない．

(2) テスト経路の作成

まず，状態遷移を，図5の(a)に示す三つの要素のみで構成された構造化状態遷移に変換する．この変換は，状態遷移をそれと等価な図5(b)に示す正規表現に直す[3]ことによって行われる．その後，初期状態から始めて，終了状態に至るテスト経路をテスト経路作成の基準を満たすように作成する．

(3) テスト項目の合成

機能図式のテスト項目（本章では，これ以降，まぎらわしくない場合，たんに，‘テスト項目’と呼ぶことがある）は，初期状態から終了状態までに通過すべき状態の列と，各状態での入力データについての条件の成立・不成立および出力データの出力または不出力の組合せ，として作成する．そのために，各テスト経路の各状態に，その状態の部分テスト項目の中で遷移先の状態とテスト経路の次の状態が一致するものを，その状態の条件として割り当てる．

テスト経路の作成とテスト項目の合成とのアルゴリズムは次節で述べる．

3.3 テスト項目の合成アルゴリズム

構造化状態遷移は，前節で述べたように正規表現で表されている．テスト項目を合成する準備として，正規表現を，図5の(c)の木表現によって表現しておく．この木表現を条件構造木と呼ぶことにする．条件構造木の例として「現金自動支払い機」の条件構造木を**図6**に示す．テスト項目の合成は以下のステップで行う．

ステップ1

各状態の部分テスト項目で遷移先状態が同じものの個数を数え，条件構造木の対応した葉（遷移）にその個数をテスト項目数として設定する．

ステップ2

条件構造木の各節点をポストオーダで辿りながら，**図7**に示す規則に従って各節点でのテスト項目数を求めて設定する．

ステップ3

1番目から条件構造木の根のテスト項目数番目まで順番にテスト項目を合成する．その際，各節点における n 番目のテスト項目は，演算子の種類に応じて以下

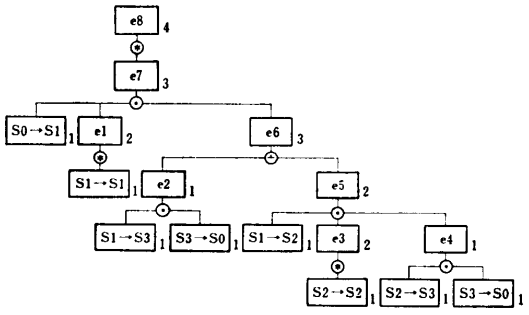


図6 条件構造木の例
Fig. 6 An example of the condition structure tree.

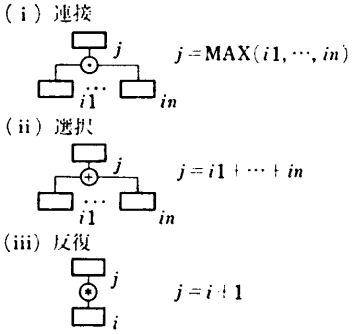


図7 テスト項目数数え上げ規則
Fig. 7 The rules for counting the number of test cases.

の規則によって取り出す．

（a）連　接

各子節点ごとに i 番目のテスト項目を取り出してつなぐ．ただし，i は次式を満たす．

$$i=\mathrm{MOD}\ (n-1,\ \text{子節点のテスト項目数})+1 \quad (1)$$

（b）選　択

j 番目の子節点の k 番目のテスト項目を取り出す．ただし，j,k は次式を満たす．

$$i=\mathrm{MOD}\ (n-1,\ \text{子節点のテスト項目数})+1 \quad (2)$$

$$\sum_{l=1}^{j-1}(l\ \text{番目の子節点のテスト項目数})<i \leq \sum_{l=1}^{j}(l\ \text{番目の子節点のテスト項目数}) \quad (3)$$

$$k=i-\sum_{l=1}^{j-1}(l\ \text{番目の子節点のテスト項目数}) \quad (4)$$

（c）反　復

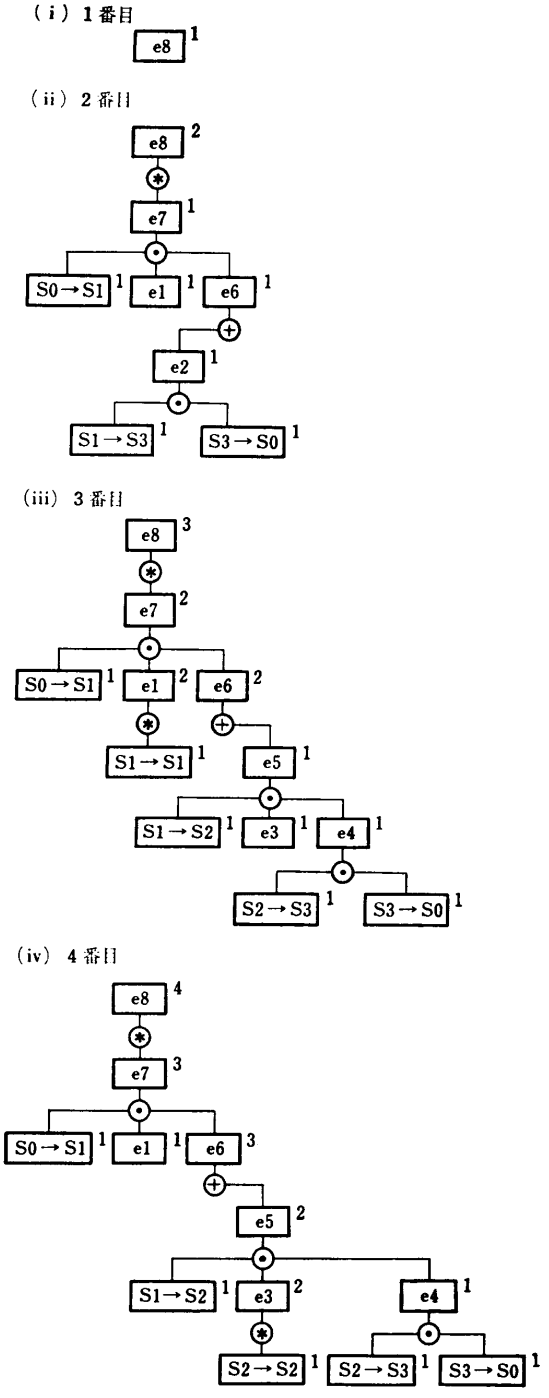
子節点の i 番目のテスト項目を取り出す．ただし i が0のときは空である．

$$i=\mathrm{MOD}\ (n-1,\ \text{子節点のテスト項目数}) \quad (5)$$

（d）葉

遷移元の状態と n 番目の部分テスト項目を取り出す．

図6の条件構造木には，各節点におけるテスト項目数を各節点を表す箱の右下の添字としてつけてある．この「現金自動支払い機」の例では四つのテスト項目



（a）条件構造木の探索

（i）テスト項目1

ϕ（空）

（ii）テスト項目2

S0　（カード入力）
↓
S1　¬（暗証番号登録と一致）∧（暗証番号3回誤り）
↓
S3　（＊）
↓
S0

（iii）テスト項目3

S0　（カード入力）
↓
S1　¬（暗証番号登録と一致）∧¬（暗証番号3回誤り）
↓
S1　（暗証番号登録と一致）∧¬（暗証番号3回誤り）
↓
S2　（金額≦残高）
↓
S3　（＊）
↓
S0

（iv）テスト項目4

S0　（カード入力）
↓
S1　（暗証番号登録と一致）∧¬（暗証番号3回誤り）
↓
S2　（金額＞残高）
↓
S2　（金額≦残高）
↓
S3　（＊）
↓
S0

（b）テスト項目

図8　テスト項目合成例

Fig. 8　An example of test case synthesis.

がある．各テスト項目は，図8の(a)に示すように条件構造木が辿られ，(b)に示すようなテスト項目が得られる．

4. AGENT プログラム

AGENT-Ⅱ システムは，機能図式からテスト項目を自動的に作り出す（原因結果グラフからテスト項目を作成するプログラムを AGENT システムと呼んだので，区別のため一応 AGENT-Ⅱ と呼んでおく）．

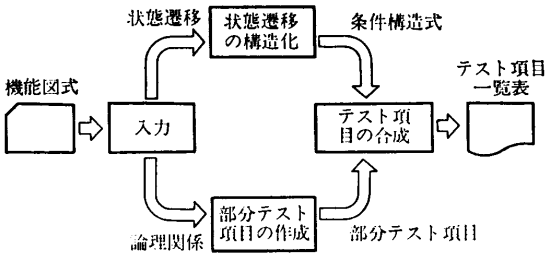


図 9 AGENT-Ⅱ システム構成
Fig. 9 The components of AGENT-Ⅱ system.

```
**  SOURCE LIST  **
SEQ.    SOURCE LIST
     ---------1---------2---------3---------4-------
  1 TITLE  CD = 'ｹﾞﾝｷﾝｼﾞﾄﾞｳｼﾊﾗｲｷ'
  2 STATE  (S0)
  3   NODE  CAUSE  C0 = 'ｷｱﾂｼｭ ｶｰﾄﾞ'
  4         EFFECT  M1 = 'ｱﾝｼｮｳ ﾊﾞﾝｺﾞｳ ﾆｭｳﾘｮｸ'
  5   DECISION
  6         C01 = (C0)/M1->S1
  7   CONST
  8                U01 = C0 ONLY
  9 STATE  (S1)
 10   NODE  CAUSE  AA = 'ｱﾝｼｮｳ ﾊﾞﾝｺﾞｳ ﾄｳﾛｸｽﾞﾐ'
 11                AB = 'ｱﾔﾏﾘ = 3 ｶｲ'
 12         EFFECT M2 = 'ｷﾝｶﾞｸﾆｭｳﾘｮｸ'
 13                M3 = 'ｱﾝｼｮｳﾊﾞﾝｺﾞｳ ｻｲﾆｮｳﾘｮｸ'
 14                M4 = 'ｼｮﾘｳﾁｷﾘ'
 15                OC1 = 'ｶｰﾄﾞ ﾄｳﾛｸﾄﾘｹｼ'
 16   DECISION
 17                C11 = (AA NOT AB)/M2->S2
 18                C12 = (NOT AA AB)/M4 OC1->S3
 19                C13 = (NOT AA NOT AB)/M3->S1
 20   CONST
 21                U11 = AA AB EXCLUSIVE
 22 STATE  (S2)
 23   NODE  CAUSE  KA = 'ｷﾝｶﾞｸ <= ｻﾞﾝﾀﾞｶ'
 24                KB = 'ｷﾝｶﾞｸ > ｻﾞﾝﾀﾞｶ'
 25         EFFECT M21 = 'ｷﾝｶﾞｸ ｻｲﾆｮｳﾘｮｸ'
 26                FR = 'ﾌｳﾁｮｳ ﾊﾂｺｳ'
 27                FR1 = 'ｻﾞﾝﾀﾞｶ <- ｻﾞﾝﾀﾞｶｰｷﾝｶﾞｸ'
 28                OM = 'ｵｶﾈ ｼﾊﾗｲ'
 29                OM1 = 'ｵｶﾈ = ｷﾝｶﾞｸ'
 30   RELATE
 31         R1 = KA SIMP FR
 32         R2 = KA SIMP FR1
 33         R3 = KA SIMP OM
 34         R4 = KA SIMP OM1
 35         R5 = KA SIMP ->S3
 36         R6 = KB SIMP M21
 37         R7 = KB SIMP ->S2
 38   CONST
 39         U21 = KA KB  ONLY
 40 STATE  (S3)
 41   NODE  EFFECT  M7 = 'ｶｰﾄﾞ ﾆｭｳﾘｮｸ'
 42   DECISION
 43         C31 = (*)/M7->S0
 44 INITIAL  S0
 45 FINAL    S0
 46 END
```

図 10 ソースリストの例
Fig. 10 An example of the source list.

TRANSITION		K	NODE	MEANING	* テスト * ケース * ID *				
TAIL	HEAD	N				1	2	3	4
S0	S1	I	C0	ｷｱﾌｼｭ ｶｰﾄﾞ		O	O	O	
		O	M1	ｱﾝｼｮｳ ﾊﾞﾝｺﾞｳ ﾆｭｳﾘｮｸ		O	O	O	
S1	S1	I	AA	ｱﾝｼｮｳ ﾊﾞﾝｺﾞｳ ﾄｳﾛｸｽﾞﾐ			X		
			AB	ｱﾔﾏﾘ = 3 ｶｲ			X		
		O	M2	ｷﾝｶﾞｸﾆｭｳﾘｮｸ			X		
			M3	ｱﾝｼｮｳﾊﾞﾝｺﾞｳ ｻｲﾆｮｳﾘｮｸ			O		
			M4	ｼｮﾘｳﾁｷﾘ			X		
			OC1	ｶｰﾄﾞ ﾄｳﾛｸﾄﾘｹｼ			X		
S1	S3	I	AA	ｱﾝｼｮｳ ﾊﾞﾝｺﾞｳ ﾄｳﾛｸｽﾞﾐ		X			
			AB	ｱﾔﾏﾘ = 3 ｶｲ		O			
		O	M2	ｷﾝｶﾞｸﾆｭｳﾘｮｸ		X			
			M3	ｱﾝｼｮｳﾊﾞﾝｺﾞｳ ｻｲﾆｮｳﾘｮｸ		X			
			M4	ｼｮﾘｳﾁｷﾘ		O			
			OC1	ｶｰﾄﾞ ﾄｳﾛｸﾄﾘｹｼ		O			
S3	S0	I	*			O			
		O	M7	ｶｰﾄﾞ ﾆｭｳﾘｮｸ		O			
S1	S2	I	AA	ｱﾝｼｮｳ ﾊﾞﾝｺﾞｳ ﾄｳﾛｸｽﾞﾐ			O	O	
			AB	ｱﾔﾏﾘ = 3 ｶｲ			X	X	
		O	M2	ｷﾝｶﾞｸﾆｭｳﾘｮｸ			O	O	
			M3	ｱﾝｼｮｳﾊﾞﾝｺﾞｳ ｻｲﾆｮｳﾘｮｸ			X	X	
			M4	ｼｮﾘｳﾁｷﾘ			X	X	
			OC1	ｶｰﾄﾞ ﾄｳﾛｸﾄﾘｹｼ			X	X	
S2	S2	I	KA	ｷﾝｶﾞｸ <= ｻﾞﾝﾀﾞｶ				X	
			KB	ｷﾝｶﾞｸ > ｻﾞﾝﾀﾞｶ				O	
		O	M21	ｷﾝｶﾞｸ ｻｲﾆｮｳﾘｮｸ				O	
			FR	ﾌｳﾁｮｳ ﾊﾌｺｳ				X	
			FR1	ｻﾞﾝﾀﾞｶ <- ｻﾞﾝﾀﾞｶｰｷﾝｶﾞｸ				X	
			OM	ｵｶﾈ ｼﾊﾗｲ				X	
			OM1	ｵｶﾈ = ｷﾝｶﾞｸ				X	
S2	S3	I	KA	ｷﾝｶﾞｸ <= ｻﾞﾝﾀﾞｶ			O	O	
			KB	ｷﾝｶﾞｸ > ｻﾞﾝﾀﾞｶ			X	X	
		O	M21	ｷﾝｶﾞｸ ｻｲﾆｮｳﾘｮｸ			X	X	
			FR	ﾌｳﾁｮｳ ﾊﾌｺｳ			O	O	
			FR1	ｻﾞﾝﾀﾞｶ <- ｻﾞﾝﾀﾞｶｰｷﾝｶﾞｸ			O	O	
			OM	ｵｶﾈ ｼﾊﾗｲ			O	O	
			OM1	ｵｶﾈ = ｷﾝｶﾞｸ			O	O	
S3	S0	I	*				O	O	
		O	M7	ｶｰﾄﾞ ﾆｭｳﾘｮｸ			O	O	

図 11 テスト項目一覧表の例
Fig. 11 An example of the test-case table.

AGENT-Ⅱ システムの構成を**図 9** に示す．記述言語は PL/I であり，約 16k 行の大きさである．

「現金自動支払い機」のソースリストとテスト項目の出力を**図 10**，**図 11** に示す．図 7 の各サブシステムの概要は以下のとおりである．

（1）入　力

テキスト形式の機能図式を入力し，状態遷移部と論理関係部に分ける．図 10 に示すように機能図式の記述は，表題文(TITLE)，状態文(STATE)，初期状態文(INITIAL)，終了状態文(FINAL)，終了文(END)で行う．状態文において，論理関係を記述するため

に，条件定義 (NODE)，関係定義 (RELATE) あるいは決定表定義 (DECISION) および，制約条件定義 (CONST) を用いる．

（2）状態遷移の構造化

有限オートマトンを等価な正規式に変換することにより，状態遷移から構造化状態遷移を示す．条件構造式を作成する．

（3）部分テスト項目の作成

Elmendorf の方法により，各状態ごとに論理関係から部分テスト項目を作成する．

（4）テスト項目の合成

3.2節で述べたアルゴリズムに従って，条件構造式と部分テスト項目から機能図式のテスト項目を合成する．テスト項目は，機能図式の状態遷移の全遷移を少なくとも1回は通り，ループについては0回と1回の繰返しの2通りを実現し，論理関係の入力データに対する条件の成立・不成立と出力データの出力を確認するのに十分で最小に近いものである．

図11のテスト項目一覧表は，図10に示した「現金自動支払い機」のソーステキストから作成されたものである．遷移 (TRANSITION) は，遷移元の状態 (TAIL) と遷移先の状態 (HEAD) から構成されている．遷移元の状態での条件には，入力データに対する条件 (I) と出力データ (O) がある．各テスト項目は列として示されており，空白はその遷移を通らないことを示し，'O' は成立を，'X' は不成立を示す．図10のテスト項目1は，どの遷移も通らないことを示しているが，「現金自動支払い機」では初期状態 (S0) と終了状態 (S0) が一致しているために，ループを1回も通らないテスト項目として作成された．

テスト項目2は，状態 S0, S1, S3 を通り状態 S0 に達する．各状態での入力データの条件と確認すべき出力が示されている．たとえば，状態 S1 では登録ずみでない暗証番号を入力し，かつ，誤りを3回繰り返したとき，'金額入力' や '暗証番号再入力' メッセージが出されずに，'処理打切り' メッセージを出し，カードの登録が取り消されることを確認する．

AGENT-Ⅱ システムは，この「現金自動支払い機」の処理に，HITAC M-180/VOS 3 の下で約6.8秒を要した．

5. おわりに

AGENT 技法は，ソフトウェアが実現しようとしている機能から基準の明確なテスト項目を機械的に作成しており，テストの系統化の基礎を与えたと考えている．このテスト法だけでテストは完了するのではなく，テスト実施時の被覆率の測定によるテスト十分性の評価や，誤りを発見しやすいテストデータの選択[6]など他のテスト法との有機的な結合が必要である．

今後の課題としては，機能設計段階で機能仕様を記述する言語として機能図式を整備することと，テスト項目だけでなく実際のテストデータを作成して[2]テスト実施の効率化を図ることなどが考えられる．

参考文献

1) Chow, T. S.: Testing Software Design Modeled by Finite-State Machines, *IEEE Trans. Softw. Eng.*, Vol. SE-4, No. 3, pp. 178-187 (1978).
2) Duncan, A. G. and Hutchison, J. S.: Using Attributed Grammars to Test Designs and Implementations, Proc. of 5th ICSE, pp. 170-178 (1981).
3) Elmendorf, W. R.: Functional Analysis Using Cause-Effect Graphs, Proc. of SHARE XLIII, N. Y., SHARE (1974).
4) Goodenough, J. P. et al.: Toward a Theory of Test Data Selection, *SIGPLAN NOTICES*, Vol. 10, No. 6, pp. 493-510 (1975).
5) Howden, W. E.: Reliability of Path Analysis Testing Strategy, *IEEE Trans. Softw. Eng.*, Vol. SE-2, No. 3, pp. 208-214 (1976).
6) Howden, W. E.: Functional Program Testing, *IEEE Trans. Softw. Eng.*, Vol. SE-6, No. 2, pp. 162-169 (1980).
7) Myers, G. J.: *The Art of Software Testing*, John Wiley & Sons Inc., New York (1979).
8) Roth, J. P.: Diagnosis of Automata Failures: A Calculus and A Method, *IBM J. Res. Dev.*, Vol. 10, No. 4, pp. 278-294 (1966).
9) Weyuker, E. J. et al.: Theories of Program Testing the Application of Revealing Subdomains, *IEEE Trans. Softw. Eng.*, Vol. SE-6, No. 3, pp. 236-246 (1980).
10) 古川他：ソフトウェアテスト項目作成支援システム AGENT の開発，情報処理学会第22回全国大会予稿集，pp. 323-324 (1981).
11) 古川他：ソフトウェアテスト項目作成支援システム AGENT における仕様記述法の拡張，情報処理学会第25回全国大会予稿集，pp. 437-438 (1982).
12) 古川他：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).
13) 本多波雄：オートマトン・言語理論，コロナ社，東京 (1972).

(昭和58年11月29日受付)
(昭和59年2月14日採録)