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　２９　年　　６　月　１５　日現在

機関番号：　３４５０４
研究種目：　基盤研究(C)（一般）
研究期間：　2013～2016
課題番号：　２５３３００７３

研究課題名（和文）正確な期待値計算および生成コードの記号実行に基づくコンパイラのランダムテスト

研究課題名（英文）Random testing of compilers based on precise computation of expected values and symbolic execution of generated codes

研究代表者

石浦　菜岐佐（Nagisa, ISHIURA）

関西学院大学・理工学部・教授

研究者番号：６０１９３２６５

交付決定額（研究期間全体）：（直接経費）　　3,800,000円

研究成果の概要（和文）：本課題では，コンパイラの信頼性確保を目的に，Cコンパイラのランダムテストの新しい手法について研究を行った．ランダムに生成したプログラムの期待値をいかに求めるか，および未定義動作を含むプログラムの生成をいかに避けるかという課題を解決するため，プログラムの生成と並行して期待値を計算する方法，およびプログラムの等価変換によって与えられた期待値を出力するプログラムを生成する方法を開発した．また，コンパイラのコードの正当性だけでなく最適化の性能をテストする手法，C以外にLLVMとJavaの処理系をテストする手法，およびCコンパイラの算術最適化用テストスイートも開発した．

研究成果の概要（英文）：This project has conducted researches on new methods for random testing of C compilers. Three challenges in compiler random testing are 1) how to tell the valid outputs of randomly generated programs, 2) how to avoid generating programs with undefined behavior, and 3) how to minimize the error programs to help developers to track down the bugs. We have developed two strategies to overcome these difficulties. One is to compute expected behavior during test program generation and to avoid generating undefined behavior, the other is to generate random programs by equivalence program transformations from trivial seed programs whose behavior is well known. We have also developed methods to detect insufficient optimization by random programs, methods to test LLVM and Java compilers and their back-ends, and a test suite dedicated to test arithmetic optimization of C compilers.

研究分野：　情報処理

キーワード：　コンパイラ　ランダムテスト　テストスイート　等価変換　不具合検出　最適化機会抽出　Orange4　CF 3

１．研究開始当初の背景

ソフトウェア開発の基盤ツールの一つであるコンパイラには極めて高い信頼性が要求される.しかし, http://gcc.gnu.org/bugzillaにも報告されているように, 広範に使用されている GCC にもリリース後に多くの不具合が検出されている.

コンパイラのテストは, テストスイートや実際のプログラムを用いて徹底的に行われるが, これを経てもなお不具合が潜在する. これを検出する言わば最後の手段がランダムテストである. ランダムテストでは, 乱数を用いて生成したプログラムによるテストを時間の許す限り行うものである.

コンパイラのランダムテストに関する当時の主要な研究としては下記[1] [2] [3]があり, GCC や LLVM 等のオープンソースコンパイラの不具合を数多く検出することによってこれらのコンパイラの信頼性向上に貢献している.

[1] Christian Lindig: “Find a Compiler Bug in 5 Minutes,”in Proc. ACM International Symposium on Automated Analysis-Driven Debugging (Sept. 2005).

[2] Eric Eide and John Regehr: “Volatiles Are Miscompiled, and What to Do about It,”in Proc. International Conference on Embedded Software, (Oct. 2008).

[3] Xuejun Yang 他: “Finding and Understanding Bugs in C Compilers,”in Proc. ACM Conference on Programming Language Design and Implementation (June 2011).

ランダムテストによるコンパイラの不具合検出の主要課題は次の３点が挙げられる.

(1) プログラムの正しいと考えられる出力(期待値) をいかにして求めるか.

(2) 不正なプログラム (ゼロ除算, オーバーフロー等の未定義動作を含み, テストに用いることができないプログラム) の生成をいかに回避するか.

(3) エラープログラムの最小化 (エラー検出する最小プログラムへの縮約) をいかに行うか. これらは先行研究[2][3]のようにプログラミング言語の構文を広くカバーしようとする際に大きな課題となる. [2][3]では, 複数のコンパイラ (あるいはオプション) を用いた結果を比較する「differential testing」により(1)の期待値計算を回避している. しかし, (2)の不正プログラムの生成が完全に排除できず, 生成できるプログラムのクラスや規模が限定される. (3) はエラープログラムの分析のために極めて重要であるが, [2][3]ではプログラム最小化の過程で不正プログラムが生成されるため, 最小化の自動化が実現できていなかった.

また, ランダムテストに限らず, コンパイラのテストに置ける大きな課題は, 「特定のデータでしかテストできない」ことである. このため, どれだけテストすれば十分かの保証が難しいということも課題の一つであった.

２．研究の目的

本研究では以上の背景を踏まえ, (A) テストプログラムに期待される動作や出力がわかるようにランダム生成を行う手法, および (B) 記号実行により限定的に網羅的なテストを行う手法を確立することを目的に研究を行った. 主な対象は C コンパイラとした.

(A) 期待値計算に基くランダムテスト生成

プログラムのランダム生成時に, 言語仕様に沿って実装依存項目も含めてプログラムの出力する期待値を忠実に計算することにより, 不正なテストプログラムの生成を回避するランダムテスト手法の開発を目指した. また, あらかじめプログラムの出力すべき期待値を設定し, それを変化させないようにプログラムを等価変換することによりランダムなプログラムを生成する手法についても研究を行った.

これらの手法により, 先行手法では生成できないクラスのテストプログラムを生成することが可能になるとともに, 不具合の原因を突き止めるための最小化の自動化や, 最適化がきちんと行われているかどうかを調べる性能テストも可能になる.

また, これらのランダムテストで有効なテストプログラムを集めることによって, コンパイラのテストスイートを形成することも目指した.

(B) コードの記号実行によるテスト

コンパイラが生成するコードをそのまま実行するのではなく, BDD (二分決定グラフ)等のデータ構造を用いて非明示的に変数の値の全ての組合せについて実行する手法の開発を目指した.

３．研究の方法

研究代表者の指揮の下，研究協力者（研究代表者の研究室の大学院学生）がプログラムの開発，実験・評価，改良を行うという体制で研究を実施した．期待値計算に基づくランダムテスト生成(A)に関しては，これまでに実装して来た算術式対象のテストシステムを拡張・改良する形で開発を進めた．記号実行に基づく方法(B)は，MIPS をターゲットとし，算術式の計算を対象としたテストシステムの実装を進めた．

４．研究成果

期待される動作や出力がわかるテストプログラムのランダム生成に関しては，正確

な期待値計算に基づくプログラム生成法(成果(1))，および，等価変換に基づくプログラム生成法（成果(2)）を開発した．これらの手法を拡張することにより，コンパイラの最適化が意図通りに行われているかをテストする手法も開発（成果(3)）するとともに，C コンパイラ以外に LLVM の中間表現や Java のテストプログラム生成も可能にした（成果(4)）．また，これらのランダムテストシステムを利用して，C コンパイラの算術最適化に特化したテストスイートを開発した（成果(5)）．

これらに基づくテストシステムを実装し，GCC の最新版の不具合 36 件，LLVM の最新版の不具合 22 件を検出し，開発者に報告した．なお，実装したテスト生成システム Orange3，Orange4 およびテストスイート CF3 は，GitHub を通じて無償で公開している．

また，これらの研究成果とともに，現在のコンパイラのランダムテストの研究動向を解説論文として執筆した（雑誌論文⑥）．

なお，記号実行に基づくテスト方法については，顕著な結果を得ることはできなかった．

(1) 正確な期待値計算に基づくランダムプログラム生成法

テストプログラム中の算術式の解析木を生成する際に，言語仕様や実装仕様に基づいて期待値を計算すれば，プログラムが出力すべき値を知ることができる．未定義動作を検出すれば，その式を破棄して再度式を生成すれば良いが，さらに，その未定義動作をなくすように式を修正することもできる．例えば，零除算を起こす式を生成してしまった場合には，分母に非零数を加算するような修正を行えばよい．これによって，未定義動作を全く含まない大規模な式を生成することが可能になるとともに，エラープログラムの最小化も効率的に行えるようになった（雑誌論文⑧⑨）．また，浮動小数点演算など，従来法では難しかったテストも可能になった（学会発表 11）．本手法は算術式の最適化を対象としたものであるが，条件文の導入などの拡張も行っている(雑誌論文⑦)．

Orange3 は本手法に基づき開発した C コンパイラ用ランダムテストシステムであり，GitHub で公開している．

(2) 等価変換に基づくランダムプログラム生成法

プログラムをランダムに生成してからその動作を把握し修正するのではなく，初めから動作が既知のプログラムをランダムに生成するという手法を考案した．例えば，x=10 という文があれば，これは a=7; b=3; x=a+b; に展開できる．このように，プログラムに対する等価変換を繰り返すことによって動作が既知の大きなランダムプログラムを生成する手法を開発し，算術式だけでなく，条件文やループ文を含むランダムプログラムを生成できるようにした（雑誌論文②，学会発表⑤）．また，算術式においても，式のオペランドの値を自由に制御できるため，不具合の検出能力を向上させることができた（学会発表③）．

Orange4 は本手法に基づき開発した C コンパイラ用ランダムテストシステムであり，GitHub で公開している．

(3) コンパイラの性能テスト

正しいコードを生成することは当然として，性能の良いコードを生成することがコンパイラにとっては極めて重要になっている．本研究では，コンパイラが意図通りの最適化を行っているかどうかのテストをランダムプログラムによって行う手法についても研究を行い，2 つの手法を開発した．1 つは，同一のテストプログラムを 2 つのコンパイラでコンパイルし，そのアセンブリコードを比較する方法である（雑誌論文④，学会発表⑦）．同一コンパイラの異なるバージョンでこのテストを行えば，リグレッションテスト（改良による機能劣化の有無のテスト）が行える．もう一つの手法が，等価な 2 つのプログラムを生成してコンパイラが生成するコードを比較する方法である(雑誌論文⑤，学会発表⑧)．いずれの場合もアセンブリコードの比較方法が課題になるが，これに関しても改良を重ねた（学会発表①）．

(4) LLVM IR および Java プログラムのランダム生成法

解析木よりプログラムを生成する方法では，プログラム生成部を改良することにより複数の言語に対応可能である．本研究では，LLVM IR および Java のランダムテストを行える処理系を開発した．LLVM IR のテスト生成では，バックエンドのコード生成において，C プログラムではテストできないパターンのテストを可能にした（雑誌論文③，学会発表④）．Java 処理系のランダムテストでは，Android 用コンパイラ dx のレジスタ割り当ての不具合を検出することができた(学会発表②)．

(5) C コンパイラの算術最適化のテストスイート

コンパイラのランダムテストは強力なツールではあるが，コンパイラの完成度が高くない場合には，同じ不具合が何度も検出されるため，その最小化や分析が効率的に行えないという課題がある．本研究では Orange3 が検出するほとんどのエラープログラムは最小化後には式数 1 で演算子数 3 程度の小さなプログラムになることに着目し，そのようなテストを集めて集中的にテストできるテストスイート CF3 を開発した（雑誌論文①，学会発表⑥⑨）．

本テストスイート CF は GitHub で公開している．

(6) その他

記号実行に基づくテストシステムの処理系を実装したが，その不具合検出能力は通常のランダムテストにくらべて著しく劣るものであった．これは，コンパイラのほとんどの最適化がオペランドの特定の値に依存して行われていることに起因すると考えられる．

５．主な発表論文等

〔雑誌論文〕（計 9 件）

① Y. Hibino, H. Ikeo, and N. Ishiura: CF3: Test Suite for Arithmetic Optimization of C Compilers (letter), IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 査読有り, vol. E100-A, no. 7, pp. 1-2 (July 2017).

② K. Nakamura and N. Ishiura: Random Testing of C Compilers Based on Test Program Generation by Equivalence Transformation, in Proc. Asia and Pacific Conference on Circuits and Systems (APCCAS 2016), 査読有り, pp. 676-679 (Oct. 2016).

③ K. Tanaka, N. Ishiura, M. Nishimura, and A. Fukui: Random Testing Back-end of Compiler Infrastructure LLVM (short paper), in Proc. the Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2016), 査読有り, R2-1, pp. 88-89 (Oct. 2016).

④ M. Iwatsuji, A. Hashimoto, and N. Ishiura: Detecting Missed Arithmetic Optimization in C Compilers by Differential Random Testing (short paper), in Proc. the Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2016), 査読あり, R1-1, pp. 2-3 (Oct. 2016).

⑤ Hashimoto and N. Ishiura: Detecting Arithmetic Optimization Opportunities for C Compilers by Randomly Generated Equivalent Programs, IPSJ Transactions on System LSI Design Methodology, 査読有り, vol. 9, pp. 21-29 (Feb. 2016).

⑥ 石浦菜岐佐: コンパイラのファジング (解説論文), 電子情報通信学会 Fundamentals Review, 招待論文, vol. 9, no. 3, pp. 188-196 (2016 年 1 月).

⑦ K. Nakamura and N. Ishiura: Introducing Loop Statements in Random Testing of C Compilers Based on Expected Value Calculation (short paper), in Proc. the Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2015), 査読あり, R3-3, pp. 226-227 (Mar. 2015).

⑧ E. Nagai, A. Hashimoto, and N. Ishiura: Reinforcing Random Testing of Arithmetic Optimization of C Compilers by Scaling up Size and Number of Expressions, IPSJ Transactions on System LSI Design Methodology, 査読有り, vol. 7, pp. 91-100 (Aug. 2014).

⑨ E. Nagai, A. Hashimoto, and N. Ishiura: Scaling up Size and Number of Expressions in Random Testing of Arithmetic Optimization of C Compilers, in Proc. the Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2013), 査読あり, R2-3, pp. 88-93 (Oct. 2013).

〔学会発表〕（計 11 件）

① 北浦幸太，岩辻光功，石浦菜岐佐: C コンパイラの最適化のリグレッションテストのためのアセンブリコード比較法，電子情報通信学会ソサイエティ大会，北海道大学 (北海道・札幌市)，2016 年 9 月 21 日．

② 清水遼太朗，池尾弘史，石浦菜岐佐: コンパイラのランダムテストシステム 0range3 の拡張による Java 処理系のテスト，電子情報通信学会ソサイエティ大会，北海道大学 (北海道・札幌市)，2016 年 9 月 21 日．

③ 高倉正悟，石浦菜岐佐: 等価変換に基づく C コンパイラのランダムテストにおける変数の複数回参照の導入,電子情報通信学会ソサイエティ大会，北海道大学（北海道・札幌市)，2016 年 9 月 21 日．

④ 田中健司，石浦菜岐佐，西村啓成，福井昭也: コンパイラ基盤 LLVM バックエンドのランダムテスト，電子情報通信学会総合大会，九州大学（福岡県・福岡市)，2016 年 3 月 17 日．

⑤ 中村和博，石浦菜岐佐: 等価変換に基づくテストプログラム生成を用いた C コンパイラのランダムテスト，電子情報通信学会 VLSI 設計技術研究会，青年会館（沖縄県・那覇市)，2016 年 2 月 29 日．

⑥ 池尾弘史，石浦菜岐佐: C コンパイラ用テストスイート CF3 の検出能力向上，電子情報通信学会ソサイエティ大会，東北大学 (宮城県・仙台市)，2015 年 9 月 10 日．

⑦ 岩辻光功，橋本淳史，石浦菜岐佐: 差分ランダムテストに基づくコンパイラの最適化機会の検出,電子情報通信学会ソサイエティ大会，東北大学（宮城県・仙台市)，2015 年 9 月 10 日．

⑧ 橋本淳史，石浦菜岐佐: ランダムテストによる C コンパイラの算術最適化機会の検出，電子情報通信学会 VLSI 設計技術研究会，慶應義塾大学（神奈川県・横浜市)，2015 年 1 月 30 日．

⑨ 日比野佑亮，石浦菜岐佐: C コンパイラの算術最適化を対象としたテストスイート CF3，電子情報通信学会 VLSI 設計技術研究会，慶應義塾大学（神奈川県・横浜市)，2015 年 1 月 29 日．

⑩ 中村和博，石浦菜岐佐： C コンパイラの関数呼び出し規約のランダムテストにおけるエラープログラムの最小化，電子情報通信学会ソサイエティ大会，徳島大学（徳島県・徳島市），2014 年 9 月 24 日．
⑪ 橋本淳史，石浦菜岐佐： C コンパイラの算術最適化のランダムテストにおける浮動小数点演算の導入，電子情報通信学会ソサイエティ大会，徳島大学（徳島県・徳島市），2014 年 9 月 24 日．

〔その他〕
・ランダムテストシステム 0range3
http://ist.ksc.kwansei.ac.jp/~ishiura/pub/orange3/
・ランダムテストシステム 0range4
http://ist.ksc.kwansei.ac.jp/~ishiura/pub/orange4/
・C コンパイラ用テストスイート CF3
http://ist.ksc.kwansei.ac.jp/~ishiura/pub/CF3/

６．研究組織
(1) 研究代表者
石浦 菜岐佐（ISHIURA NAGISA）
関西学院大学・理工学部・教授
研究者番号：60193265