原因流れ図によるテストケース抽出技法

1M-7

赤沢　隆夫　　松尾谷　徹*

日本電気ソフトウェア㈱　　日本電気㈱*

1．はじめに

冗長でないテストケース抽出の一方式として「原因－結果グラフ技法」にプログラム言語の構文則を定義する構文流れ図（これを「原因流れ図」と呼ぶ）を提案し、事例として原因－結果グラフ技法の５０％でテストケースを抽出できることを前回の論文で示した。1)

「原因－結果グラフ技法」や「実験計画法を用いたテストケース抽出法」では決定木（decision tree）または決定表（decision table）をテスト項目抽出結果の表現形式としているため、扱う問題すなわちプログラム仕様のうち条件やＣＡＳＥ構造は表わし得るが処理の順序（逐次処理やループ処理）については定義できない。2)、3)

そこで決定木や決定表の中間表現として、原因流れ図を用いてプログラムの制御構造および処理の流れにおける各状態を記述し、決定表に変換する方式とした。

本稿では原因流れ図の記述方法の定義（ＢＮＦによる構文則）と決定表の作成方法について考察し、さらにテスト容易なプログラムの設計技法としても有効なことを示す。

2．原因流れ図の表現形式

原因流れ図は原因と結果についてそれぞれ非終端記号と終端記号を持ち、それらは左上から右下へ向かう処理の流れとして表現される（図１、図２）。非終端記号および終端記号の意味を図３に示す。

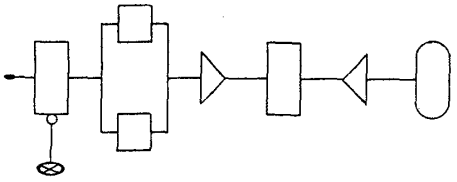

図１　非終端記号で表わされた原因流れ図

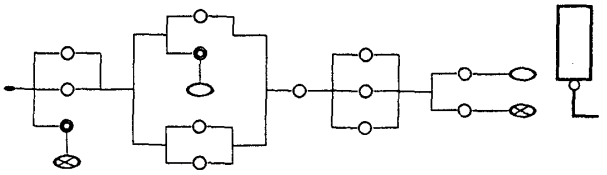

図２　図１を終端記号で表現した例

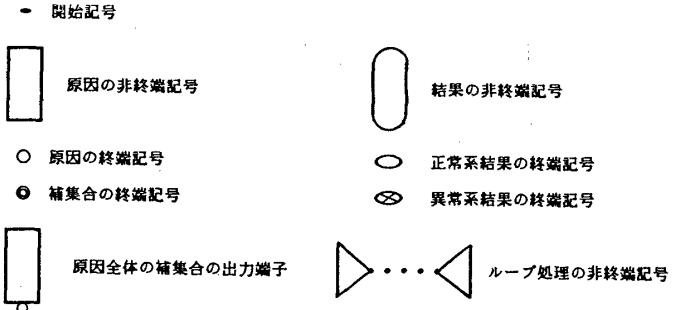

図３　原因流れ図の記号

次に原因流れ図の生成規則について述べる。

① ＜結果＞::＝＜原因１＞＜原因２＞……＜原因ｎ＞
② ＜結果＞::＝＜ ○ ＞ | ＜ ⊗ ＞
③ ＜原因＞::＝＜原因＞ | ＜結果＞
④ ＜原因＞::＝＜ ○ ＞ | ＜ ◉ ＞
⑤ ＜ﾙｰﾌﾟ処理＞::＝＜ ○ ＞ | ＜ ◉ ＞

これらの生成規則によって、プログラムの制御構造を階層的に記述することができる。原因はプログラムの外部または内部状態を示す値や条件式等で記述し、結果は実行される動作や出力を記述する。１つの結果が新たな遷移状態を示す場合その結果自体も原因とすることができる。

次にループ処理と補集合の扱いについて述べる。ループ処理は原因の非終端記号を１つ以上含み、原因の前後に置くことによりそれぞれループの開始条件と終了条件を表す。補集合は原因の非終端記号に出力端子として１つだけ付けることができる。補集合の終端記号は原因の非終端記号の補集合と同値な場合とそうでない場合がある（図４）。

ｂはａの補集合と同値な場合である。
ａの展開後ｃの様に非終端記号が存在する場合でさらに、非終端記号に対する補集合が展開された場合がｄである。この２つの補集合は同値ではあり得ない（結果はプログラムにより同一である場合もある）

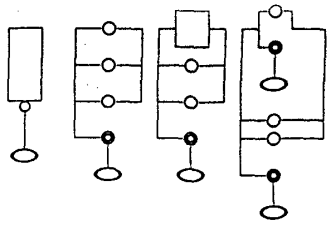

図４　補集合の生成規則　a　b　c　d

Derivation Method for Test Case by Flowchart

Takao Akazawa NEC Software, Ltd.

3．デシジョンテーブルの作成

原因ー結果グラフ技法によるデシジョンテーブル作成の問題点は原因を積の組み合せとしてテストケースとすることによる冗長性であった。原因流れ図では非終端原因から１つの終端原因が選ばれた状態での積とする。選択ルールは非終端記号からなる１つのテストケースにおいて最大の終端記号数以下サイクリックに終端記号を選ぶとする。また、異常系のテストケースの場合については正常系と異なるパスを通る様にすることによりテストケースを最小限に抑えかつプログラム上の網羅度についても高めることができる。図５、図６および図７にて原因流れ図よりテシジョンテーブルを作成した例を示す。

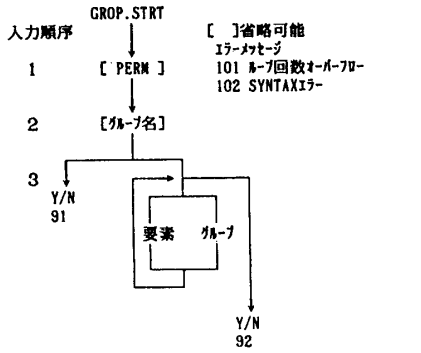

図５　テストすべき会話型プログラムのコマンドフロー

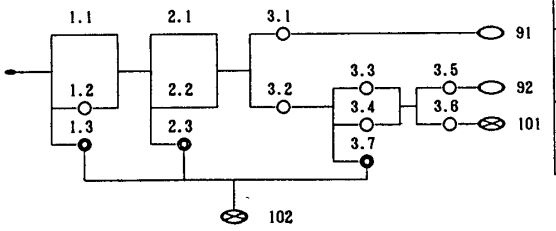

図６　会話型プログラム（図５）の原因流れ図

原因		テストケース								
結果		1	2	3	4	5	6	7	8	9
1	1.1	(1)		(1)		(1)			(1)	
	1.2		1		1		1			1
	1.3							0		
2	2.1	(1)		(1)		(1)				(1)
	2.2		1		1		1			
	2.3								0	
▷ 3	3.1	1	1							
	3.2			1	1	1	1			1
	3.3			1		1				
	3.4				1		1			
◁	3.5			1	1					
◁	3.6					1	1			
	3.7									0
正常	91	1	1							
	92			1	1					
異常	101					1	1			
	102							1	1	1

（ ）はNo Operationまたは省略処理を示す。
▷ ◁ はそれぞれループの開始条件、終了条件を示す。

図７　原因流れ図（図６）によるデシジョンテーブル

4．原因流れ図によるプログラム設計

以上、原因流れ図によるテストケース抽出の方法について述べたが、原因流れ図はプログラム構造についてSyntaxだけでなくSemantics チェックの機能を持っている。ループ処理は開始条件、終了条件および実行される処理の３つの構造から成立すると定義される。また結果は、正常系または異常系のいずれかに分類される。補集合は階層的に分離される。これらの機能をプログラム設計の制約条件とみなすことにより、簡潔なプログラム構造（機械的にテストケースを抽出しても、重複が少なくてすむ構造）を定義できる。

原因流れ図によるプログラム設計はＳＰＤなどの構造化設計やジャクソン法に代表されるデータ指向のプログラミングに対してそれらの接点となる位置を占めるものであり、これらの技法の入力となる抽象化技法とも言えるであろう。また、原因流れ図はプログラムフローからデシジョンテーブルを導くことができるため、設計時にテストケースの抽出を行いプログラムを検証することができる。トップダウンにプログラムを設計する場合、機能と構造の相互検証が重要である。このような観点から、原因流れ図の構造に対する制約性と機能に対する検証性はプログラム設計の上流でより有効であることがわかる。

5．まとめ

プログラムにおけるテストケース抽出とプログラム制御構造の設計を同時に行うことが可能な方法論として原因流れ図の理論について述べたが、今後のアプローチとしてＳＶＡ（テストケース抽出ツール）等に原因流れ図の機能を実現することが考えられる。4)

原因流れ図をツール化する場合、非終端記号と終端記号間および原因と結果間（一つの流れ図全体が次の原因となる）の相互変換および結合の方法について検討しなければならない。

参考文献

[1]　松尾谷：テストケース抽出の一方式
情報処理学会第３７回大会
[2]　大場　充著：ソフトウェアの開発技術
ＳＥシリーズ３　オーム社
[3]　J.Martin, C.McClure著：ソフトウェア構造化技法
近代科学社
[4]　織田他：原因ー結果グラフ技法を用いたレビュー支援ツールＳＶＡ
情報処理学会第３７回大会