様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　３０　年　　６　月　　４　日現在

機関番号：　１１５０１
研究種目：　挑戦的萌芽研究
研究期間：　2015～2017
課題番号：　１５Ｋ１１９８９

研究課題名（和文）ソフトウェアの欠陥発見のための逆方向シミュレーション

研究課題名（英文）Backward simulation for software defect discovery

研究代表者
平中　幸雄（HIRANAKA, Yukio）

山形大学・大学院理工学研究科・教授

研究者番号：４０１３４４６５

交付決定額（研究期間全体）：（直接経費）　　2,700,000円

研究成果の概要（和文）：ソフトウェアの網羅的安全性検証を、逆方向シミュレーションを使って行う方法を研究した。具体的には、Java プログラムを対象に機械語に相当するバイトコードを逐次実行しながら、プログラム中のバグや誤動作要因を検出するシミュレータを実現した。逆方向シミュレーションを使うことで効率的に検出できるが、逆算方法として数値範囲分割方式とシンボリック分析方式を試み、それぞれの特質と課題に関する知見を得た。

研究成果の概要（英文）：We have studied a method to verify comprehensive safety of software by using backward simulation. Specifically, we realized a method of detecting bugs in programs and malfunction factors while sequentially executing bytecodes corresponding to machine language for Java programs in the backward direction. We have tried a numerical range division method and a symbolic analysis method as the backward execution method, and obtained knowledges about characteristics of each methods.

研究分野：　情報ネットワークとシステムの安全性

キーワード：　ソフトウェア安全性検証　逆方向シミュレーション　逆実行モデル　数値範囲分析　シンボリック逆実行　Java bytecode

１．研究開始当初の背景

プログラムにバグは付き物である。バグの無い完全なプログラムの作成は困難と考えられている。バグを見つけるにはテストが効果的とされているが、すべての状況をテストすることは一般的にはほとんど不可能である。そこでテスト漏れのバグが十分少なくなるようテストを充分に実施することが実質的な安全対策となっている。しかしこれでは、問題を内在するソフトウェアが増えるばかりである。少なくとも安全性の問題を持つかどうかを明確にするような、網羅的なプログラムの検証方法が必要である。

２．研究の目的

網羅的安全性検証方法を目指す究極の方向としては、論理検証と全数テストが考えられる。しかし、論理検証はオーバフローなど理論的に扱いにくい実際的な要因が、現実のプログラムへの適用の障害になりやすい。一方、全数テストは計算時間が膨大になるため、全数テストの適用を検討することすら否定されることが多い。

しかしながら、いずれの方法も、本研究の研究者らがシステム分析で使ってきた、結果から入力に戻る逆方向シミュレーションと組み合わせれば、困難性を大幅に緩和することができるのではないかと考えている。通常のテストはテスト入力に対するプログラムの出力が設計通りであるかどうかを判定するが、逆方向シミュレーションでは図１のように、出力値からスタートし、プログラムを逆実行しながら入力条件を求める。

どちらから始めても分かることは同じはずであるが、一般には入力値の組み合わせに比べ、出力値の種類は少ないのが通例である。そのため、逆方向に想定する出力からスタートし、逆実行結果が想定する入力の条件と一致するかを判定すれば、必要な条件のみをテストすることになり、テストが効率的に行える。たとえば出力がＹとＮのいずれかとなるプログラムの場合、Ｙとなる入力条件とＮとなる入力条件を求めれば、設計通りであるか、条件漏れがあるかが明快になる。

ただし、このときの出力値の与え方と逆実行の実現方法がこの方式の有効性の鍵になる。実際に機械語プログラムを対象とする逆方向シミュレータを実現しながら、方式の有効性検証と効果的な方法の確立を行うことが、本研究の目的である。

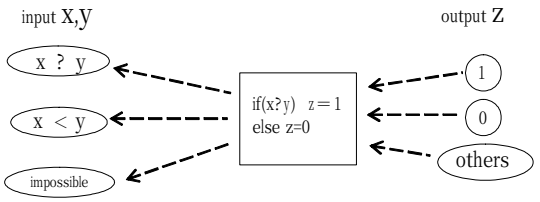


図１　逆方向シミュレーション概念図

３．研究の方法

具体的に、汎用性の高い Java プログラムを対象に、その機械語である Java バイトコードを逆実行しながら、プログラム中のバグや誤動作要因を検出できるシミュレータを作成し、逆方向シミュレータによる安全性検証の効果、特質と性能を評価する。

(1) 逆方向シミュレーションに関しては、本研究以前のシステム特性評価用のシミュレータ研究から、以下のような成果等が得られている。

① 数値範囲による処理と逆算モデルの構成
　シミュレーション開始の出力値を１個の値ではなく、数値範囲とする。これにより有限個の場合分けで網羅的検証が可能になる。もちろん、範囲に対して逆算を行う逆モデルを作成する必要がある。

② 数値範囲シミュレーションの高精度化
　逆シミュレーション途上で無効判定されると、そのときの出力範囲をそれ以後除外することができる。そして有効として残った範囲に対して、次の段階で出力数値範囲を２分の１に狭めていくことで、計算時間を抑制しながら徐々に精密な結果が得られる。

③ 逆方向シミュレータの構成と制御
　Scala 言語を使ってマルチスレッドで、順方向及び逆方向シミュレーションができる基本シミュレータを実現した。シミュレーション要素はオブジェクト化し、オブジェクトの生成、相互通信の形成、分散処理などの機能を簡単に活用できるプログラム作成方法を開発した。さらに、数値範囲や条件組み合わせのすべてを順次実行していくフロー制御も実現した。これらに、機械語実行部分のみを加えることで、プログラム検証用の逆方向シミュレータが実現できる。

(2) 本研究では上の成果を活用しながら、以下の各テーマに順次、取り組んだ。

① 数値範囲分割法
　機械語の逆算で最も困難な点は、２入力１出力演算命令のように、逆算不可能な命令があることである。これに対して例えば乗算命令は、結果が正であれば２入力とも正もしくは２入力とも負、というように入力値に対する限定が適用できることを使って、数値範囲を絞っていく方法を考えた。
　加算命令については図２のように、結果値（右下がりの線上が同じ値を表す）に対する２入力（横軸の値 x と縦軸の値 y）の可能範囲が存在し、結果値を範囲で表せば２入力値の組み合わせも有限個の範囲に限定できる。限定できた個々の範囲の組み合わせについてプログラムを逆実行していけば、それらの条件について、プログラム前段で有効か否が判定できる。このとき、x 及び y それぞれの範囲を決

めるため、逆実行処理では可能性のある値を含むように最大範囲の処理をしていく必要がある。

最終的にプログラム開始時点まで生き残った条件が最初に設定した出力値範囲に対する可能な入力条件となる。生き残った条件がなければ、当初の出力値はそのプログラムが出力することはあり得ない、ということになる。この方法で、設計外の出力値に対する入力条件が存在すれば、プログラムの不具合を検出したことになる。図２は出力値に対する２入力の正負の範囲だけを示しているが、加算がオーバフローを起こす場合も表現しており、この方法によりオーバフローを含んだ逆算も実行でき、実際的な機械語ごとの個々の条件にも対応しやすい。

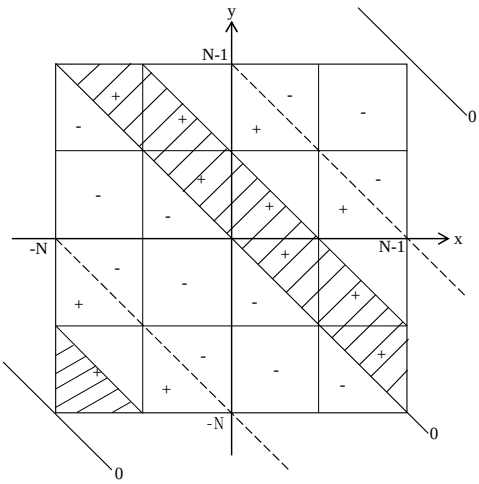


図２　iadd 命令の入出力関係図

② シンボリック分析法

ビット演算など数値範囲分割法では困難が予想されるバイトコードも存在するため、シンボリックな逆算方式も試みる。これは機械語命令の各演算の入力値を変数で表し、出力値（数値もしくは変数）に対して満たすべき条件を数式で表現し、逆方向で得られたすべての条件式を満たす条件を求めるものである。

ただし、逆実行では変数などが不明なまま処理を行う必要があるため、変数値を仮に表１のように記号で表現しておいて、逆実行が完了した時点で各シンボルの値を確定させる、もしくは変数間で成り立つべき関係式として逆方向シミュレーションの結果を得ることになる。このときも、途中で変数の値が何であろうと関係式を満たすことができないことが判明すれば、その時点でその逆実行は不可能、つまりシミュレーションを行った当初の出力値は起こりえないことが判明することになる。

表１　変数のシンボル表現規則

変数タイプ	意味
Lm(n)	ローカル変数 m：変数番号, n：順序番号
ARn	配列変数, n:順序番号
An	配列指数, n：順序番号
Vn	仮変数, n：順序番号

４．研究成果

(1) 数値範囲分割法

整数加算、条件分岐、スタック操作などのバイトコードについて逆実行、数値範囲分割、分割ケースごとの順次実行、最終結果出力、計算時間計測をできるシミュレータを作成した。そして、設定値に対して正しい入力値範囲が出力されることを確認した。

たとえば 32 ビット整数の場合、最初は、シミュレーション全体にわたって、整数範囲を２分割した範囲で逆算していく。このとき逆算精度は１ビットである。逆算が成功した範囲について、さらに２分割、つまり１ビットずつ精度を上げた逆算を行い、最終的に 32 ビット精度の逆方向シミュレーションを行う。したがって、すべての値の組み合わせに対する逆演算を行うより効率が良いが、精度を上げると計算時間が伸びる。

図３はシミュレーション精度と計算時間の関係図の例を示す。計算時間改善策としては、入力範囲を想定される値を含むケースから実行して途中の逆算成功を早期化させる、範囲を対数化して可能性のない領域は一気に棄却するなどの工夫を行った。

いくつかのプログラムに対して、実際に適用可能なことを確認した。精度を上げていくと計算時間は伸びていくが、途中で入力条件が収束してしまい、比較的短時間で結論が出るケースもあった。すべてではないが、実用的な検証方法になることも期待できる。

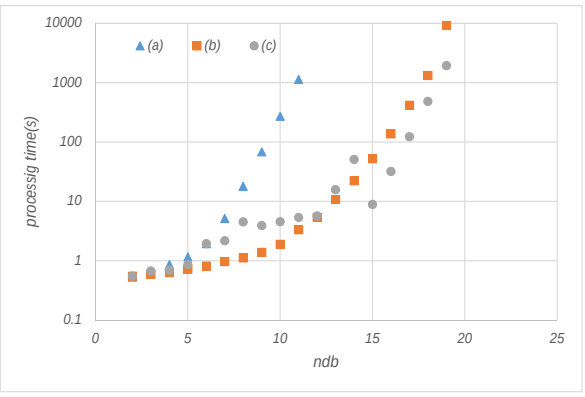


図３　範囲分割と計算時間の関係例（横軸は整数範囲の分割数の対数、縦軸は計算時間）

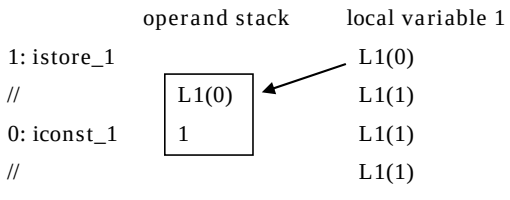
(2) シンボリック逆実行

Java の機械語命令はオペランドスタックを使う単純な仕組みであるので、各命令の実

行前、実行後の変数値に名前を付けて、演算を表す記号を使えば、すべての演算を表現することができ、逆実行も処理しやすい。

外部メソッド呼び出しなどの一部の命令を除き、ほとんどの命令について逆実行処理を実現し、Java クラスファイルの読み込みから逆方向シミュレーション結果の出力まで実行できるプログラムを作成した。

長いプログラムの実行も可能で、数値計算プログラムで可能性のある出力値を求めること、条件判定プログラムで成立する条件式を求めること、ループを含むプログラムでループを逆回りしながら出力値を特定することができることを示した。ただし、ループ処理については、厳密には無限にループの逆回りを処理する必要性がある。これには、最大ループ回数を設定することで実際的に対応することが考えられる。



(b) backward simulation outputs L1(0)=1

図４　変数名と逆実行の状況図（数字で始まる行は逆実行前の状態で、//で始まる行は逆実行後の状態を表す）

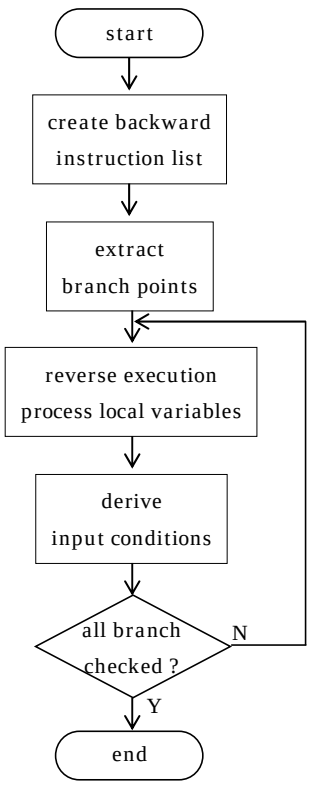


図５　シンボリック逆方向シミュレーションの実行フロー

(3) ハードウェア高速化

数値分割法では計算時間がかかるので、ハードウェア活用による計算時間の短縮も検討した。マルチコアを使うマルチスレッド処理を行っているが、途中で分岐や分岐破棄の処理のために同期の必要性が発生し、作成したシミュレータでは、マルチスレッド化の効果は大きくなかった。スレッド分散にふさわしいか否かの分岐状況の判断を伴う分散処理が必要と考えられる。

また、計画的分散を想定すると MapReduce による分散処理も考えられるが、MapReduce はオーバヘッドが大きく、充分時間のかかることが分かっている単位に分割して分散させる必要があり、本研究のシミュレーションには向かないと判断された。

(4) シミュレーション実行の可視化

シミュレータ処理の監視や操作のための仕組みの検討も行った。web ブラウザによる可視化ツールが用意されている Node-RED を利用したシミュレータも作成した。しかし、Node-RED には分岐ケース間の処理順序が明確でないという問題があり、同期処理が必要になることが分かった。そのために逆方向シミュレータ自体は Scala 言語で実行し、その監視・制御専用の独立したツールとして Node-RED を活用することが期待される。

(5)逆方向シミュレータの将来性

数値範囲分割法とシンボリック分析法は処理の単純性や分岐数による計算時間の影響にそれぞれの特徴があり、検証対象とするバグ等の種類も想定しながら、選択さらには最適な適用順序や組み合わせを検討し、実用的な方法にしたい。

５．主な発表論文等

〔学会発表〕（計２件）

① Tetsuya Inafune, Shinichi Miura, Toshihiro Taketa, Yukio Hiranaka, Symbolic backward simulation of Java bytecode program, Proc. 10th International Conference on Computer Modeling and Simulation (ICCMS2018), 2018.1.9, Sydney(Australia).

② Yukio Hiranaka, Tetsuya Inafune, Shinichi Miura, Toshihiro Taketa, Backward range simulation of Java bytecodes and reduction of its processing time, Proc. 8th International Conference on Computer Modeling and Simulation (ICCMS2017), 2017.1.21, Canberra(Australia).

〔その他〕

ホームページ等

http://eatz.yz.yamagata-u.ac.jp

６．研究組織

(1)研究代表者

平中　幸雄（HIRANAKA, Yukio）

山形大学・大学院理工学研究科・教授

研究者番号：40134465

(2)連携研究者
武田　利浩（TAKETA, Toshihiro）
山形大学・大学院理工学研究科・助教
研究者番号：90236472

(3)研究協力者
三浦　信一（MIURA, Shinichi）
稲船　哲也（INAFUNE, Tetsuya）
門馬　悠太（MONMA, Yuta）