等価変換に基づくランダムテストプログラム生成によるJava処理系のテスト

Testing Java Processing Environments by Random Program Generation Based on Equivalent Transformation

吉田 直生　　石浦 菜岐佐
Naoki Yoshida　　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や, Android アプリケーションの開発に利用されており, 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である. Java 処理系のテストとしては, C コンパイラ用ランダムテストシステム Orange3 [1] を Java に対応させて行ったテスト [2] があり, Android 処理系で不具合を検出している. Orange4 [3] は Orange3 を改良して作られた C コンパイラ用ランダムテストシステムであり, Orange3 では生成できない構文, データの型を生成可能である. 本稿では, Orange4 の拡張により Java 処理系のテストを行う.

2　Orange3 と Orange4

Orange3, Orange4 はともに C コンパイラををターゲットとするランダムテストシステムであり, 抽象構文木を構築してそこからプログラムを生成する. Orange3 を含めてランダムな算術式生成を行う手法では実行時の式の値が制御できないのに対し, Orange4 では最初に式の値を決めて, そこから等価変換 (例えば 6 を $9+(-3)$ に変換する等) によって複雑でランダムな式を生成する. これによって, 境界を超えない配列参照や無限ループに陥らないループ文を可能にしており, Orange3 や従来の方法では生成できない構文のプログラムを生成できる.

3　Java 処理系への対応

本研究では, Orange4 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. 変数の型, 修飾子に関するルールを変更し, 抽象構文木から Java のテストプログラムを生成する. 加えて, 新たにクラスの生成を行う.

Java では符号無し整数型は扱わないため, 抽象構文木生成の等価変換の際に符号付きの byte, short, int, long 型のみを選択するよう変更する. 変数の修飾子では final, private, protected, public, transient を指定する.

クラス生成は, 構造体の生成部分を変更し static 内部クラスの複数生成を可能にする. 内部クラスのメンバ変数は普通の整数型 (配列, クラス以外) のみである. またメソッドの生成は行わない.

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, Open-JDK 8 と dx-1.16, d8-1.5.13 および ART-2.1.0 のテストを行った. 生成されたテストプログラムの例を図 1 に示す. 5 行目以降で変数の宣言と初期化を, 599 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. 変数の宣言部分でクラス, 配列の生成を行うことができており, 演算部分で if 文, for 文, switch 文の生成を行うことができている.

```
1: class java{
       ...
5:     static class x0{
6:       static short m0 = 10;
7:       static int m1 = 43;
8:     }
       ...
373:   public static void main(String [] args) {
374:      short[]t16 = {5674,834,4,58,394};
       ...
599:      switch( (x99[x94]/((x98)+x96)) ){
600:         default:
601:           t0[x24[2]][x26] = (x21*x29.m3)+(x12-x6);
602:         break:
603:      }
604:      if((x39[4]+x64)+(x65*x67) != 0 ? true:false){
605:         t12.m2 = ((x653*t25)+(x45*x654));
606:      }
607:      for(i=x56*x57;i>x7[t8[1][0]][t9];i+=x58*x59){
608:         t55 = ((x572-t41)%t8[t8[1][0]][x573]);
609:      }
       ...
711:      if (t19 == -164) { ok(); } else { ng(); }
       ...
825:   }
826: }
```

図 1　本手法で生成するテストプログラムの例

表 1　テスト実行結果

compiler	time [h]	#test	#error
OpenJDK 8	24	23,834	0
dx 1.16 + ART 2.1.0	24	10,412	0
d8 1.5.13 + ART 2.1.0	24	5,700	0

結果を表 1 に示す. #test は生成したテストプログラム数であり, #error は検出したエラー数である. 24 時間テストを行ったが新たなエラーは検出しなかった.

5　むすび

本研究ではコンパイラのランダムテストシステム Orange4 の拡張による Java 処理系のテストを提案した. テストプログラムの強化と Android 用 AOT コンパイラの最適化性能のテストが今後の課題である.

参考文献

[1] 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 TSLDM*, vol. 7, pp. 91–100 (Aug. 2014).

[2] 清水, 池尾, 石浦: "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," 信学ソ大, A-6-11 (Sept. 2016).

[3] S. Takakura, M. Iwatsuji, and N. Ishiura: "Extending equivalence transformation based program generator for random testing of C compilers," in *Proc. A-TEST 2018*, pp. 9–15 (Nov. 2018).