2N-03

記号実行支援ツールの開発

木村　信一　　劉　少英

法政大学情報科学部コンピュータ科学科

1　はじめに

記号実行とはプログラムの入力を記号で表現し，入力に対しどの実行経路が選択されるかを解析する手法である．既存研究では対象プログラムをバイトコードへ変換しコマンドライン上での解析を行うツールがほとんどで，実行手順の理解が難解なものが多くある．そこで本研究のツールでは GUI 化により操作感や解析手順の理解の向上を図り，バイトコードへの変換をせず文字列データのまま解析を行う手法を取る．対象プログラムの入力変数の数を 1 つや数値型に制限することで処理の簡易化を図り低速化へ対応を行う．主機能は実行経路を抽出し記号実行により対応した入力のテストケースを出力する機能であり，選択した経路に対しての実行や全経路への一括実行を実装する．

これにより理解しやすい記号実行ツールを開発し，将来的には拡大する AI・IoT のシステム開発現場でのデバッグ・自動テストへ導入され、記号実行の発展への貢献となることを目指した．

2　関連研究

記号実行で代表的なツールには，KLEE[1]や，Java Path Finder[2]がある．Java Path Finder は NASA の研究所によって開発され 2005 年にオープンソース化されたツールである．記号実行を用いてデッドロックや競合状態の検査が可能で，火星探査機の制御システムの検証に利用された．これを文字列処理に拡張し Java アプリケーションの品質向上に貢献した富士通の技術[3,4]も存在する．これらのツールはコマンドライン上の実行となり初見では実行法が難解な場合がある．本研究では GUI による実行で操作の簡易化や分析手順の可視化を図り，使いやすいツールを目指し開発を行った．

3　システム機能

本研究で行う主な処理は，対象プログラムの文字列データに対し実行経路を抽出し，入力変数を記号で表現することで，経路による出力を記号の含む式で表現することである．

Development of a supporting tool for Symbolic Execution
Shinichi Kimura, Shaoying Liu, Hosei University, Faculty of Computer and Information Science

具体値入力は必要なく抽象的な記号のみで疑似実行が可能なため，エラーやバグがあっても安全にテストできる利点がある．

3．1　経路抽出

プログラムの実行経路は入力変数の関わる代入文・条件文のフローで表現する．このとき入力変数を記号のまま抽象的に処理することで入力に対応した経路を表現することが可能である．図-1 は経路抽出の一例を示した図である．

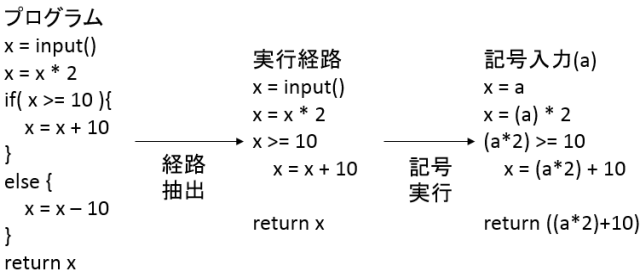

図-1　経路抽出の一例

3．2　主機能紹介

システムの主な機能として単一経路実行・全経路実行・具体値に対する経路探索・テストケース生成を実装した．

単一経路実行ではユーザーに if 文を選択させ，選択した if 文を true とする実行経路を抽出する． 抽出した経路に対応した入力が存在する場合テストケースが生成される．図-2 は単一経路実行時の GUI 例である．

全経路実行では if 文の選択を全網羅し，各経路に対し記号実行を行う．経路情報やテストケースはまとめて 1 つのテキストファイルへと出力される．

具体値に対する経路探索では具体的な数値を指定し，その値を入力とした場合に実行される経路を特定し，その経路の情報やテストケースを出力する．この実行も文字列処理による疑似実行となるためテストによる安全性は保障されている．

4　評価テスト

分析対象のプログラムの入力を 1 つかつ数値型に限定しているため，テスト対象として単純な数値計算のプログラムを用意し，各メソッド単位で探索できた経路数・テストケースの正確

性を評価指標とする．具体的には Java の標準ライブラリ Math から四則演算や三角関数に関するメソッドをいくつか選びそれを呼び出すメソッドや，分岐数の多いもの・入れ子構造・到達不可分岐などのあるメソッドを対象とする

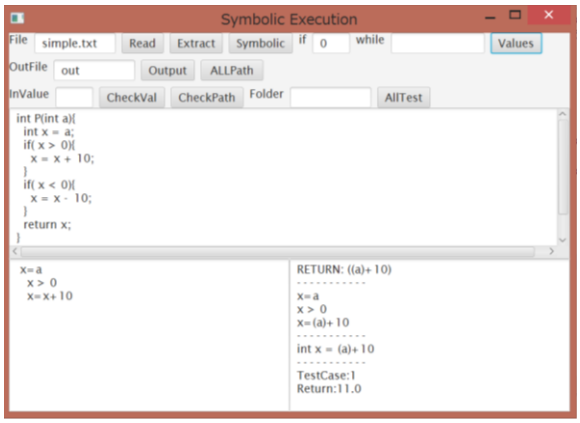

図-2　単一経路実行例

4．1　テスト結果

分岐カバレッジに関して，表-1 は分岐選択に関するテスト結果の表である．実経路数は実行可能な経路の総数，探索経路数はシステムで探索できた経路数，到達不可経路数はテストケース生成の際に到達不可と判定された経路の数，テスト誤りは生成したテストケースが誤っていた経路の数である．

表-1　実験結果

メソッド名	実経路数	探索経路数	到達不可経路数	テスト誤り
Manycond	20	95	75	14
Elseif	4	4	0	0
Nesting	5	8	0	3

5　考察

manycond(分岐数の多いメソッド)に関して，到達不可経路は x>0 かつ x<0 のような矛盾した条件の含む経路をはじいた処理のため，正常に動作したと言える．しかし経路抽出の際に選択した条件式のみを参照してテストケース生成を行ったため，選択していない条件を満たしてしまうテストケースが多く生成された結果，テスト誤りが多く出てしまった．誤生成された例では，x>10 のみを選択した経路でテストケース 11 が生成されたが，実際には x>0 と x>10 を両方満たすため誤りとなった．この場合到達不可と判定されることが正しい処理であった．

改善策はテストケース生成の際に，選択しなかった分岐条件を参照し，false とする判定を追加する方式に変更することで対処可能と考える．

nesting(入れ子構造)に関して，外側の分岐を選択した場合には内側全てが true とされ，内側の分岐を選択した場合は内側のみが true となった．実際には外側のみの経路も存在し，内側のみの経路は存在しないためテスト誤りとなった．

改善策は矛盾する場合は経路選択の際にはじく等，elseif の際に実装した処理と似た処理を追加することで対応可能と考える．

6　おわりに

本研究で行った記号実行支援ツールの開発では，基本的な記号実行を行うシステムを実装し，GUI 化によって操作性の簡易化を図るほか，プログラムの実行経路を入力変数に関する条件文・代入文によって示すことで，記号実行の一連の動作の可視化を行った．

本研究の課題として，経路選択が分岐の番号を指定する方式で行うことや，テストケース生成の正確性が低いことなどが挙げられる．考察の章で記述したように，テストケース生成の判定方式の変更や入れ子構造の分岐へ対応を行うことで完成度を上げる必要がある．また GUI のデザインも複数のウィンドウを工夫することやメニューにまとめるなど，更に使いやすく改良が可能である．また入力変数の数や型を限定しているため，将来的な改善策としては文字列型や Point など複雑な型への拡張や入力の数の制限緩和，未対応な文法に対する処理の実装などが挙げられる．他メソッドの呼び出し処理への対応やデータベースの参照なども追加することで，再帰構造など複雑な処理が分析可能となる．

これらの将来課題を解決することで，システム開発の現場へ導入可能なほど完成度を向上させることでプログラム解析や記号実行の発展に貢献できると考える．

参考文献

[1] The KLEE Team , KLEE , ” http://klee.github.io/ , 2017 (参照 2018/12/20).

[2] NASA Open Source Agreement version 1.3 , Java Path Finder , http://javapathfinder.sourceforge.net/ , 2005, (参照 2018/12/20) .

[3] Fujitsu Laboratories of America, Inc. , Web アプリケーション向け品質保証の基礎技術を開発 , http://pr.fujitsu.com/jp/news/2008/04/4-1.html, 2008/4/4 , (参照 2018/12/22).

[4] Fujitsu Laboratories of America, Inc. , Java プログラムを網羅的に検証する技術を開発 , http://pr.fujitsu.com/jp/news/2010/01/12-1.html,2010/1/12 , (参照 2018/12/22).

[5] Liu Shaoying, “Formal Engineering for Industrial Software Development Using the SOFL Method” , Springer-Verlag, March 2004.