Testing static analyzers with randomly generated programs

Pascal Cuoq* Benjamin Monate Anne Pacalet
Virgile Prevosto John Regehr Boris Yakobowski
Xuejun Yang

CEA, LIST

INRIA Sophia-Antipolis

University of Utah

Plan: 1- Csmith 2- Frama-C 3- Using Csmith with Frama-C

1- is previous work from CEA, LIST and INRIA Sophia-Antipolis
2- is previous work from University of Utah
3- is new and is joint work from all three institutes

Csmith: a random generator of C programs

Csmith: a random generator of C programs

Representative "line":

```
(*l_1209) ^= func_2(func_7(g_13, g_13.f0, g_13.f0,
func_14(func_20(l_25, ((*l_29)--), func_32(
func_34((((((*l_963) = func_37(
(safe_unary_minus_func_int32_t_s((
safe_rshift_func_uint16_t_u_s((
safe_add_func_uint8_t_u_u(func_47(l_25),
(safe_mul_func_uint16_t_u_u(((
safe_add_func_uint64_t_u_u(((*l_102) = 0xC3DD168C87B4EC18LL
(safe_mul_func_int16_t_s_s((((g_117 = (safe_sub_func_int16_
safe_lshift_func_int16_t_s_u((
g_116 = (g_115 = ((*l_112) ^= (((g_76.f1.f1 ,
g_110[2]) != &g_111) , 0xF5F5L)))), l_25))
|| g_76.f1.f1), g_76.f1.f2))) < l_118)
>= (-8L)), g_65)))) < l_25), g_13.f0)))), g_76.f1.f3)))),
g_119, l_26[1], &g_65)) == g_964) == 0x24AE7014CC3713C7LL) |
l_25), g_966)), g_415), l_988, g_967, l_118, l_25), l_25), l_
```

Csmith: a random generator of C programs

What is a random program good for?

Csmith: a random generator of C programs

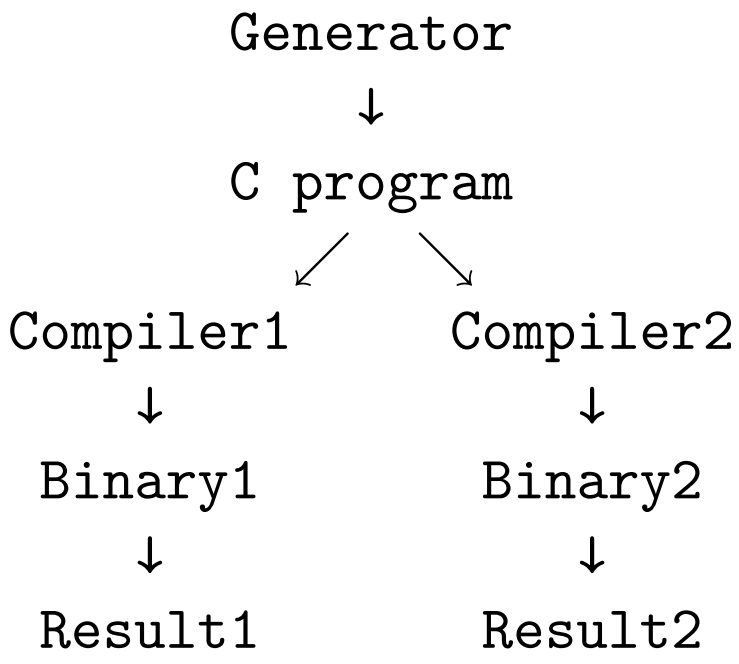
What is a random program good for?

Differential testing of compilers

Csmith: a random generator of C programs

What is a random program good for?

Differential testing of compilers



Frama-C: a static analysis platform for C

Initial focus on verification of critical embedded code

Plug-in architecture

Frama-C: a static analysis platform for C

Initial focus on verification of critical embedded code

Plug-in architecture

Used industrially
(in some cases with plans to claim DO-178B certification credit)

Some plug-ins are automatic

Differential testing static analyzers

Differential testing static analyzers

It is not straightforward.

Oracle 1

1. Transform abstract interpreter into plain interpreter
2. Apply differential testing with reference compiler

```
          Generator
              ↓
          C program
            ↙   ↘
Compiler1
    ↓           Value analysis
 Binary1            ↓
    ↓
 Result1        Result2
```

Oracle 2

- Generate abstract state as executable assertion

```
Generator
    ↓
C program
  ↙   ↘
      Value analysis
 ↓          ↓
      Executable invariant
  ↘   ↙
Instrumented program
    ↓
Compiler
    ↓
 Binary
    ↓
Diagnostic
```

```
[value] Values at end of function main:
...
          g_125 ∈ {61; 72; 73}
          g_228.f0 ∈ {-1; 7}
...
```

```
[value] Values at end of function main:
...
          g_125 ∈ {61; 72; 73}
          g_228.f0 ∈ {-1; 7}
...
```

```
(*(unsigned int*)&i == 1
...
&& 61 <= *(unsigned char*)&g_125
&& *(unsigned char*)&g_125 <= 73
&& -1 <= *(char*)&g_228 && *(char*)&g_228 <= 7
...
```

```
[value] Values at end of function main:
...
          g_125 ∈ {61; 72; 73}
          g_228.f0 ∈ {-1; 7}
...
```

```
(*(unsigned int*)&i == 1
...
&& 61 <= *(unsigned char*)&g_125
&& *(unsigned char*)&g_125 <= 73
&& -1 <= *(char*)&g_228 && *(char*)&g_228 <= 7
...
```

```
...
int seems_ok = <generated expression> ;
printf("seems ok:%d\n", seems_ok);
```

Conclusion

- Started as casual experiment to “find some bugs”
- Produced some by-products (an interpreter for C is useful)
- Produced methods that will be used for tool qualification
- Found about 50 bugs in Frama-C
- Found bugs in Csmith
- Article and annex with details at http://j.mp/csmithtesting