コンパイラのランダムテストシステムOrange3の拡張によるJava処理系のテスト

Testing Java Processing Environments by Extending Compiler Random Test System Orange3

清水 遼太朗　Ryotaro Shimizu
池尾 弘史　Hirofumi Ikeo
石浦 菜岐佐　Nagisa Ishiura

関西学院大学 理工学部　School of Science and Technology, Kwansei Gakuin University

1　はじめに

Java は基幹系のシステムだけでなく, 近年では Android アプリの開発にも利用されており, そのコンパイラおよび実行系の信頼性確保は非常に重要な課題である. Java の処理系のテストとしては, ランダムに生成したバイトコードによる JIT コンパイラのランダムテスト [1] があるが, コンパイラや, 複雑な式を計算するバイトコードによる実行系のテストは行っていない. そこで本稿では, C コンパイラ用ランダムテストシステム Orange3 [2] を拡張することにより, Java 処理系のテストを行う手法を提案する.

2　ランダムテスト Orange3

Orange3 は, C コンパイラの算術最適化をターゲットとするランダムテストシステムである. Orange3 が生成する C プログラムは, 変数の宣言と初期化, 算術式を右辺に持つ代入文, および計算結果の照合を行う文により構成される. 実行結果の正誤によりコンパイラの生成するコードの正当性が確認できる.

3　Java 処理系のランダムテスト

本稿では, Orange3 の拡張によって Java 処理系のテストを行う手法を提案する. 図 1 に示すように, Orange3 では一旦抽象構文木 (ASTs) を構築してそこから C プログラムを生成しているので, 抽象構文木から Java のプログラムを生成するとともに, 抽象構文木生成時の型に関するルールを変更する.

Java では符号無し整数型は扱わないので, 符号付きの byte, short, int および long 型のみを扱うようにする. また, Java では比較演算の結果や論理演算には boolean 型を用いるため, 例えば表 1 のように, 式中で適切な型変換を行うようにする. 変数の修飾子には final, private, protected, public, または transient を指定する. 本手法が生成するテストプログラムの例を図 2 に示す. 9 行目以降で変数の宣言と初期化を, 598 行目以降で演算を行い, 711 行目以降で結果の照合を行っている.

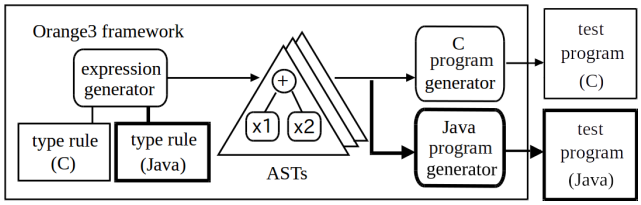

図 1　テストプログラム生成の流れ

表 1　Java における比較演算子と論理演算子の型変換

C	Java
x1 > x2	(x1 > x2) ? 1 : 0
x1 && x2	((x1 != 0) && (x2 != 0)) ? 1 : 0

```
  1: class java{
  2:     public static void ok()  {
  3:         System.out.print("@OK@\n");
  4:     }
  5:     public static void ng(Number num)  {
  6:         System.out.print("@NG@ (test = "+num+")\n");
  7:     }
  8:
  9:     static byte x0 = 10;
 10:     static int x3 = 519355;
                 ...
373:     public static void main(String[] args){
374:
375:         short x2 = -1387;
376:         byte x4 = -1;
                 ...
598:         t0 =(x349+((x207+k426)/(x52*x29)));
599:         t1 =(((x3<=x213)?1:0)>>(x57%x245));
                 ...
711:         if (t0 == -164L) { ok(); } else { ng(t2); }
712:         if (t1 == 0) { ok(); } else { ng(t4); }
                 ...
825:     }
826: }
```

図 2　本手法で生成するテストプログラムの例

4　実験結果

本手法に基づくランダムテストシステムを実装し, OpenJDK 8 と dx (1.1.2) および ART (2.1.0) のテストを行った. 1 プログラム中の演算数は 1000 とした. 結果を表 2 に示す. #test は生成したテストプログラムの数であり, #error はエラーを検出したものの数である. OpenJDK と dx の最適化なし (--no-optimize オプション) ではエラーを検出しなかったが, dx で最適化を行う場合には多数のエラーを検出した. 現在, このエラーの原因を分析中である.

表 2　実験結果

compiler	time[h]	#test	#error
OpenJDK 8	775	720,090	0
dx 1.1.2 + ART 2.1.0 (最適化なし)	60	47,235	0
dx 1.1.2 + ART 2.1.0 (最適化あり)	66	50,729	9,768

(Ubuntu 14.04 LTS, Xeon 3.60GHz, RAM 16GB)

5　むすび

本稿では, Orange3 の拡張による Java 処理系のランダムテストを提案した. 検出したエラーの原因の分析, および他の処理系のテストが今後の課題である.

謝辞 本研究は一部 JSPS 科研費 25330073 の助成による.

参考文献

[1] T. Yoshikawa, K. Shimura, and T. Ozawa: "Random program generator for Java JIT compiler test system," *Proc. Quality Software*, pp. 20–23 (Nov. 2003).

[2] 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 Trans. SLDM*, vol. 7, pp. 91–100 (Aug. 2014).