FAILURE-DIRECTED PROGRAM TRIMMING

Kostas Ferles[1] Valentin Wüstholz[1]
Maria Christakis[2] Isil Dillig[1]

1. The University of Texas At Austin, US 2. Max Planck Institute SWS

MOTIVATION:

Safety-Checking tools are sensitive to #Program-Paths

```
int fact (int n) {
  assume(0 <= n);

  int r = 1;
  if (n != 0) {
    r = n * fact(n-1);
  }

  assert(n != 0 || r == 1);   // AI
  return r ;
}

void main () {
   int m = *;

   int f = fact(m);
   assert(m != 123 || f == 0); // DSE
}
```

TRIMMER

Implemented in SeaHorn, an LLVM-based analysis tool

SeaHorn

IDEA:

A fast program transformation that eliminates safe paths.

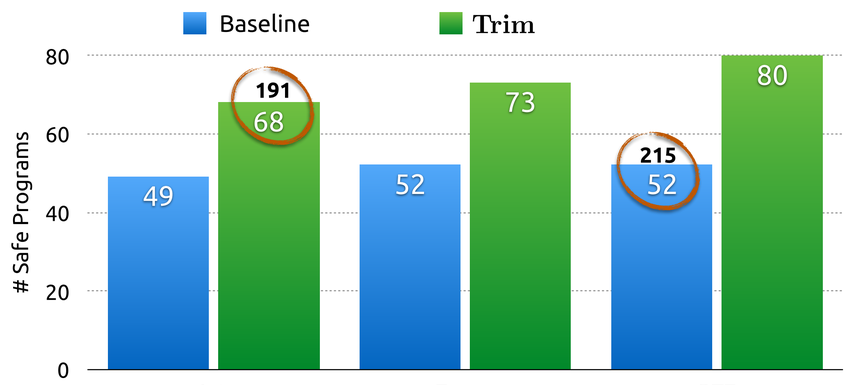
```
int fact (int n) {
  assume(0 <= n);

  assume(n != 0);

  int r = 1;
  if (n != 0) {
    r = n * fact(n-1);
  }

  assert(n != 0 || r == 1);   // AI
  return r ;
}

void main () {
   int m = *;

   assume(m == 123);

   int f = fact(m);
   assert(m != 123 || f == 0); // DSE
}
```

KEY INSIGHTS:

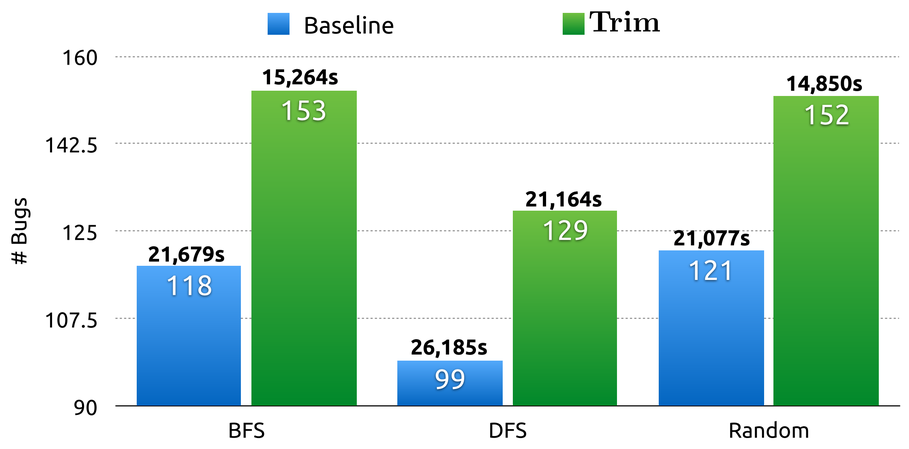
- The transformation produces an "equi-safe" program, i.e., has a bug iff the original program has a bug, that has fewer execution paths.
- Instruments the program with necessary conditions for failure at several program points.

RESULTS:

- Crab-LLVM: Up to 54% more safe programs!



- KLEE: Up to 30% more bugs found!
- Trimming + KLEE is much faster



EVALUATION:

KLEE: Dynamic symbolic execution engine

Crab-LLVM: Abstract interpreter

Ran on SV-COMP benchmarks