C コンパイラのランダムテストにおける 非実行パス上の被代入変数の参照の導入

Introducing References of Variables Assigned on Unexecuted Paths in C Compiler Random Testing

前田 紘輝 Hiroki Maeda
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

コンパイラのランダムテストは, ランダムに生成したプログラムによりコンパイラに潜在する不具合の検出を試みる手法である. 文献 [1] は, プログラムの等価変換によって複雑な算術式, 条件文, ループ文を含むテストプログラムを生成する手法を提案している. この手法では, 実行されないパス上の文が後続の文に影響を及ぼさないようなプログラムを生成するが, 文献 [2] では, 非実行パスに挿入した文がコンパイラの潜在的な不具合を検出する例が報告されている. そこで本稿では, 文献 [1] の手法に非実行パス上の被代入変数の参照を導入する手法を提案する.

2 ランダムテストシステム Orange4

Orange4 [1] は, C コンパイラのランダムテストシステムであり, 図 1 のように, 算術式と代入文, if 文, for 文から成るプログラムを生成することによりテストを行う. この手法では, if 文や for 文の条件を生成時に決定できるため, 代入文の右辺には宣言時か実行パス上で値が定義される変数のみ出現させている. 例えば図 1 では, 15 行目の if 文の条件は真であるため, 非実行パス上の被代入変数である t1 が以降の文で参照されることはない.

```
01: #include <stdio.h>
02: #define OK()  printf("@OK@\n")
03: #define NG(test,fmt,val) printf("@NG@(test="fmt")\n",val)
04:
05: unsigned char t0 = 98U;
06: signed char t1 = 0;
07: unsigned char t2 = 38U;
08: unsigned int t3 = 66;
09: const volatile unsigned short x152 = 375U;
10: int main (void)
11: {
12:   static unsigned long long x2 = 305LLU;
13:   static const volatile signed int x153 = 34645;
14:   signed int i;
15:   if(x153/x152) { t0 = ((unsigned char)(x152%x2)); }
16:   else { t1 = ((signed char)(t0==t0)); }
17:   t2 = ((unsigned char)(x153%((signed short)t0)));
18:   for( i = (t0|x152); i > (x153*t2); i += (x152|t2)) {
19:     t3 = ((unsigned int)(t2*(signed int)t0));
20:   }
21:   if (t0 == 70U) { OK(); } else { NG("t0", "%u", t0); }
22:   if (t2 == 65U) { OK(); } else { NG("t2", "%u", t2); }
23:   return 0;
24: }
```

図 1 Orange4 が生成するテストプログラム

3 非実行パス上の被代入変数の参照の導入

本稿では, 不具合検出能力の向上を目的に, 文献 [1] の手法に非実行パス上の被代入変数の参照を導入する. 本手法により生成されるプログラムを図 2 に示す. 図 2 において, 16 行目の else 文は非実行パスである. 17 行目の代入文では, 16 行目の非実行パス上代入されている t0 を参照している.

非実行パスで上の被代入変数の参照は, 実行パス上の被代入変数と同一の型, 名前の変数を新たに生成することにより実現する. 図 2 において, 16 行目の非実行パス上の代入文を生成する際には, 実行パス上の被代入変数から t0 を選択し, 同じ名前, 同じ型の変数 t0 を生成し, これを被代入変数として利用する. 17 行目の代入文では, 15 行目の実行パス上で値が定義される t0 を参照しているが, 16 行目の非実行パス上の被代入変数 t0 を参照していることにもなる.

```
15:   if(x153/x152) { t0 = ((unsigned char)(x152%x2)); }
16:   else { t0 = ((signed char)(t0==t0)); }
17:   t2 = ((unsigned char)(x153%((signed short)t0)));
```

図 2 非実行パス上の被代入変数参照

4 実験結果

提案手法を Orange4 に実装した. GCC の 3 つのバージョンに対してテストを行った結果を表 1 に示す. “#test” はテスト総数, “#error” は検出したエラー数を示す. エラー検出数に差は見られないが, 従来法では検出できないエラーが検出ができた. 本手法により検出した GCC-4.7 の不具合を検出したプログラムを最小化したものを図 3 に示す. 5 行目の b = 0; を削除するとエラーは消失する.

表 1 実験結果

compiler	Orange4 (**従来**)		Orange4 (**提案**)	
	#test	#error	#test	#error
GCC-4.7	120,000	67	120,000	66
GCC-4.8	120,000	70	120,000	73
GCC-4.9	120,000	69	80,000	75

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

```
1: int main (void) {
2:   int a = 1;
3:   int b = -1;
4:   volatile int c = 0;
5:   if(c) { b = 0; }
6:   int t = b - (-0x7FFFFFFF - a);
7:   if (t != 0x7FFFFFFF) __builtin_abort();
8:   return 0;
9: }
```

図 3 GCC-4.7 の不具合を検出したプログラム

5 むすび

プログラムの等価変換に基づくランダムテストへの非実行パス上の被代入変数参照の導入を提案した. エラープログラムの自動最小化の実装が今後の課題である.

参考文献

[1] K. Nakamura and N. Ishiura: “Random testing of C compilers based on test program generation by equivalence transformation,” in *Proc. APCCAS 2016*, pp. 676–679 (Oct. 2016).

[2] V. Le, C. Sun, and Z. Su: “Finding deep compiler bugs via guided stochastic program mutation,” in *Proc. OOPSLA 2015*, pp. 386–399 (Oct. 2015).