Cコンパイラの算術式を対象としたランダムテストにおけるエラープログラムの最小化

Minimization of Error Programs for Random Testing of C Compilers Targeting Arithmetic Expressions

永井絵里子 石浦菜岐佐
Eriko Nagai 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

コンパイラのランダムテストは, テストスイートによるテストを経てもなお潜在する不具合の検出を目的とする. 粟津ら [1] は算術式を対象としたCコンパイラのランダムテスト手法を提案し, 複数のコンパイラで不具合を検出した. しかし, エラーの原因を絞り込むための最小化は手動で行っており, 時間を要することが課題となっていた. そこで本研究では, このランダムテストにおけるエラープログラムの自動最小化手法を提案する.

2 算術式を対象としたランダムテスト

[1] の手法では, ランダムに生成した算術式を含むプログラムをコンパイルし, 実行結果を期待値と照合する, という処理を時間の許す限り繰り返すことによりコンパイラの不具合の検出を試みる. この手法により, 例えばi686-cygwin-gcc4.4.1 に対して図 1 のようなエラープログラムが得られる. しかし, コンパイラの不具合の解析を行うためには, このプログラムをより簡潔でエラーが発生するもの) に変換する必要がある.

```
#include <stdio.h>
const volatile unsigned char x1 = 2U;
const volatile signed long long x6 = 1476669LL;
static const unsigned short x8 = 35U;
int main (void)
{
   int rc = 0;
   signed long long test = 0;
   test = (((x8*(x6<<x8))>=x1)/x6);
   if(test == 0LL) {
      printf("OK, %lld\n",test);
   }
   else {
      rc = 1;
      printf("NO, %lld\n",test);
   }
   return rc;
}
```

図 1 エラープログラム

3 エラープログラムの算術式の最小化

エラープログラムの最小化は, 1) トップの演算子の一方のオペランドの選択, 2) 変数への値の代入, 3) 演算の評価, という 3 つの基本操作の繰り返しにより行える. 本稿では, それ以上どの操作を適用してもエラーが発生しなくなるものを最小化されたプログラムと定義する.

本手法では, 「トップダウン最小化」と「ボトムアップ最小化」を効果がなくなるまで交互に繰り返すことにより, 算術式の最小化を行う.

トップダウン最小化は基本操作 1) を行うものである. 解析木を受け取ると, 根の各子を新たな根とする部分木からプログラムを生成する. いずれかのプログラムでエラーが検出される限り, 再帰的にトップダウン最小化を適用する.

ボトムアップ最小化は, 基本操作 2), 3) を行うものである. 解析木中変数を表す節点, または両方の子が定数となっている演算節点を 1 つ選び, それぞれ変数の初期値, または演算結果の値を持つ定数節点に置換する. この解析木から生成したプログラムでエラーが検出されれば同様の操作を繰り返す. そうでなければ解析木を元に戻し, 他の節点を選んで置換の操作を試みる, という操作を候補節点がなくなるまで繰り返す.

4 実行結果

本手法を Perl 5.10.1 で実装し, [1] のランダムテストに組み込んだ. 図 1 のプログラムは図 2 のように最小化される. 実行に要した時間 (i686-cygwin-gcc4.4.1, Core i5, CPU 3GHz) を表 1 に示す. 十数個の演算子を含む算術式を 20 秒以内に最小化できている.

表 1 最小化に要した時間

最小化前の演算子数	最小化後の演算子数	時間 [sec]
5	1	8.75
6	3	8.25
8	2	6.76
8	2	5.49
11	1	13.48
14	2	17.25

```
#include <stdio.h>
const volatile signed long long x6 = 1476669LL;
int main (void)
{
   int rc = 0;
   signed long long test = 0;
   test = (x6<<(signed int)35);
   if(test == 50737960496136192LL) {
      printf("OK, %lld\n",test);
   }
   else {
      rc = 1;
      printf("NO, %lld\n",test);
   }
   return rc;
}
```

図 2 最小化されたエラープログラム

5 むすび

本研究では, 算術式を対象としたCコンパイラのランダムテストにおけるエラープログラムの最小化を提案した. 算術式中の演算子数が増えた場合には最小化の高速化が課題になると考えられる.

参考文献

[1] 粟津, 石浦: “算術式の最適化を対象としたCコンパイラのランダムテスト,” 信学技報, VLD2008-127 (Mar. 2009).