Finding Compiler Bugs with Random Testing

Xuejun Yang, Yang Chen, Eric Eide, John Regehr
School of Computing, University of Utah
{jxyang, chenyang, eeide, regehr}@cs.utah.edu

A GCC Bug

```
int g[1];
int *p = &g[0];
int *q = &g[0];

int main(void)
{
  g[0] = 1;
  *p = 0;
  *p = *q;
  printf("%d\n", g[0]);
}
```

End result of g[0]:
Expected: 0
GCC: 1

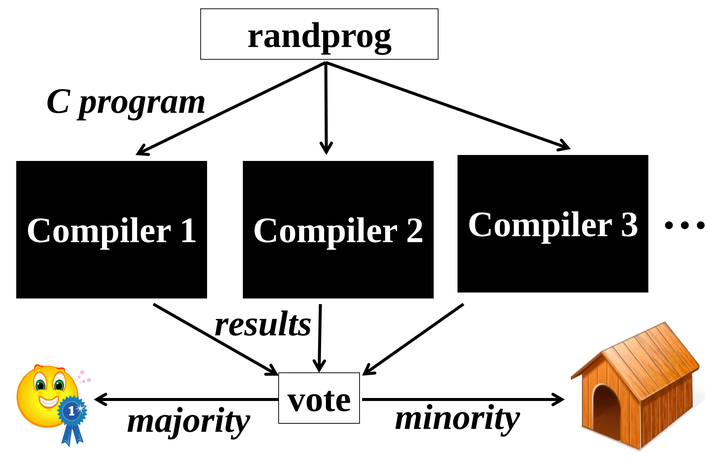
A LLVM Bug

```
short g, i ;
void f()
{
   short l[5];
   for (i=0; i<1; i++)
      l[i] = 0;
lbl:
   l[0] ^= 1;
   assert(&g != 0);
   goto lbl;
}
```

Compiler Crashed

randprog - the tool

- A random C program generator
- Detects incorrectness of compilers by voting
- We used it to find numerous compilers bugs



Good Vs Bad Programs

A program that can be compiled ⇏ Good program

A program can be compiled And Has ZERO undefined or unspecified behavior = Good program

Undefined behavior (such as divide by 0) or unspecified behavior (such as order of evaluation) gives compilers freedom to diverge, causing failure to our voting mechanism

Bad program example

```
int f(void)
{
   int a[5];
   int i, j;
   int* p;

   a[0] = *p;
   j = 6;
   a[j] = 5;
   goto lbl;
   p = &j;
lbl:
   a[1] = *p;
}
```

dead pointer dereference

array index out of bound

Jump over variable initialization

The work behind the scene that avoids generating bad programs in randprog

Pointer Analysis

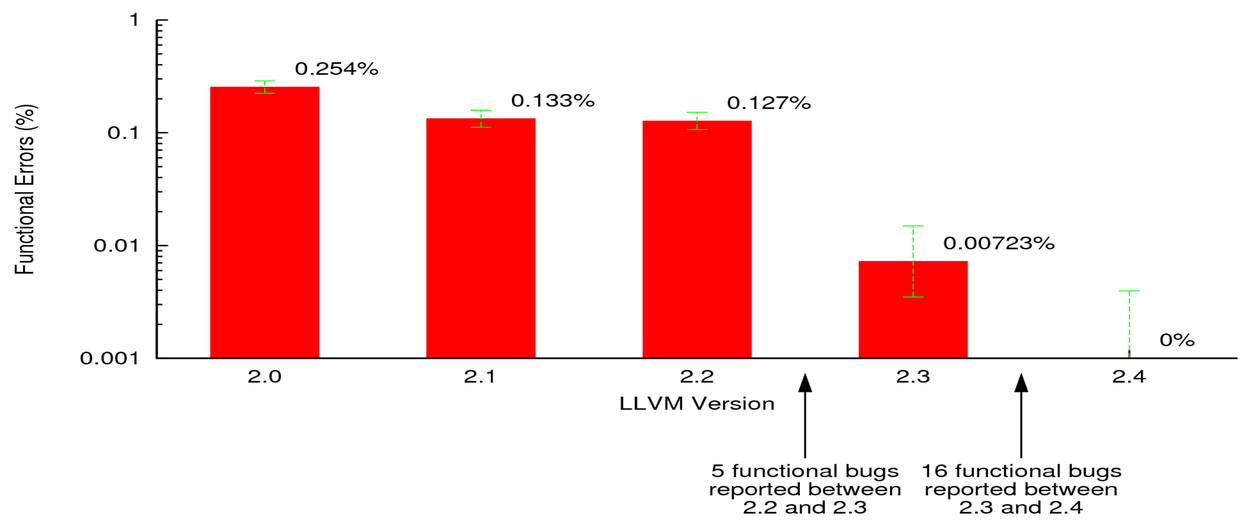
- a context-sensitive flow-sensitive inter-procedural point-to analyzer
- tracks the point-to relationship between variables. For example:
  x = &y => x -> {y}
- the analysis is performed on-the-go after each statement generation
- randprog consults the analyzer to avoid null/dead pointer dereferencing

Control Flow Analysis

- a control flow analyzer that can handle abnormal edges created by jump statements (goto / break / continue)
- Preemptively analyze possible effects a new jump statement would cause, and reject it if undefined behavior is introduced
- Backward jumps are treated as loop creator. Possible undefined behaviors are identified after a fixed point analysis

Out-of-bound array indexing is avoided by taking modulo

How we helped a compiler to be more reliable



Functional Errors we found in LLVM