

## 機能図式法 前編

## (A Test Cases Generation Method for Functional Testing)

2011/7/14 (木)

秋山 浩一、古川 善吾

## 1. はじめに

1984年に、『AGENT：機能テストのためのテスト項目作成の一手法』（古川、野木、徳永）と題した論文が情報処理学会論文誌に掲載されました。<sup>[1]</sup>

この論文は、ソフトウェア機能テストのためのテスト項目を系統的に作成する手法（AGENT法、もしくは、機能図式法と呼ぶ）について書かれたものです。

日本で発明されたテスト技法であるにも関わらず、機能図式は、これまでJaSST等での事例報告がありません。ソフトウェアテストの勉強会で聞いてみても「名前だけは聞いたことがあるけど使ったことはない」という人がほとんどです。使用していない理由を聞いてみると、「易しく書かれた本や、参考資料がない」からということでした。

そこで、今回は、機能図式についてできるだけ簡単に誰でもわかるように紹介してみようと思います。なお、本稿は全体を秋山が記述し、技術的なチェックを機能図式法の考案者である古川先生にお願いしました。

## 2. 機能図式法とは？

機能図式法とは、機能仕様を状態遷移図と原因結果グラフを組み合わせて形式的に記述する「機能図式記法」と、そうして記述した機能図式から網羅的なテスト項目を機械的に作成する手法のことです。

機能図式法のことをAGENT技法（=A Test Cases Generation Method for Functional Testing）とも呼び、AGENT技法を用いて自動的にテスト項目を作成するAGENTという名称のプログラムが日立製作所で作成されています（市販はされていないようです）。

機能図式法では、まず、機能仕様を入力や出力の順序に依存する部分を状態遷移図で表します。次に、各状態での入力データと出力データや遷移先状態と

の論理的関係を原因結果グラフ（あるいはデシジョンテーブル）で表現した機能図式を作成します。そして、作成した機能図式の状態遷移については構造化状態遷移のパス網羅（C1 カバレッジ）とループ0回・1回、論理的関係については原因結果グラフの網羅基準でテスト項目を生成します。

機能図式は、原因結果グラフと比べて、入力を行う順序や変換の順序を状態遷移図で表現することができますから、機能仕様をより自然な形で図として表現することができます。

また、機能仕様が評式的な機能図式に変換できれば、そこから機械的に何らかの網羅基準を使用してテスト項目を自動生成することができるというわけです。

それでは、次節から機能図式記法と、テスト項目生成方法について解説します。

### <コラム1:機能図式誕生秘話>

原因結果グラフの制約条件の中で Require を分析していくと論理的な関係としてではなく、入力の順序による制約であることに気がつきました。

この入力の順序を Require 制約で設定するのが自然でないと思ったので状態遷移を導入することを考えました。

## 3. 機能図式記法

機能図式は、大きく見ると状態遷移部と、論理関係部から成り立っています。

### 3.1 状態遷移部の記法

図 1 は、状態遷移部で使用する記法です。

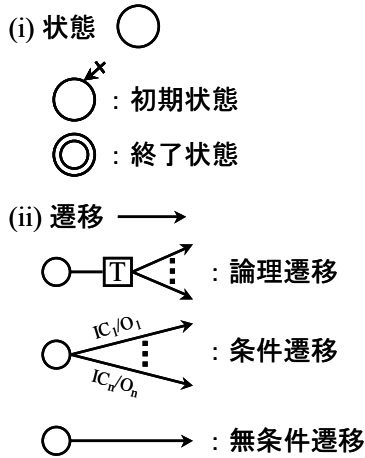


図 1 状態遷移部で使用する記法

状態は丸印「○」で表記します。状態とは、データの入力待ちをしている部分で、データを入力することで、次の状態へ遷移します。特別な状態として「初期状態」と「終了状態」があります。初期状態は、プログラムが動作し始める状態を示し、終了状態はそこでプログラムの動作が停止する可能性があることを示しています。

遷移は矢印「→」で表記します。論理遷移は、デジジョンテーブルで示される論理関係 T によって入力データによって、出力データと遷移先が決まります。論理関係 T は原因結果グラフを用いて作成します。

条件遷移は、入力「IC<sub>i</sub>」が互いに排他的でどれか一つが成立し、出力「O<sub>i</sub>」と遷移先が決まるときに使用します。無条件遷移は、状態からの遷移が一つで必ずその遷移が起こる場合に使用します。

### 3.2 論理関係部

論理関係部は原因結果グラフの表記方法を用います。原因結果グラフについては、G. J. Myers の『ソフトウェアテストの技法』や拙書『ソフトウェアテスト技法ドリル』を参照してください。論理関係部は機能図式中には、論理遷移「[T]」として表現されます。したがって、論理遷移「[T]」のテスト設計を別図で原因結果グラフを用いて実施すると考えてください。

## 4. 機能図式の例

準備が整いましたので、実際に機能図式の例を示します。ここでは、前述の論文にある「現金自動支払機」を使って説明します。

### <現金自動支払機の仕様>

現金自動支払機は、つねにカードが入力されるのを待っており(S0の状態)、カードが入力されると‘暗証番号入力’(M1)というメッセージを出す(S1へ遷移)。

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’(M2)というメッセージを出す(S2へ遷移)。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‘処理打ち切り’(M4)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’(M6)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(S3経由で無条件にS0へ遷移)。

不一致が、1回目、2回目のとき、‘暗証番号再入力’(M3)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(S1へ戻る)。

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’(M5)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(S2へ戻る)。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’(M6)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(S0へ遷移)。

### 4.1 状態遷移部

図 2 は、上記仕様を機能図式の状態遷移部に書き直したものです。

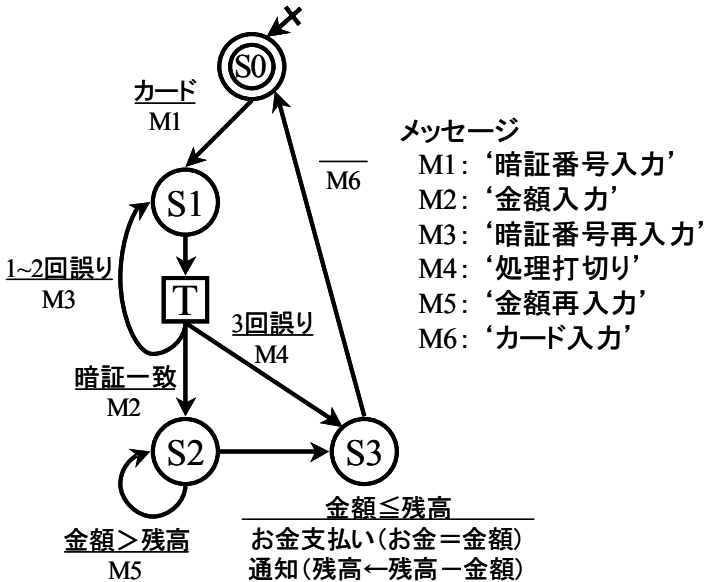


図 2 現金自動支払機の機能図式 (状態遷移部)

入力の順序が状態遷移図で表されていることが読み取れると思います。現金自動支払機の機能を実現する処理の流れが明らかになっています。

ここで、図 2 の中にある「カード / M1」という表記は、「入力条件 / 出力」を示しています。S0 から S1 に向かう「カード / M1」は、「S0 の状態でカードを入れると、「暗証番号入力」というメッセージを出力し S1 へ遷移する」仕様を表現しています。

## 5. 機能図式を用いたテスト項目作成方法

機能図式では、最終的に 1 枚のデシジョンテーブルを作成することになります。機能図式が網羅する条件は次の通りです。

- 機能図式の遷移 (C1) カバレッジ 100%
- ループに対して 0 回、1 回

C1 カバレッジ100%とループ網羅0回、1回は、バイザー本などに載っている状態遷移テストのパスの検出方法を用いることができます。そして、求めたパスに対してデシジョンテーブルTを使用しながら網羅的に大きなデシジョンテーブルを作成すればOKです。

ここでは、機能図式の論文に記載された方法を用いて上記網羅性を担保するパスの導き方を示します。

#### <コラム2:何もしないテスト>

グラフにループが存在した時に、ループに対して0回と1回のテストを実施するというのはソフトウェアテストの定石です。ところが、図2のように、構造化状態遷移図全体が一つのループになっていることも多いものです。

このような場合、機能図式では全体のループが0回、すなわち図2であれば「現金自動支払機の前に立って何もしなければ何も起こらない」というテストケースを生成します。ところが「何もしないテスト」はテストなのかというクレームが現場から上がりました。「当然テストだ!」「いやテストじゃない!」と侃々諤々議論したものです。

## 5.1 構造化状態遷移

一般的な状態遷移図に対してループを認識することは困難です。そこで、ループに複数の入り口がある状態遷移図はそれと等価な、ループへの入り口が一つしかない構造化状態遷移図に描きなおします。

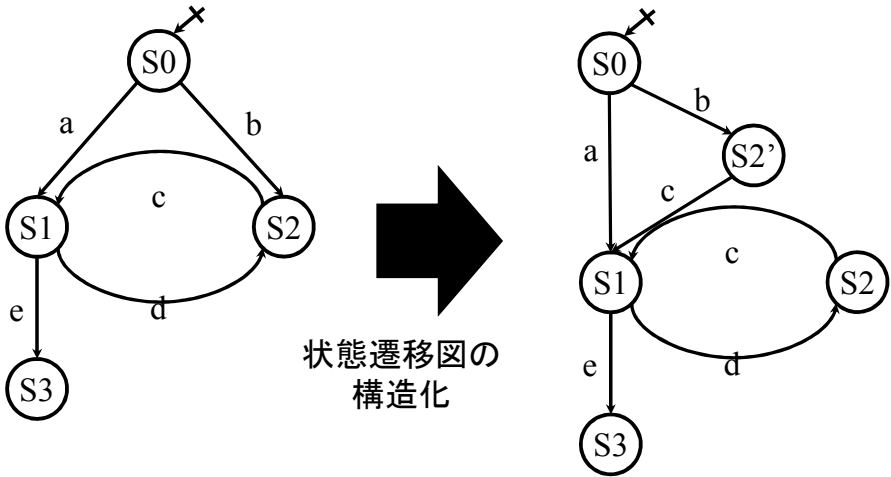


図 3 構造化状態遷移図への書換え

図 3 の左の状態遷移図は、状態 S1 と S2 の間に遷移 d と c によってループが生じています。また、このループへの入り口は状態 S0 から遷移 a で入る S1 への入り口と、状態 S0 から遷移 b で入る方法の 2 つが存在します。

そこでこれを図 3 の右の状態遷移図のように S2 を S2' として増やすことによってループへの入り口を S1 だけにします。これを構造化状態遷移図への書換えと呼びます。

## 5.2 構造化状態遷移図を正規表現に変換する

正規表現とは、文字列検索の時に使用するあれです。

たとえば、「This is an apple.」という文字列があった時に、「a」か「an」を検索したければ、「an\*」（前後をスペースで挟んでいます）と指定することで a が一文字とそれに続く n が 0 文字もしくは複数の n がヒットします。

もっともこの例では、a と an 以外にも、ann や annnnn もマッチングしてしましますが。

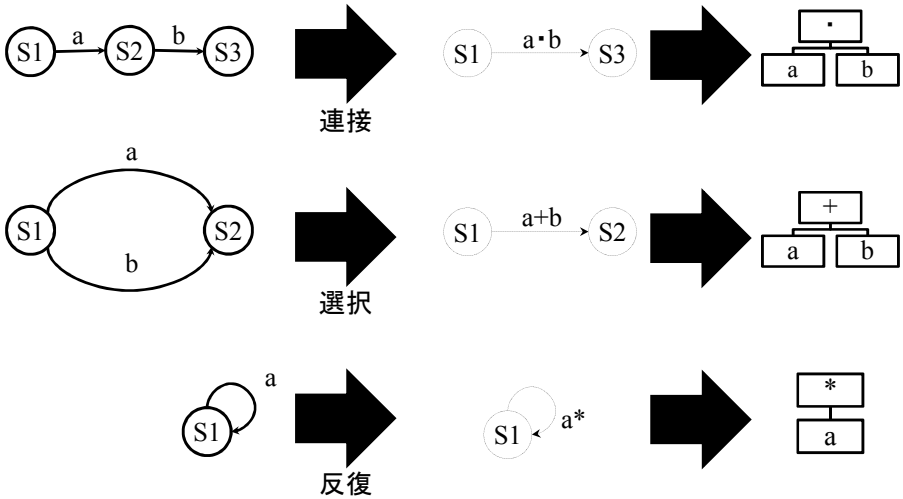


図 4 構造化状態遷移図の正規表現への変換

図 4 は、構造化状態遷移図の遷移の要素を正規表現に描きなおしたところです。構造化状態遷移図は、構造化されていますので接続・選択・反復の3つの構造を入れ子としたもので表現することができます。したがって、どんな構造化状態遷移図であってもそれを正規表現に変換することができます。

構造化状態遷移図を正規表現に変換する手段は次の通りです。

- (1) 遷移に名前を付ける
- (2) 同一状態から同一状態に至る遷移について遷移の和を取る
- (3) 任意の状態を削除する。この時、自己ループがあれば遷移の\*を作る
  - (ア) 削除した状態に入ってくる状態を数え上げる⇒遷移元状態
  - (イ) 削除した状態から出る状態を数え上げる⇒遷移先状態
  - (ウ) 全ての遷移元状態から全ての遷移先状態への遷移を作ってその正規表現式を作成する
- (4) 開始・終了状態以外の状態があれば(2)から繰り返す

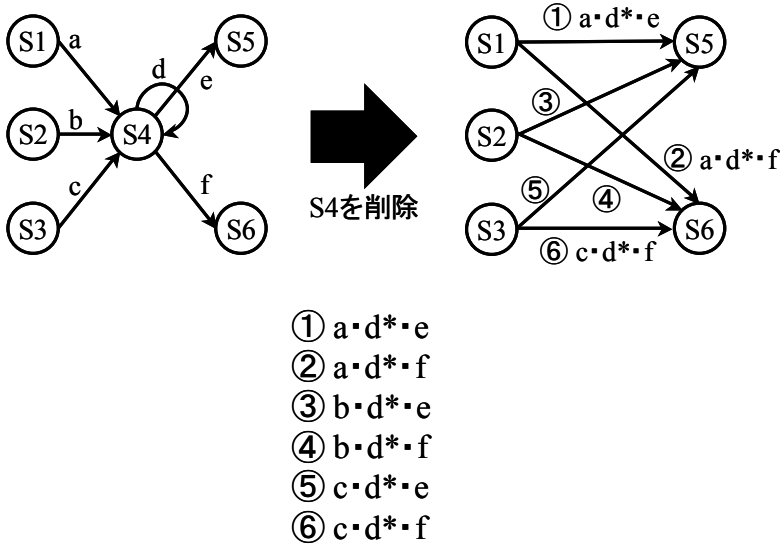


図 5 状態 S4 の削除による変化

は、状態 S4 を削除した時に、遷移がどう変化するかを示したものです。の左の構造化状態遷移図を見ると、S1 から S5 に向かうパスは S1 で a という遷移で、S4 の状態に遷移し、そこで d という遷移を 0 回から複数回受けたのち、e という遷移で S5 に遷移します。

したがって、ステップ(2)~(3)を適用することで、それが「 $a \cdot d^* \cdot e$ 」という正規表現式に変換されたわけです。

### 5.3 構造化状態遷移図を正規表現に変換する例

もう少し、具体的な構造化状態遷移図を正規表現に変換する例を示します。ここでは、図 3 の変換を行ってみます。

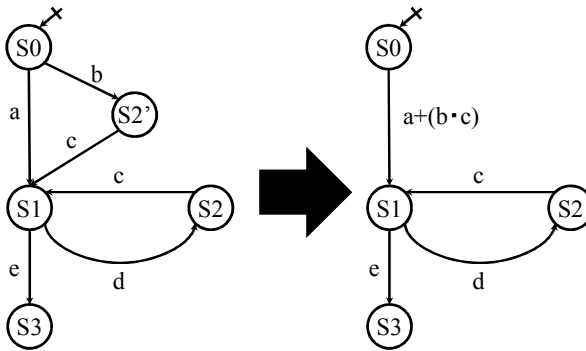


図 6 S2'の削除

図 6 は、状態  $S2'$  を削除したところです。  $S2'$  を削除したことで、  $S0$  から  $S1$  へ向かうパスは「 $a+(b \cdot c)$ 」に変わっています。

まず、  $S0$  から  $S2'$  を経由して  $S1$  に行くパスは **接続** ですから、「 $b \cdot c$ 」になります。次に、  $S0$  から  $S1$  に向かうパスは、「 $a$ 」と今変形した「 $b \cdot c$ 」の 2 通りになりましたので、 **選択** を使用して「 $a+(b \cdot c)$ 」となります。

このように、一つずつ状態を削除していきながら正規表現の式に変換していきます。

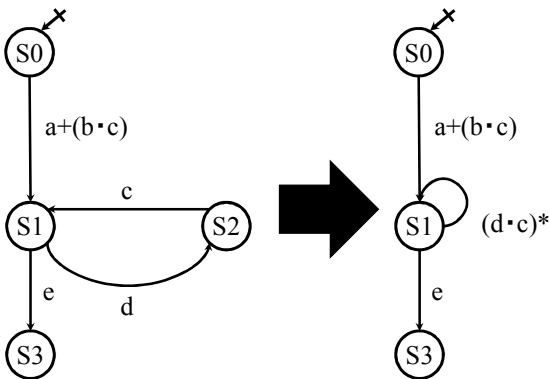


図 7 S2 の削除

図 7 は、状態 S2 を削除したところです。S2 を削除したことで、S1 から S1 へ向かうパスは「d・c」の反復になりましたので、正規表現式は「(d・c)\*」となります。

あとは、S1 を削除すればよいわけです。

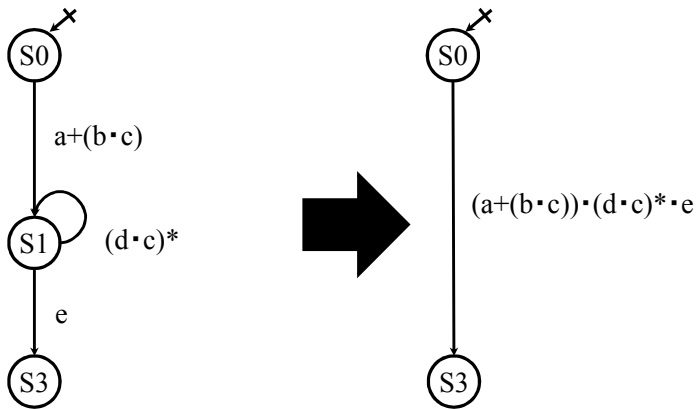


図 8 S1 の削除

図 8 は、状態 S1 を削除したところです。S0 から S3 へ向かうパスは 3 つの接続から成り立っています。したがって、接続を「・」でつなげば OK です。

結果は、「(a+(b・c))・(d・c)\*・e」となります。あとは、この正規表現式をツリー形式に書き換えてデシジョンテーブルを参照しながら大きなデシジョンテーブルを作るだけですが、それについては次回の後編で解説することになります。

## 6. 参考文献

- [1] 古川 善吾, 野木 兼六, 徳永 健司: 「AGENT: 機能テストのためのテスト項目作成の一手法」(1984), 情報処理学会, Vol. 25 No. 5