Test Case Reduction, Fuzzing, Code Coverage

Juneyoung Lee

May 28

Test Case Reduction

- Common(?) scenario:
  - You add an optimization
  - One of the benchmark set fails
  - You investigate the reason..... but it is often a daunting job
  - Give up?
- Can we automatically reduce the input program into a smaller, shorter one?

Creduce

- An automatic C program reduction tool for compilers
- Made for C programs, but works for other languages too (Javascript, Rust, ...)
- Given an original program and a test script,
  - Repetitively replace a portion of code into a simpler one (guided by heuristics)
  - See whether it still makes miscompilation happen

How to use Creduce?

- Ubuntu: `apt install creduce`
- Mac: `brew install creduce`
- How to run: `creduce <test.sh> main.c`
  - The script compiles main.c with two different compilers and sees diff
  - If outputs are different, it exits with zero
  - practice/10.materials/creduce has a sample
- Demo

What about unseen cases?

- Rather than reducing existing tests...
- Can we generate inputs & run tests?
- Fuzzing!

Fuzzing

- Generate random inputs
  - Which can be pretty exotic from the inputs that programmers had expected
  - Especially good at finding security holes from softwares
- Often combined with...
  - Sanitizers: to find possible security problems
  - Code Coverage: to generate more effective inputs
- Scalable and effective
  - To find more bugs, simply run more fuzzers!
  - Google has a cluster for fuzzing (https://google.github.io/clusterfuzz/)

Csmith

- A random C program generator for compilers
- The generated program does not have undefined behavior!
- Found numerous bugs from gcc/clang
- Combined with Creduce, it can generate high-quality bug report

How to use Csmith?

- Ubuntu: apt install csmith
- Mac: brew install csmith
- How to run: csmith (with some option)
  - practice/10.materials/csmith

Clang code Coverage

- https://clang.llvm.org/docs/SourceBasedCodeCoverage.html