Vol.2013-SE-179 No.17
2013/3/11

シーケンス制御プログラムのテストに適した新しいカバレッジ基準の提案

丸地康平[†1]　酒井政裕[†1]　進博正[†1]

本論文では，シーケンス制御プラグラミング言語に適した新しいカバレッジ基準 MTC を提案する．MTC は，必要となるテストケース数を抑えつつ，効果的なテストを実現するためのカバレッジ基準であり，論理回路におけるトグル網羅とソフトウェアにおける MC/DC 網羅の両性質を併せ持つことを特徴とする．ミューテーションテストによる評価実験により，MTC の有効性を確認した．

MTC: A New Test Coverage Criterion for Sequence Control Programs

KOHEI MARUCHI[†1]　MASAHIRO SAKAI[†1]
HIROMASA SHIN[†1]

In this paper, we propose new coverage criterion called MTC, which is suited to testing for sequence control programs. MTC combines two popular coverage criteria, namely toggle coverage commonly used for digital circuits, and modified condition/decision coverage (MC/DC) commonly used for safety-critical software, to achieve effective testing while keeping the number of necessary test cases relatively small. We have evaluated the effectiveness of MTC by using mutation testing method.

1. はじめに

社会インフラシステムの信頼性を確保することは，我々が安全に安心して生活するために必要不可欠である．システムの信頼性を確保するためには，適切なテスト基準の下で，網羅的な試験を行うことが重要である．

汎用的なプログラム言語 C や Java に関しては，全てのプログラム行を実行する命令網羅や全ての条件分岐を実行する分岐網羅といった構造カバレッジ基準が広く知られ，プログラムの信頼性を確保するために活用されている．

一方，発電プラントの制御プログラムは，専用のシーケンス制御プログラム言語を用いて開発されるため，これらのプログラムの信頼性を確保するには，専用のカバレッジ基準が必要となる．

本論文では，シーケンス制御プログラム言語に適した新しいカバレッジ基準 MTC (Modified Toggle Coverage)を，論理回路におけるトグルカバレッジとソフトウェアの汎用言語において，航空業界で実績のある MC/DC (Modified Condition / Decision Coverage)の考え方を取り入れて，提案する．

また，典型的な不具合をどれだけ発見できるかをみるミューテーションテスト技法を用いた評価実験を行い，提案カバレッジ基準の有効性を示す．

2. シーケンス制御プログラム言語

本論文で対象とするシーケンス制御プログラミング言語は，図 1 のようにダイアグラムで記述する言語であり，IEC 61131-31[1]及び JIS B3503[2]で規定される 5 つのプログラマブルコントローラ向けプログラミング言語の内，FBD (Function Block Diagram)に相当する言語である．

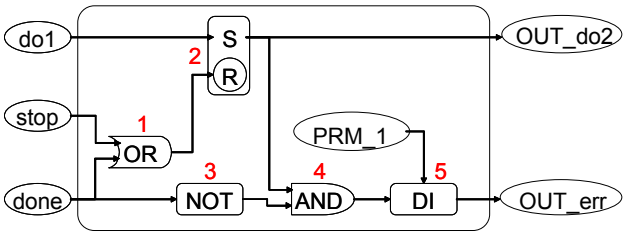

図 1　シーケンス制御言語の例

FBD では，構成要素となるファンクション・ブロックの入出力を信号線で結ぶことでプログラムを作成する．例えば，図 1 のサンプルの入力 stop と done は OR ゲートの入力となり，入力 do1 と OR ゲートの出力はフリップフロップ（図 1 におけるブロック番号 2）の入力となる．

東芝の発電プラント制御プログラムも，図 1 のようなダイアグラム表現で記述する独自言語により，開発されている．ダイアグラムで表現されたプログラムは，図 2 のようなファンクション・ブロックの羅列によるテキスト形式の表現に変換することで，機械が解釈して実行可能となる．

†1 (株)東芝研究開発センター　システム技術ラボラトリー
Toshiba Corporation Corporate Research & Development Center

```
1 OR  stop,done↓
2 OUT r1↓
3 FF  r2,do1,r1↓
4 OUT OUT_do2↓
5 NOT done↓
6 AND r2↓
7 DI  PRM_1↓
8 OUT OUTerr↓
```

図 2　テキスト形式の表現例

3. 新カバレッジ基準 MTC の提案

テストカバレッジとは，テストが十分であるかを知るための尺度である．確認すべき項目を決め，テスト中どれだけカバーできたかを示す網羅率を用いて定量化する．全ての入力と状態の組み合わせをテストするのが理想であるが，組み合わせ爆発により，現実的な時間内で実施するのは不可能である．そのため，適切なテストカバレッジを用い，そのカバレッジを網羅するテストを行うという基準を定めることで，テストに要する時間を抑えつつ製品の品質を保証することが可能となる．

本節では，シーケンス制御プログラム言語向けのテストカバレッジ基準 MTC を提案する．MTC は論理回路におけるトグルカバレッジとソフトウェアにおける MC/DC の両方の考え方を併せたカバレッジ基準である．

3.1 トグルカバレッジ

トグルカバレッジは，論理回路に使われるカバレッジである．論理回路上の信号線に注目し，テスト中にどれだけの信号線が“0 から 1”と“1 から 0”への両方の値の変化が行われたかを見る．全ての信号線が両方の値の変化をした場合，トグル網羅という．信号線 m 本の内，n 本の信号線が両方の変化をした場合，トグル網羅率は n/m となる．

表 1 And ゲートのテストケース

テストケース	入力1	入力2	出力
t00	0	0	0
t01	0	1	0
t10	1	0	0
t11	1	1	1

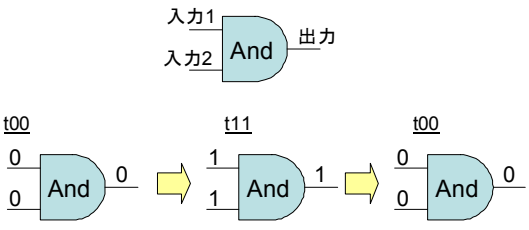

図 3 And ゲートのトグル網羅のテスト例

例として，2 入力の And ゲートをテストする場合を考える（図 3）．表 1 は 2 入力の And ゲートに対し，考えられる全てのテストケースを表にしたものである．このとき，テストケース t00，t11，t00 の順番で実施すると，全ての信号線において 0 から 1, 1 から 0 の両方の値の変化を確認できるため，トグル網羅達成となる（図 3）．

一方， t01，t10，t01 の順番でテストを行った場合は，入力 1 と 2 において 0 から 1, 1 から 0 の両方の変化を確認できるが，出力は 0 のままで値の変化を確認できない．この場合はトグル網羅率 2/3 となる．

トグル網羅のテストを実施することで，論理回路図上の信号線全ての値の変化を確認できる．各信号線は与えられた役割を担っており，もしテスト中に片方の値しか持たない信号線があれば, その信号線が関わる仕様や機能に対し，十分なテストができていないことが分る．トグル網羅率を見ることにより，このようなテストの抜けが無いかを確認することができる．但し，トグル網羅できないからといってテストに抜けがあるとは限らない．論理回路図の方に冗長なゲートや信号線があると考えられるためである．このような冗長な箇所は不具合の混入に繋がりやすいため，取り除くことが望ましい．トグル網羅率を見て，網羅しない理由を考察することにより，このような冗長な部分の発見に繋げることができる．以上のようなメリットがあることから，トグル網羅は論理回路の検証に実際に活用されるカバレッジ基準である[3].

一方，トグル網羅のテストを実施していれば十分なテストとは限らない．例えば，トグル網羅のテストでは，And ゲートと Or ゲートを誤って付け間違えるといったシンプルなミスを検出できない場合がある．

表 2　And ゲートと Or ゲートのテストケース比較

テストケース	入力1	入力2	出力(AND)	出力(OR)
t00	0	0	0	0
t01	0	1	0	1
t10	1	0	0	1
t11	1	1	1	1

2 入力の And ゲートにトグル網羅のテスト（t00, t11, t00 の順番のテスト）を行うことを考える．このとき，誤って And ゲートではなく Or ゲートをテスト対象としたときに，テスト対象の違いを検出できるかを考える．表 2 は And と Or のテストケースを表にしたものである．表 2 より，トグル網羅を達成する t00，t11，t00 の順番のテストを実施しても， And ゲートと Or ゲートの出力が変わらず，テスト対象の違いを検出できないことが分かる．

今回ターゲットとしているシーケンス制御プログラム言語は図 1 のようにグラフィカルなエディタ上で，ファンクション・ブロックを結びつける言語である．ファンクション・ブロックの 1 つである And ゲートと Or ゲートを付け間違えることは典型的なミスの 1 つである．トグル網羅の

テストでは, これらのミスを検出できない場合があるため, より厳しいカバレッジ基準が必要である.

3.2 MC/DC (Modified Condition / Decision Coverage)

MC/DC とは，ソフトウェアのカバレッジの中でも，プログラムコードの分岐構造に注目して，十分テストしているかを判断するコードカバレッジの 1 つである.

コードカバレッジには，テスト中にソースコードの命令語を全て実行したかを見る命令網羅 C0, ソースコード上の分岐命令の真偽値の両方を全て実行したかを見る分岐網羅 C1，条件文の真偽値の組み合わせを全て実行したかを見る複合条件網羅 C2 といったカバレッジ基準が広く知られている．これらカバレッジ基準には包含関係が成り立つことも知られている[4]. カバレッジ基準 A がカバレッジ基準 B を包含するとは，カバレッジ基準 A を満たすテストが必ずカバレッジ基準 B を満たすときに言う．複合条件網羅 C2 は分岐網羅 C1 を包含し，分岐網羅 C1 は命令網羅 C0 を包含する．ゆえに，この 3 つのカバレッジ基準であれば，複合条件網羅 C2 が最も厳しい信頼性の高い基準であるが，条件数が増えるにしたがい指数関数的に組み合わせ数が増えるため，現実的に実施が困難なカバレッジ基準である.

一方, 命令網羅 C0 や分岐網羅 C1 は現実的な時間で実施できる実用可能なカバレッジ基準であるが，テストの実施が十分なことを示すには，より厳しい基準が求められる.

実用的なテスト数で済み，分岐網羅 C1 より厳しいカバレッジ基準として，MC/DC[4]網羅がある．MC/DC は米国の航空機産業団体である RTCA で策定されたカバレッジであり，ソフトウェア開発のためのガイドラインである D-178 に「故障が人命に関わるソフトウェアに対し，MC/DC100%を満たすテストをすること」と規定される[5]. このように, MC/DC 網羅は航空産業において安全性が高く求められるシステムの検証に対し，実績のあるカバレッジ基準である.

MCDC1：プログラムの全入力、出力を少なくとも一回はテストすること
MCDC2：プログラムの判定に含まれる全条件に対し，可能な値を少なくとも一回はテストすること
MCDC3：プログラムの全判定に対し，可能な値を少なくとも一回はテストすること
MCDC4：プログラムの判定の全条件は判定の出力に独立に影響することを示すこと

図 4　MC/DC 網羅の条件

MC/DC を網羅するには，図 4 の 4 つの条件（MCDC1～MCDC4）を全て満たすことが必要になる.

ここで, if ((x == 3) or (y > 2)) {then z++;} のプログラムを例に MC/DC 網羅について説明する．表 3 は，今回のプログラムに対し MC/DC 網羅を満たすテストである.

表 3　MC/DC 網羅のテストケース例

テストケース			判定 (Decision)	条件 (Condition)	
名前	x	y	(x == 3) or (y > 2)	x == 3	y > 2
t1	1	1	false	false	false
t2	1	3	true	false	true
t3	3	1	true	true	false

まず，MCDC1 を満たすには，プログラムの全入力，全出力をテストしたかどうかを確認する．今回のプログラムは（簡易的なサンプルであり，入力や出力を明確に定義していないが），MCDC1 を満たすものとする.

次に MCDC2 には判定（Decision）と条件（Condition）という用語が出現する．判定とは，if 文のような判定文における述語全体のことであり，今回のプログラムでは，((x == 3) or (y > 2))が判定に相当する．条件とは，判定を構成する最小の述語であり，今回のプログラムでは，(x==3)と(y>2)が条件に相当する．ゆえに，MCDC2 を満たすには，判定に含まれる全条件である(x==3)と(y>2)が true と false 両方の値をテスト中に保持すればよい．表 3 のテストは，t2 より(x==3)が false，(y>2)が true，t3 より(x==3)が true，(y>2)が false の値を取るため，MCDC2 を満たすことが分る.

MDCD3 では, プログラムに含まれる全判定が true と false 両方の値をテスト中に保持すればよい. 表 3 のテストでは, 判定は((x == 3) or (y > 2))のみであり，t1 により false，t2 により true の値を取るため，MCDC3 を満たすことが分る.

最後の MCDC4 が，最も特徴的な条件である.「条件が独立に影響すること」は，他の条件値はそのままで，注目する条件値のみを変えて判定値が変化することを示すことで確認する．MCDC4 を満たすには同様の確認を全ての条件に対し行う必要がある.

表 3 のテストでは，t1 と t3，t1 と t2 がそれぞれ，条件(x==3)と条件(y>2)の独立影響を確認するテストケースのペアとなる．t1 と t3 を見ると，(x==3)の値は false と true で反転し，それ以外の条件である(y>2)は false で等しく，判定は false と true で反転していることが確認できる. 同様に t1 と t2 を見ると，(y>2)の値は false と true で反転し，それ以外の条件である(x==3)は false で等しく，判定は false と true で反転していることが確認できる.

このように，各条件に対しテストケースのペアが必要となるため，MC/DC は条件数 n に対し，n+1~2n 個のテストケースが必要となる[4]．これは十分にテスト可能な数である.

3.3 MTC (Modified Toggle Coverage)

3.1 節で紹介したトグル網羅は，論理回路の構成要素である信号線に注目したカバレッジ基準である．ソフトウェアの命令網羅や分岐網羅もソースコード上の命令や分岐といったソースコードの構成要素に注目するカバレッジ基準

であり，トグル網羅も，命令網羅や分岐網羅も，どちらも構成要素に注目したカバレッジ基準といえる．

3.2 節で紹介したように，ソフトウェアにおけるカバレッジ基準では，構成要素に注目した基準より厳しく実用的なテストケース数で済む MC/DC 網羅がある．

本節では，論理回路において構成要素に注目したトグル網羅より厳しく実用的な基準を，MC/DC を参考に考案し，シーケンス制御プログラム言語への適用を検討する．

ここで，図 5 を検証対象の論理回路の構成とする．これを論理式で表すと Out = (In1∧In2)∨(In3∧In4)となる．入力 In1，In2，In3，In4 が条件に相当し，出力 Out が判定に相当することが確認できる．そのため，ソフトウェアで言う条件を論理回路の入力，判定を出力とみなすことで MC/DC を論理回路に適用する．

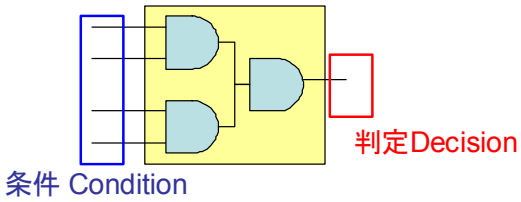

図 5　MC/DC を論理回路へ適用する様子

このように MC/DC を論理回路に適用したカバレッジと，トグルカバレッジとを不具合の検出能力，カバレッジの包含関係に関して比較する．

不具合の検出能力に関しては，3.1 節で紹介したトグル網羅のテストで，And ゲートを Or ゲートの違いを検出できないケースがあるが，MC/DC 網羅のテストにも該当するのか考える．

2 入力 And ゲートに対して MC/DC 網羅となる最小テストケース数のテストは(t01，t10，t11)となるが（表 1），t01 と t10 は Or ゲートと出力が異なり（表 2），違いを検出できることがわかる．ゆえに，MC/DC 網羅のテストは，2 入力の And ゲートを Or ゲートと付け間違えたミスを検出できることが分かる．

テストケース	A	B	C	Y
T1	0	0	1	0
T2	1	0	1	1
T3	0	1	1	1
T4	0	1	0	0

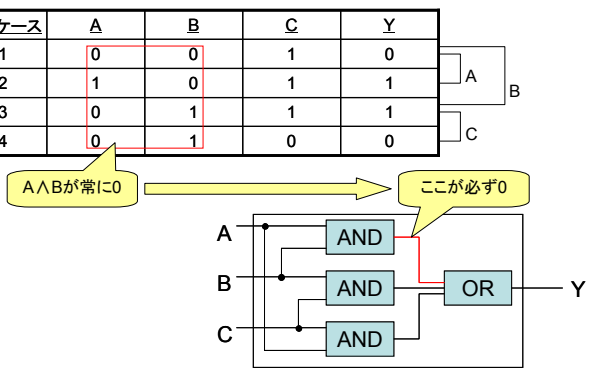

図 6 MC/DC 網羅だがトグル網羅で無い例

カバレッジの包含性に関しては，図 6 の例からトグル網羅と MC/DC 網羅は包含関係に無いことが分かる． 図 6 は，論理式(A∧B)∨ (B∧C)∨ (C∧A)を表現した回路図と，その回路図に対し，MC/DC 網羅となるテストの表で構成される．MD/DC 網羅のテスト表から，A∧B の値が必ず 0 であることが分る．図 6 の回路上の信号線の一つが A∧B に対応しているため，この信号線はテスト中 0 しか値を取らないことが分る．ゆえに，MC/DC 網羅のテストが必ずしもトグル網羅でなく包含関係にないことが分かる．

以上より，論理回路に適用した MC/DC 網羅はトグル網羅よりも誤りの検出力で優れていることを期待できるが，包含関係にあるカバレッジ基準ではない．そこで，両方の性質を兼ね備えているカバレッジ基準として MTC 網羅を，図 7 記載の 2 条件を満たす基準として提案する．

MTC1:各結線が0から1，1から0の両方の値の変化をすること
MTC2:各入力の変化が出力に独立に影響することを示すこと

図 7　MTC 網羅の条件

MTC 網羅するには，トグル網羅を満たすこと（条件 MTC1）と，MC/DC 網羅の条件（図 4）の中で最も特徴的な MCDC4 を満たすこと（条件 MTC2）が必要である．

図 8 は MTC2 のイメージ図である．図 8 では，一番上の入力線（条件）が出力（判定）に独立に影響を与えている様子を表している．一番上の入力線（条件）への入力値だけが異なるテストケースのペアを用意し，これらテストケースにより出力線（判定）の値が異なることを確認できれば，図 8 のように入力線 1 つの値を動かした時に出力線に寄与しているかを確認ができたこととなる．MTC2 を満たすには，同様の確認を全ての入力線で行う．

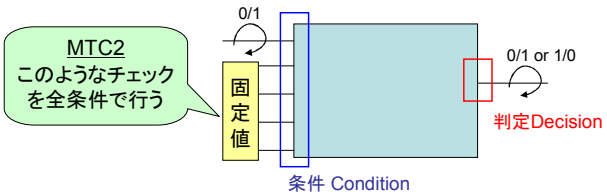

図 8　MTC2 のイメージ図

なお，シーケンス制御プログラム言語は図 1 のように，通常内部状態を持ち，出力も複数ある．図 9 は図 1 に内部状態を追記した図である．2 番のブロックはリセット優先フリップフロップであり，ブール値の状態 s2 を持つ．入力 S が 1 になれば，s2 は 1 にセットされ，入力 R が 1 になれば，s2 は 0 にリセットされる．5 番のブロックは遅延タイマであり，入力が 1 になってから一定時間（PRM_1 の値）後に出力が 1 となる動作をする．遅延タイマは，タイマのカウント値を保持する内部状態 s5 を持つ．このようなプロ

グラムに対して MTC2 の条件は，出力に繋がる内部状態も入力と見なすこととし，「全てのブール型の出力に対し，出力に繋がる入力と内部状態から独立に影響すること示すこと」と定義する．入力に関しては，ブール型に限定しないことに注意する．

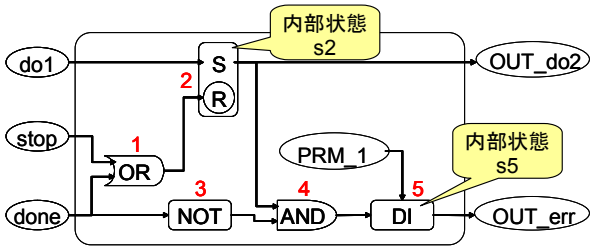

図 9　内部状態を記したシーケンス制御プログラムの例

図 9 の例で，MTC2 を満たすには，出力 OUT_do2 に入力 do1，stop，内部状態 s2 がそれぞれ独立に影響することと，出力 OUT_err に入力 do1，stop，done，PRM_1，内部状態 s2，s5 がそれぞれ独立に影響することを示す必要があり，計 9 項目の確認が必要となる．これに必要なテストケース数は 1 項目辺り 2 つとなるため，最大 18 となる．

MTC のカバレッジ網羅率は，条件 MTC1 と MTC 2 でそれぞれ別に定義する．条件 MTC1 は，トグルカバレッジと同様，プログラム内部のブール値の信号線の内，テスト中に 0 から 1，1 から 0 の両方の変化をしたものの割合とし，条件 MTC2 は独立影響を確認する必要のある項目の内，テスト中に確認した割合とする．これらは，それぞれ，トグルカバレッジ，MC/DC の網羅率と同義であるため，以降はトグル網羅率，MC/DC 網羅率と表記する．

4. 評価

本節では，提案したカバレッジ基準を実際の発電プラント制御で使われるプログラムをモチーフに，ミューテーションテストによる評価を行う．すなわち，各カバレッジ基準に対して，カバレッジ網羅のテストパターンを生成し，各テストパターンがどれだけ典型的な不具合を見つけることができるかで，カバレッジ基準を評価する．

4.1 ミューテーションテスト

ミューテーションテスト[6]とは，ソフトウェアにおけるテストケース評価技法である．機械的に不具合をソースコードに埋め込み，どれだけ不具合を検出できるかで評価する．不具合の埋め込みは，ソースコードの一部を改変（ミューテート）することで行い，改変されたソースコードはミュータントと呼ばれる．

図 10 は，ソースコードからミュータントを 4 つ生成した様子を表す模式図である．各ミュータントは元のソースコードから異なる 1 箇所を改変され作られる．あるテストパターンが4つのミュータントの内3つを検出できる場合，ミューテーションスコア 75% (3/4)のテストパターンであると，定量的に評価することができる．ミューテーションスコアは，高い程良いテストであることが言える．

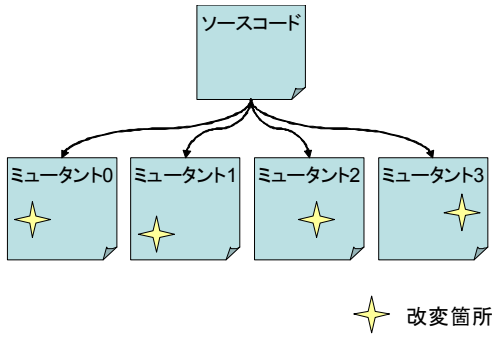

図 10　ミューテーションテスト概要図

ミューテーションテストを行う際に，どのような不具合モデルを用いて改変を行うかが重要となる．今回は，典型的な不具合モデルとして，「And ゲートと OR ゲートの付け間違い」と「Not ゲートの付け忘れ（誤って付けるのも含める）」を用いる．これらは，不具合報告書や設計担当者へのヒアリングを通じて得られた典型的な不具合事例である．

4.2 実験方法

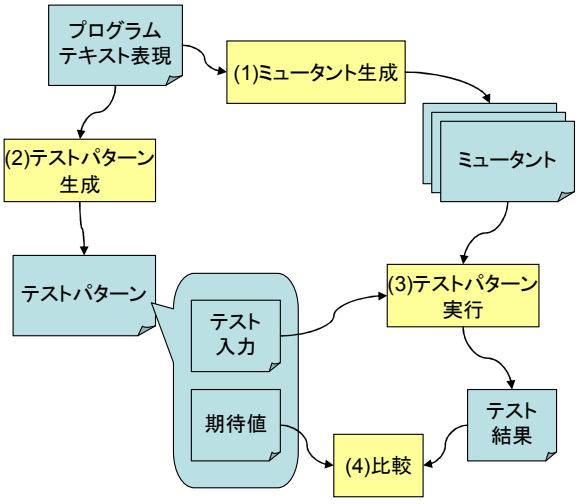

図 11　テスト評価実験概要

カバレッジ基準の評価は，各カバレッジ基準を満たすテストパターンにミューテーションテストを行い，評価する．その手順を図示したのが，図 11 である．まず，プログラムのテキスト表現（図 2 相当）を元にミュータントを作成する（1）．次に，元のプログラムをテストするためのテストパターンを生成する（2）．テストパターンは，テストケースの集合体であり，テストケースはテスト入力と期待値のペアである．続いて，各ミュータントにテストパターン

の入力を与え，出力結果を得る（3）．最後に，この結果と期待値が異なるかを比較する（4）．異なるテストケースを含むテストパターンは，このミュータントに相当する不具合を検出できることを意味する．

以下に，各ステップの内容を詳しく説明する．

(1) ミュータントの生成方法

プログラムのテキスト表現の中身を書き換えることで行う．「And ゲートと OR ゲートの付け間違い」に相当するミュータントは， And 命令を Or 命令に，Or 命令を And 命令に書き換えて生成する．

「Not ゲートの付け忘れ」に相当するミュータントは，ブール値の変数の符号を書き換えることで表現する．今回の実験では，ブール演算を司る And 命令, Or 命令，Xor 命令（排他的論理和）に限定し，これら入力の符号を反転させるように書き換えて生成する．

(2) テストパターンの生成方法

図 12 が，今回行ったテストパターン生成方法の構成図である．制約ソルバを用いてカバレッジ網羅のテストパターンを作ることを特徴とし，制約ソルバは SMT ソルバの CVC3[7]を用いる．

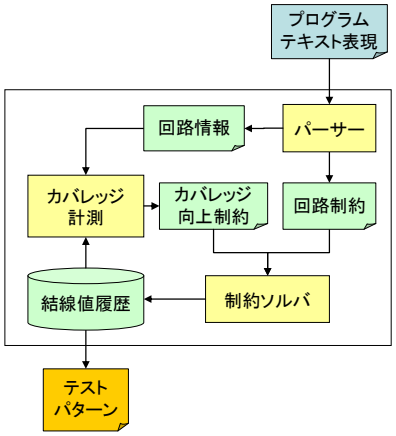

図 12 テストパターン生成構成図

テスト生成の手順は下記の通りである．まず，プログラムのテキスト表現をパースして，制約式に変換した回路制約, カバレッジを計測する上で必要となる回路情報を得る．回路制約は，各基本命令を制約式で表記し，論理積で結ぶことで作成する．図 13 のプログラムの場合の回路制約は図 14 のようになる．このとき，内部状態は事前状態と事後状態で分けて考える．

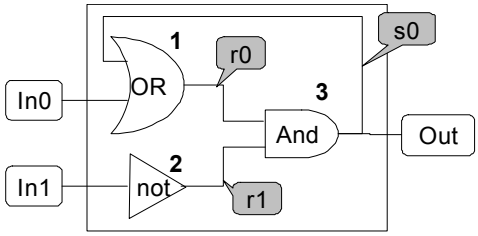

図 13 プログラム例

```
1 trans(In0,In1,s0_pre,Out,s0_post)↓
2 =(↓
3     (r0 = In0 or state0_pre)↓
4     and (r1 = not In1)↓
5     and (state0 = r0 and r0)↓
6     and (Out =state0_post) ↓
7 )↓
8 ↓
```

図 14 回路制約の例

カバレッジ基準により必要となる回路情報は異なる．トグルカバレッジの場合は，回路内のブール値の信号線のリストが必要となる． MC/DC の場合は，出力と出力に繋がった入力のリストが必要となる．

次に，これまで作成したテスト履歴から，カバレッジの計測を行い，カバレッジを網羅していない項目を探す．そして，その項目のカバレッジを満たすために必要な制約式であるカバレッジ向上制約を作成する．例えば，トグルカバレッジで信号線 r1 が false から true への変化に関して，未網羅であれば，図 15 のようになる．

図 15 の 1 から 3 行目は 2 つのテストケースが必要であるため，2 つの回路制約を定義している．更に，トグルカバレッジでは，これら 2 つのテストケースの順番を指定する必要があるため，状態を持つ場合は 5 行目のように片方の遷移の事後状態がもう片方の遷移の事前状態となる必要がある．7 行目は信号線 r0 が false から true へ変化することを表している．

```
1 trans(t0_In0,t0_In1,t0_s0_pre,t0_Out,t0_s0_post)↓
2 and↓
3 trans(t1_In0,t1_In1,t0_s0_pre,t1_Out,t0_s1_post)↓
4 and↓
5 t0_s0_post = t0_s0_pre↓
6 and↓
7 t0_r0 = false and t1_r0 = true↓
8 ↓
```

図 15 トグルカバレッジの向上制約例

また，MC/DC で入力 In0 が出力 Out に独立に影響していることが未網羅であれば，カバレッジ向上制約は図 16 のようになる．トグルカバレッジと同様に，MC/DC も 2 つのテストケースが必要なため，1 から 3 行目で 2 つの回路制約を定義している．MC/DC の場合はトグルと異なり，必ずしも 2 つのテストケースを連続して実行しなくてもよいため，図 15 の 5 行目に相当する制約は不要である．5 から 6 行目の制約式が，入力 In0 が出力 Out に単独で寄与するこ

とを表している．

```
1 trans(t0_In0,t0_In1,t0_s0_pre,t0_Out,t0_s0_post)↓
2 and↓
3 trans(t1_In0,t1_In1,t0_s0_pre,t1_Out,t0_s1_post)↓
4 and↓
5   (t0_In0 != t1_In0) and (t0_In1 = t1_In1) and ↓
6   (t0_s0_pre = t1_s0_pre) and (t0_Out != t1_Out)↓
7 ↓
```

図 16　MC/DC の向上制約例

カバレッジを向上させるテストケースは，回路制約とカバレッジ向上制約を満たす入力値の組み合わせである．そのため，これら制約を制約ソルバで解く事により，所望のテストケースを得ることができる．

(3)　テストパターンの実行方法

ミュータントにテスト入力を与え，実行結果を得る．この際，プログラムが内部状態を持つことに注意する．直前のテストケースを実行して遷移した状態が，これから実行するテストケースの事前状態と等しければ，入力のみを与えればよいが，等しくなければ，これから行うテストケースの事前状態の値もテスト入力として与える必要がある．

(4)　比較方法

得られた結果と，テストパターンの出力値が同じであるか比較する．出力が 1 つでも異なる場合は，ミュータントを識別できたと判断する．

4.3　評価結果

表 4　評価に用いたプログラム

プログラム					ミュータント	
ID	命令数	入力数	状態数	出力数	And_Or	Not
P1	165	36	19	26	51	162
P2	158	37	14	25	48	155
P3	115	24	10	23	33	104
P4	115	14	7	11	20	58
P5	113	29	10	18	34	103
P6	58	13	10	9	16	50

表 4 は評価実験で用いたプログラムの特徴を整理した表である．これらプログラムは全て実際の発電プラントの制御プログラムの一部である．命令数はプログラムに含まれるファンクション・ブロックの総数である．ミュータント数の And_OR と Not は，それぞれ「And ゲートと OR ゲートの付け間違い」と「Not ゲートの付け忘れ」に相当するミュータントの総数である．

これらプログラムに対し，MTC 網羅，MC/DC 網羅，トグル網羅のテストパターンをそれぞれ 3 つ作成し，ミューテーションスコアを計測した結果が，図 17 である．

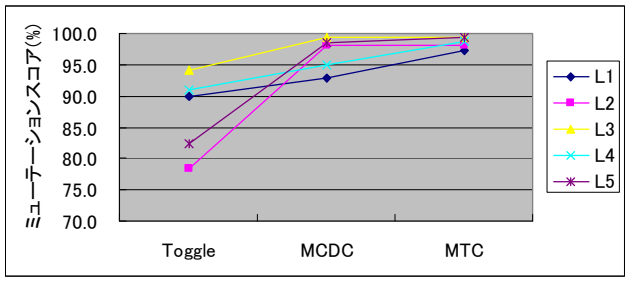

図 17　ミューテーションスコア結果

MTC 網羅，MC/DC 網羅，トグル網羅の順でスコアが高い傾向が見られた．トグル網羅と MC/DC 網羅のスコアの差に比べ，MC/DC 網羅と MTC 網羅のスコアの差が小さいことから，MC/DC 網羅率が特に有効に働いていることが分かる．これは，出力への影響をみる MC/DC の性質が効いていると考えられる．トグルカバレッジは個々の信号線の値が変化するところまでみるが，出力まで影響するかどうかは考慮しない．そのため，演算結果の違いが出力まで到達し，表面化しないケースもある．一方，MC/DC は出力までの影響をみて，表面化するところまで考慮するため，MC/DC 網羅のテストの方が不具合を検出しやすいと考えられる．

表 5　プログラム P5 のテストパターン

テストケース種類	平均テストケース数	平均トグル網羅率(%)	平均MCDC網羅率(%)	平均ミューテーションスコア(%)
Toggle	166.7	100.0	11.1	82.5
MCDC	104.7	5.0	91.8	98.5
MTC	264.8	100.0	91.8	99.3
Rnd100	100.0	4.1	8.8	78.6
Rnd200	200.0	4.1	10.3	81.4
Rnd400	400.0	4.4	12.1	83.2
Rnd800	800.0	4.4	12.7	83.2

プログラム P5 に対し，生成したテストパターンのテストケース数，トグル網羅率，MC/DC 網羅率，ミューテーションスコアを整理したのが表 5 である．テストケース種類の Toggle，MCDC，MTC はそれぞれ，トグル網羅，MC/DC 網羅，MTC 網羅になるように生成したテストパターンである．MC/DC 網羅率が 100%にならないのは，固定値が設定される入力線があるためであり，それを除くとカバレッジ網羅している．

比較対象としてランダム生成したテストパターンを加える（Rnd100，200，400，800）．語尾に付いた数字はランダム生成したテストケース数である．

全ての種類のテストはそれぞれ 3 パターン用意し，表 5 にはその平均値を記載している．

表 5 より，ランダム生成したテストはテスト数を 100 か

ら 800 と 8 倍増やしても，ミューテーションスコアは 5%程度しか向上しないことが確認できる．これは，やみくもにテストを行っても，効率的に不具合を見つけることができないことを示唆している．同様にランダムテスト数を増やしても，カバレッジの網羅率が殆ど上がっていないことが確認できる．テスト数を増やすだけでは，カバレッジの網羅率を容易に向上できないことを示唆している．

また，テストケース数が少ないランダム生成（Rnd100）によるスコアにおいても，8 割近くのミュータントを見つけることが確認できる．これは発見しやすいミュータントが多数含まれているためと考えられる．例えば，出力に直結した信号線に Not ゲートを付加したミュータントの場合，その出力の値は必ず反転するため，どんなテストケースでも違いを検出できるミュータントとなる．今回のミュータントは機械的に作成したものであり，このように発見が容易なミュータントも多く含まれている．そこで，MTC においても発見されないミュータントに注目し分析してみる．今回実験したどのプログラムにおいても，そのようなミュータントは存在しており，

・　内部状態にのみ繋がる信号線に対するミュータント
・　出力までの伝播ルートが他に存在するミュータント

といった特徴を持つミュータントであった．

前者は，状態変数の入力に Not ゲートを付けたミュータントが代表例である．今回我々が採用した MC/DC の基準では，内部状態を含む各入力が出力への影響を考慮するが，事後状態までは考慮しない．そのため，必ずしも見つけることができなかったと考えられる．しかし，仮に事後状態までの伝播を考慮しても，内部状態の変化に留まり，出力まで変化が伝播しなくては，検出できないことに変わりはない．

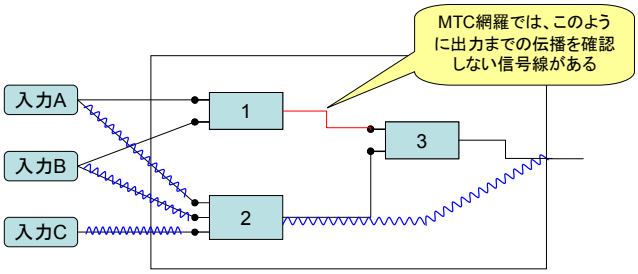

図 18　複数の伝播ルートがある例

後者は図 18 のように入力から出力に複数の伝播ルートがある場合，MC/DC 網羅ではどれか一つのルートを確認して要れば網羅できるため，残りの伝播ルート上に不具合が存在した場合，出力まで現れず検出されない．

上記の両タイプのミュータントに対して，MC/DC 網羅のみのテストパターンでは，改変された信号線やファンクション・ブロックに関わる値が変化することを考慮に入れていなく，これらのミュータントを検出できないケースが散見された．トグル網羅では，各信号線の値が変化するところまでは考慮するため，MC/DC 網羅よりも検出しやすい傾向にある．今回の実験では，MC/DC 網羅で発見できなかったミュータント 26 個の内 12 個をトグル網羅で発見した．この結果は，MC/DC 網羅にトグル網羅の観点を加えた MTC 網羅は，MC/DC 網羅により検出できなかった不具合を見つけうることを示唆している．とはいえ，トグル網羅の観点を加えても発見できないミュータントが存在する．これらは，トグル網羅が全ての信号線の値の変化を見るが，その変化が出力まで伝播し，表面化するかまで考慮していないためと考えられる．ゆえに，各信号線の値の変化が出力まで伝播することを考慮に入れることで，より適切なカバレッジ基準を策定できると考えられる．

5.　おわりに

本論文では，シーケンス制御プログラミング言語に適した新しいカバレッジ基準 MTC を提案した．MTC は，論理回路におけるトグルカバレッジ，ソフトウェアにおける MC/DC の性質を併せ持つことを特徴とする．発電プラント向けの実プログラムをモチーフに，ミューテーションテスト技法を用いた評価を行い，MC/DC 網羅のテストパターンが高いミューテーションスコアであることや，MC/DC 網羅のテストパターンで発見できないミュータントは，トグル網羅の観点を加えることで発見しやすくなることを確認し，カバレッジ基準 MTC を満たすテストパターンが有効であることを示した．

今後は，トグルカバレッジでは考慮しない，信号線の値の変化が出力まで伝播するかを考慮することにより，更に適切なカバレッジ基準の策定に取り組む．

参考文献

1)　International Electrotechnical Commission : IEC 61131-3 (Ed. 2.0), Programmable controllers – Part 3: Programming controllers – Part 3: Programming languages (2002).
2) 日本規格協会: JIS B 3503，プログラマブルコントローラ プログラム言語（1997）
3) D. Drako, P. Cohen : HDL Verification Coverage, Integrated System Design (1998).
4) J. J. Chilenski, S. P. Miller: Applicability of modified condition/decision coverage to software testing, Software Engineering Journal Vol. 9, No. (1994).
5) 松本充広: MC/DC による現実的な網羅のススメ, CATS(2009) .
6) DeMillo, R. et al. : Hints on Test Data Selection: Help for the Practicing Programmer, Computer, vol 11, pp 34-41 (1978)
7) New York University Computer Science Department, CVC3, http://cs.nyu.edu/acsys/cvc3/
8) P. Ammann, J. Offutt: Introduction to software testing, Cambridge university press (2008).