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

Introducing Multiple Variable References in C Compiler Random Testing Based on Equivalence Transformation on Test Programs

高倉 正悟 Shogo Takakura
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. しかし, この手法ではプログラム内において各変数を一回しか参照していなかった. 本稿では, 等価変換に基づくランダムプログラム生成において変数の複数回参照をする手法を提案する.

2 ランダムテストシステム Oraneg4

Orange4 [1] は, C コンパイラの算術最適化を対象とするランダムテストシステムであり, 図 1 のようなプログラムを生成することによりテストを行う. プログラム中の算術式 (15–16 行目) は, 算術式の等価変換により生成する. この等価変換は, 例えば `t0=x0;` (`x0==36`) を `t0=x1<<x2;` (`x1==9, x2==2`) に展開するという操作を繰り返すものである. この際, 一方のオペランドの値は許容される最小値と最大値の間から選択するが, 境界値や特定の値を高い確率で選択することにより, 不具合検出の向上を図ることができる.

しかし [1] では, この展開の都度新規の変数を生成している. 式中で既定義変数の参照は行わないため, 生成する式の形が限定されていた.

```
01: #include <stdio.h>
02: #define OK()  printf("@OK@\n")
03: #define NG(fmt,val)  printf("@NG@ (test = "fmt")\n",val)
04:
05: int main (void)
06: {
07:   unsigned int x10 = 2U;
08:   static volatile signed int x11 = 3;
09:   volatile signed char t0 = 15;
10:   signed short t1 = 190;
11:   volatile unsigned long long x9 = 2LLU;
12:   static volatile unsigned long long x15 = 1LLU;
13:   static const volatile unsigned long long x16 = 3LLU;
14:   static volatile signed char x18 = 1;
15:   t0 = ((signed char)(((signed int)(x9*x10))*(x11)));
16:   t1 = ((signed short)(((signed char)(x15*x16))<<x18));
17:   if (t0 == 12) { OK(); } else { NG("t0", "%d", t0); }
18:   if (t1 == 6) { OK(); } else { NG("t1", "%hhd", t1); }
19:   return 0;
20: }
```

図 1 Orange4 が生成するテストプログラム

3 変数の複数回参照の導入

本稿では, 等価変換に基づくランダムプログラムの生成において, 変数の複数回参照を行う手法を提案する. これによって図 3 のように, 同じ変数 `x0` が代入文の右辺に複数回出現したり, 式の計算結果 `t0` が別の式で参照されるようになる.

既存変数の参照は, 値でソートした変数のリストから所望の値の変数を検索することにより実現する. まず, オペランドの生成時に, 新規変数を生成するか既存変数を参照するかをランダムに決定する. 既存変数を参照する場合, 値が指定されていれば, その値を持つ変数をリストから二分探索で検索する. 値が範囲で指定されている場合には, 最小値以上の最も小さい値を持つ変数と, 最大値以下のもっとも大きい変数を検索し, その間にある変数をランダムに選択する. いずれの場合も, 所望の変数が存在しなければ, 新規に変数を生成する.

```
t0 = x0 * x1 + x0;
t1 = x2 * x3 * t0;
```

図 2 変数の複数回参照

4 実装結果

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 24 時間行った結果を表 1 に示す. “#test” はテスト総数, “#error” は検出したエラー数を示す. 提案手法は, 従来法に比べて多くのエラーを検出している. 変数の複数回参照は Orange3 [2] でも行っているが, オペランドの値の分布を制御する Orange4 の方がエラー検出能力が高い. 本手法により検出した GCC-7.0.0 (開発版) の不具合 2 件を開発者に報告した [1].

表 1 実験結果

compiler	Orange3		Orange4 (**従来**)		Orange4 (**提案**)	
	#test	#error	#test	#error	#test	#error
GCC-4.7	56,813	0	27,382	18	32,575	21
GCC-4.8	64,622	1	34,342	9	39,348	19
GCC-4.9	66,752	1	26,550	10	39,824	26

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

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変数の複数回参照の導入を提案した. 配列, 構造体, 関数呼び出し等を含むテストの生成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] 中村, 石浦: “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,” 信学技報, VLD2015-112 (Feb. 2016).

[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).

[1] https://gcc.gnu.org/bugzilla/ id=71608, id=71631