B-001

データフロー解析を用いた命令網羅のためのスタブ生成手法の提案
Proposal of A Stub Generation Method for Instruction Coverage using Dataflow Analysis

山本 一道† 曾我 遼† 鹿糠 秀行†
Kazumichi Yamamoto Ryo Soga Hideyuki Kanuka

1. はじめに

元となる COBOL プログラムとそれから自動リライトした Java プログラムの機能等価性を評価するために COBOL プログラムから C0 網羅単体テストケースを自動生成し、生成テストケースを Java プログラムに適用して現新一致確認するシステムを開発している[1]。

解析対象 COBOL 関数からの呼出先を置き換えるスタブで、戻り値を持ちうるすべての変数に値を設定すると解析対象関数内の変数が上書きされ、検証が阻害される場合があった。

本稿では、解析対象関数のデータフローを解析し、命令網羅に必要最小限なスタブ内設定変数を特定する手法を提案する。

2. 課題

2.1 テストケース自動生成でのスタブ生成の流れ

テスト実行時、メインルーチン内呼出命令から呼び出されるサブルーチンの戻り値を置き換えるスタブ項目表(戻り値変数とその値の組み合わせの表)を用意する。

一手法としてサブルーチン内ロジックを記号実行により分析し全ての戻り値を定める方法がある[2]。記号実行の目的は命令網羅のための入力値の組み合わせを得ることだが、値の変化する変数を特定し、得た入力値を与えた場合の変数の値を決定することができる。サブルーチン内ロジックの記号実行により、戻り値とその値を決定できるが、サブルーチンへの入力値は命令網羅のためにメインルーチンのロジックとは独立に定めたものであるから、戻り値の値はメインルーチンで期待している値とは乖離している可能性が高いことに注意が必要である。

2.2 スタブ生成における課題

記号実行で得たサブルーチンの戻り値をスタブ内設定変数の値としてしまうと、変数値とテストケースで定める期待値との乖離を生じテストが失敗する場合があった。

図 1-A に例示する COBOL プログラムは MONTH と DAY の 2 つの変数を初期化し、サブルーチン CHECK-DAY を呼び出している。CALL 文による呼出ではサブルーチン内のロジックはメインルーチンとは別スコープで、参照渡しの引数である上記 2 つの変数以外は共有せず、また 2 変数とも戻り値を持ちうる。このように呼出命令文による呼出先サブルーチンの作用によって値が上書きされる可能性のある変数を「ダーティ変数」と呼ぶことにする。例示のプログラムでは MONTH と DAY がダーティ変数である。

2.2.1 変数の値の上書きによる検証漏れの発生

記号実行により CHECK-DAY 内を分析すると、戻り値は決まるもののメインルーチンのデータフローとは関係なくランダムな値になりうる。MONTH の期待値は図 1-A の #2 の代入命令文の結果により定まっているが、これをランダムな値で上書きすることとなり期待値との乖離を生じテストが失敗する原因となる (図 1-B-(i)) 。

2.2.2 変数の値設定不足による命令網羅不足

スタブ内設定変数により戻り値を一切指定しない(引数の初期値のままとする)と、CALL 文の後の分岐命令文の TRUE 節を網羅することができない(図 1-B-(ii))。

3. 提案手法

メインルーチンのデータフロー解析により、C0 網羅に必要最小限のサブルーチンからの戻り値を持つ変数を特定する。

呼出命令文の後の分岐命令文の条件文にダーティ変数がある場合には、その変数のフロー解析を行い、テストケースで初期値を与えることによりその値を制御できるかどうかを判断する。テストケースから制御できない場合には呼出先サブルーチンを置き換えるスタブにてスタブ内設定変数を定義する必要がある。

3.1 フロー解析

図 2-A のフロー解析説明図に示すように、DAY にはプログラム#3 でリテラルとして 5 が代入されるため、上記ケースに該当しスタブ内設定変数として DAY の値を定義する必要があることがわかる。

図 1. 単体テストにおけるスタブでの設定変数の定義と課題

† (株)日立製作所 研究開発グループ Hitachi, Ltd., Research & Development Group

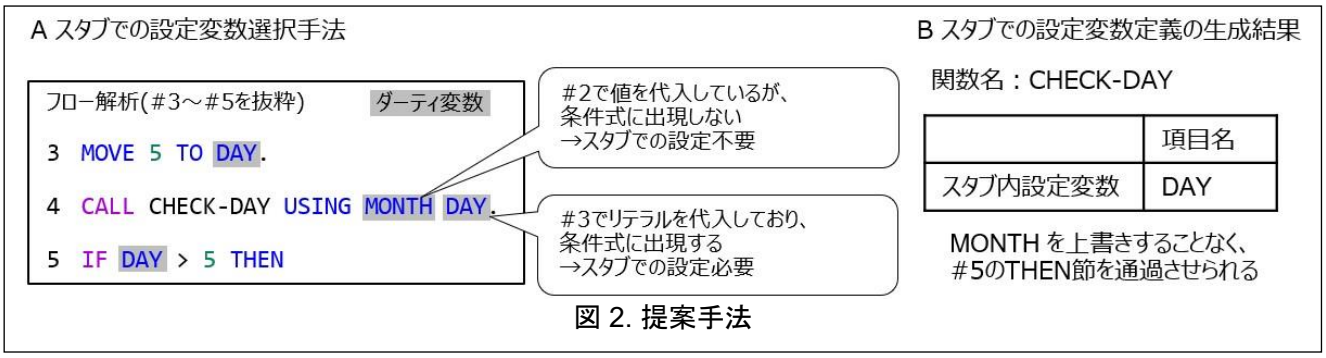

図 2. 提案手法

また、MONTH についてはプログラム#2 の代入命令文の結果から、その期待値は 1 となっている。もしスタブ内設定変数として MONTH を上書きしてしまうとテスト失敗の原因となりうるので MONTH はスタブ内設定変数としない。

3.2 スタブ内設定変数の出力

従って、図 2-B に示すようにサブルーチン CHECK-DAY を置き換えるスタブでは、スタブ内設定変数として DAY だけを定義すればよいことがわかる。

結果として、呼出先サブルーチン内部ロジックを一切分析することなく C0 網羅テストに必要十分なスタブを得ることができる。

4. 手法の評価

実際の COBOL→Java リライト案件において、2 章で挙げた課題を 3 章で提案したスタブ内設定変数の特定手法を適用することで解決できるかを評価した。下記の RQ(Research Question)に従って、ケーススタディを通じた実験とその結果を考察する。

RQ. 提案手法によってスタブ内設定変数によるテスト対象関数の命令実行結果の上書きを削減できたか？

4.1 実験方法

例にとった解析対象関数には 1 件の CALL 文があり、参照渡し引数が 10 件ある。CALL 文の後に分岐命令文があり、その条件文で呼出先からの戻り値を参照し、条件評価結果により代入命令文を実行するかどうかを決定する。参照渡し引数 10 件のうち 9 件を初期化専用の代入命令文で初期化する。初期化しない 1 件の変数についてはテストケースで値を制御可能なのでスタブ内設定変数としない。

条件文内で参照する変数 1 件は初期化専用の代入命令文で初期化されるため、その値をテストケースで制御できない。

4.2 実験結果

提案手法に従えば、条件文内で参照する変数は解析対象関数の C0 網羅に影響を与え、かつその値をテストケースで制御できないためスタブ内設定変数としなければならないことがわかる。

この 1 件以外の 8 件の変数に関しては、スタブ内設定変数として値を上書きしてしまうと期待値との乖離を生じテストが失敗する可能性があるため値を設定してはいけない。

RQ への回答として、表 1 に示すように C0 網羅に必要のない解析対象関数の命令実行結果の上書きを 8 件削減したことを確認した。

表 1. 設定変数の選択手法の適用結果

	値を上書きする変数の数
全ての引数に値を設定した場合	9
設定変数の選択手法を適用した場合	1

5. 結論

C0 網羅単体テストにおいて、提案手法によりサブルーチンをスタブに置き換える際に対象サブルーチンの内部ロジックを解析することなくスタブ内設定変数を削減できることがわかった。

COBOL 解析対象関数から C0 網羅単体テスト用のテストケース定義を自動生成するにあたり、スタブ内設定変数として不必要な変数の設定を回避できるため、テストケースで定義している期待値を上書きする等、検証を阻害する恐れが無くなることを確認した。

参考文献

[1] 木下崇央、曾我遼、大原貴都、野尻周平、鹿糠秀行、齋田雄一 、栗田繁、 “言語変換されたプログラムの現新比較テスト自動化方式,”、電子情報通信学会 2021 年総合大会(D-3-2)、2021.3.

[2] R. Soga, T. Yonemitsu, M. Inagaki, Y. Fujisaki, H. Sugou and H. Kanuka, "A Program Simplification Method for Generating Test Input Values Using Symbolic Execution," 2020 27th Asia-Pacific Software Engineering Conference (APSEC), Singapore, Singapore, 2020.