B-031

ミューテーション解析における等価ミュータント検出機能の実装
Implementation of Equivalent Mutants Detection Function in Mutation Analysis

上芝 貴也† 末広 暁久† 芳賀 博英†
Takaya Ueshiba Akihisa Suehiro Hirohide Haga

1. はじめに

今日では，スマートフォンや自動車，鉄道システムなど，さまざまなところでソフトウェアが利用されている．ソフトウェアに致命的なバグが含まれていると，大事故につながりうる．そのため，ソフトウェアには高い品質が求められている．ソフトウェアの品質の高さを示すためには，バグ検出能力の高いテストセットを使って，テストをしなければならない．

ミューテーション解析は，テストセットの品質を評価する手法である．この手法では等価ミュータントというものが生成されることがあり，これは品質評価の妨げとなるので取り除く必要がある．ところが，人間による等価ミュータントの除去には，時間がかかる，ミスが起こりうるなどの問題がある．また，ミューテーション解析によるテストケース自動生成の研究があるが[1]，等価ミュータントを検出されなかったミュータントとして扱っているため，テストセットの品質を正しく評価できない．

これらの問題を解決するため，Offutt の研究[2] を基にして，等価ミュータントを検出するツールを作成した．

2. 提案手法

本論文で提案する手法は Offutt の手法[2] を基にしている．なお，1 つのミュータントへの変更は 1 つとし，ミューテーション解析の対象とするプログラムは C 言語で記述されたものとしている．

2.1 ミューテーション解析の概要

ミューテーション解析はテストセットを評価する手法である．テストセットの対象となるプログラムをわずかに変更することで故意にバグを埋め込み，そのバグをテストセットが検出できるかによって評価する．変更前のプログラムをオリジナル，変更後をミュータントという．また，どのようなバグを埋め込むかという規則を，ミューテーションオペレータと呼ぶ．あるテストケースについてオリジナルとミュータントを実行したとき，それらの出力が異なれば，そのテストケースはそのミュータントのバグを検出したということになる．このとき，そのテストケースはそのミュータントを検出したという．

ミューテーション解析の結果として，ミューテーションスコアという，0 以上 1 以下の値が得られる．これは，生成された検出可能なミュータントがどのくらい検出されたかを示す値であり，次の式で表される．

$$\text{ミューテーションスコア} = \text{検出数} / (\text{生成数} + \text{等価数})$$

このミューテーションスコアが，ミューテーション解析によって評価されたテストセットの品質を表す．なお，等価数とは生成された等価ミュータントの数を表す．

†同志社大学 Doshisha University

ミュータントの生成時，構文的にはオリジナルと等価ではないが，意味的には等価であるミュータントが作られることがある．これは等価ミュータントと呼ばれる．等価ミュータントは，任意のテストケースに対して出力がオリジナルと同じになるため，どんなテストケースもそのミュータントのバグを検出できない．したがって，ミューテーションスコアを正しく算出するためには，等価ミュータントを取り除かなければならない．

2.2 検出方法

ミュータントは，バグが埋め込まれた箇所，すなわち変更箇所以外は，オリジナルと全く同じプログラムである．そのため，あるテストケースについて，ミュータントの出力がオリジナルのそれと異なるためには，変更箇所が実行されなければならない．また，その変更箇所の式や文が評価されたとき，その評価結果がオリジナルとミュータントで異なっていなければならない．

以上をもとに，以下の 2 条件を定義する．

・到達条件：
　ミュータントの変更箇所が実行されるための条件．
・必要条件：
　ミュータントの変更箇所の評価結果がオリジナルのそれと異なるための条件．

あるテストケースに対して，変更箇所が実行されないか，もしくは実行されても変更箇所の評価結果が同じならば，オリジナルとミュータントの出力は同じになる．つまり，（（not 到達条件） or （not 必要条件））が真ならば，出力はおなじになる．言い換えれば，（到達条件 and 必要条件）が，偽，ならば，出力は同じになる．この（到達条件 and 必要条件）を制約条件と呼ぶ．

あるテストケースを制約条件に当てはめたときに偽となれば，そのテストケースに対して，オリジナルとミュータントの出力は同じになる．もし任意のテストケースに対して，制約条件が偽となれば，任意のテストケースに対して出力は同じになる．すなわち，そのミュータントは等価ミュータントである．

図 1 の例で考える．

```
1 | int f(int x, int y) {
2 |     if (x > 10) {
3 |         return x;
4 |     } else {
5 |         return y;
6 |     }
7 | }
```

図 1　関数 f（3 行目の x を abs(x) に変更すると等価ミュータント）

この関数において，3 行目の x を abs(x) に変更したミュータントを考える．このとき，各条件は以下のようになる．

・到達条件： x > 10
・必要条件： x ≠ abs(x)，すなわち， x < 0

よって，制約条件は x > 10 and x < 0 となる．これを満たす x は存在しないため，任意のテストケースに対して制約条件は偽となる．したがって，このミュータントは等価ミュータントである．

次に，3 行目の x を y に変更したミュータントを考える．このときの各条件は以下のようになる．

・到達条件： x > 10
・必要条件： x ≠ y

制約条件は x > 10 and x ≠ y となる．これを満たすテストケースには，例えば(x, y) = (20, 30) などがある．よって，この場合のミュータントは等価ミュータントではない．

このように制約条件の真偽を見ることによって，等価ミュータントか否かという判定が可能になる．

2.3 制約条件の取得

ソースコードからミュータントの制約条件を得るには，構文解析が必要である．論文[1]を参考にして C 言語のプログラムを XML に変換することで構文情報を得る．

3. 評価実験

作成したツールによってどのくらいの等価ミュータントが検出されるかを調べるために，評価実験をした．実験には 2 種類のプログラムを用いた．一つは，int 型の値を 3 つとり，中央値を返す mid である．もう一つは，int 型の値を 3 つとり，それらによって作られる三角形の型を返す trianglетype である．それぞれのプログラムについて，ミュータントを生成し，等価ミュータントを人間の手で調べたのち，作成したツールで検出した．なお，利用したミューテーションオペレータは表 1に示す．これらは，ミュータント生成ツール[1]の改良版で利用されているものである．

表 1　使用したミューテーションオペレータ

名前	意味
ABS	Absolute value insertion
AOR	Arithmetic operator replacement
ROR	Relational operator replacement
LOR	Logical operator replacement
ASR	Assignment operator replacement
UOI	Unary operator insertion
SVR	Scalar variable replacement

実験結果は表 2のとおりである．平均で約 27 % の等価ミュータントが検出された．また，等価ではないミュータントを，等価ミュータントであると判断してしまうような誤検出はなかった．

表 2　実験結果

名前	生成数	等価数	検出数	検出率
mid	151	15	0	0%
triangletype	444	44	24	54.5%

4. 考察

検出できなかった等価ミュータントには，制約条件に関数のローカル変数が含まれるものが多かった．

評価実験で利用したプログラムは行数が大きいため，図 2 の短い例で考える．

```
1 | int max(int x, int y) {
2 |     int result = x;
3 |     if (x < y) {
4 |         result = y;
5 |     }
6 |     return result;
7 | }
```

図 2　関数 max（3 行目の x を result に変更すると等価ミュータント）

このプログラムの 3 行目の x を result に変更すると等価ミュータントになる．制約条件は True and x ≠ result，すなわち， x ≠ result である．直前の 2 行目で result に x を代入しているので，変更箇所を実行するときには必ず， x = result である．よって，制約条件は x ≠ x であり，常に偽であるから，等価ミュータントである．しかし，作成したツールではこのような等価ミュータントを検出できない．ローカル変数を，関数の引数からなる式に書き換えれば検出できる．

5. おわりに

等価ミュータントはミューテーション解析によるテストセット品質評価の妨げとなるため，取り除く必要がある．従来は人間が取り除いていたため，時間がかかったり，誤検出してしまったりする問題があった．この論文では，Offutt の研究をもとにして，等価ミュータントを検出するツールを作成した．評価実験では，生成された等価ミュータントのうち，平均で約 27 %が検出された．全体の 4 分の 1 くらいであり，残りのおよそ 4 分の 3 は人間が検出しなければならないため，依然として，人間による検出が占める割合のほうが高い．検出率を高めるためには，考察で述べたように，ローカル変数を関数の引数からなる式に書き換える必要がある．しかし，この研究の成果によって，等価ミュータントを検出する作業の一部を自動化することが可能となった．

参考文献

[1]末広 暁久, 佐々木 亮太, 芳賀 博英, “XML によるプログラムの表現とそれに基づく統合テスト支援環境の提案”, 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス, Vol.110, No.61, pp. 33-38 (2010).
[2]A.Jefferson Offutt, Jie Pan, “Automatically Detecting Equivalent Mutants and Infeasible Paths”, The Journal of Software Testing, Verification, and Reliability, Vol.7, No.3, pp. 165-192 (1997).