Compiler Validation via Equivalence Modulo Input

Vu Le
Mehrdad Afshari
Zhendong Su

compiler bugs

- Lead to bugs in other programs
- Hard to notice
- Hard to track down
- Weaken source-level analysis & verification

existing approach

- Compiler testing
  - Regression tests
  - Compiler test suites (e.g., Plum Hall, SuperTest)
  - Random program generators (e.g., Csmith[1])
- Verified compilers (e.g., CompCert[2])
- Translation validation

[1] X. Yang, Y. Chen, E. Eide, and J. Regehr. Finding and understanding bugs in C compilers. In *PLDI 2011*.

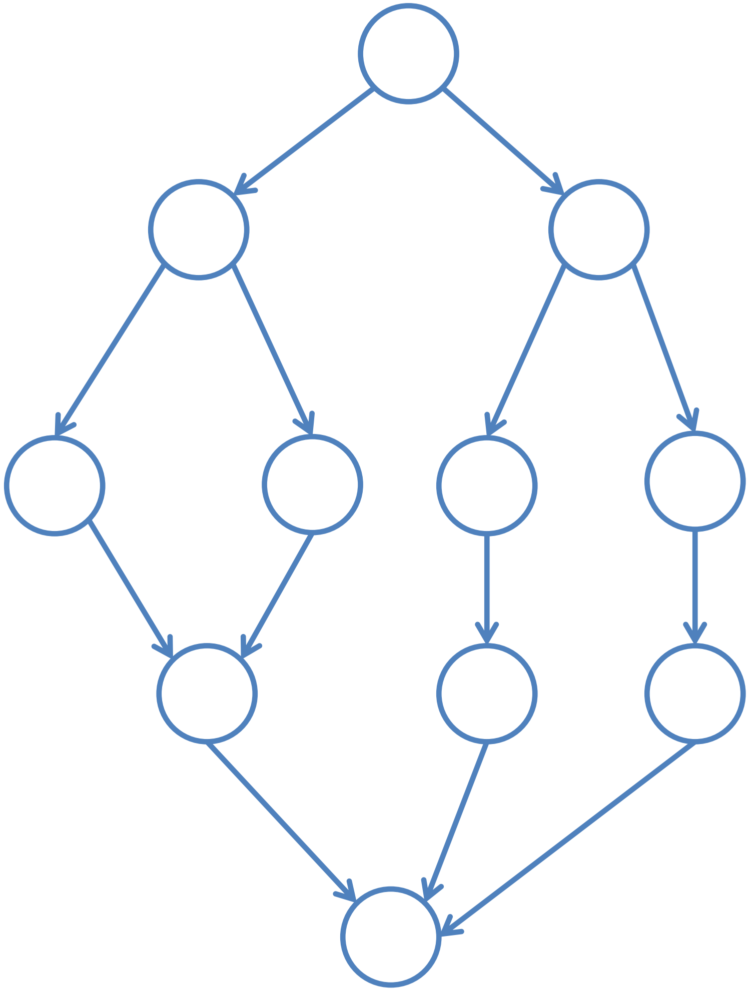
[2] X. Leroy. Formal certification of a compiler back-end, or: programming a compiler with a proof assistant. In *POPL 2006*.

Csmith

- Generate random C programs
- Differential testing: validate against different compilers (or compiler versions)
- Problems
  - Generate bizarre, unnatural code
  - Require different compilers

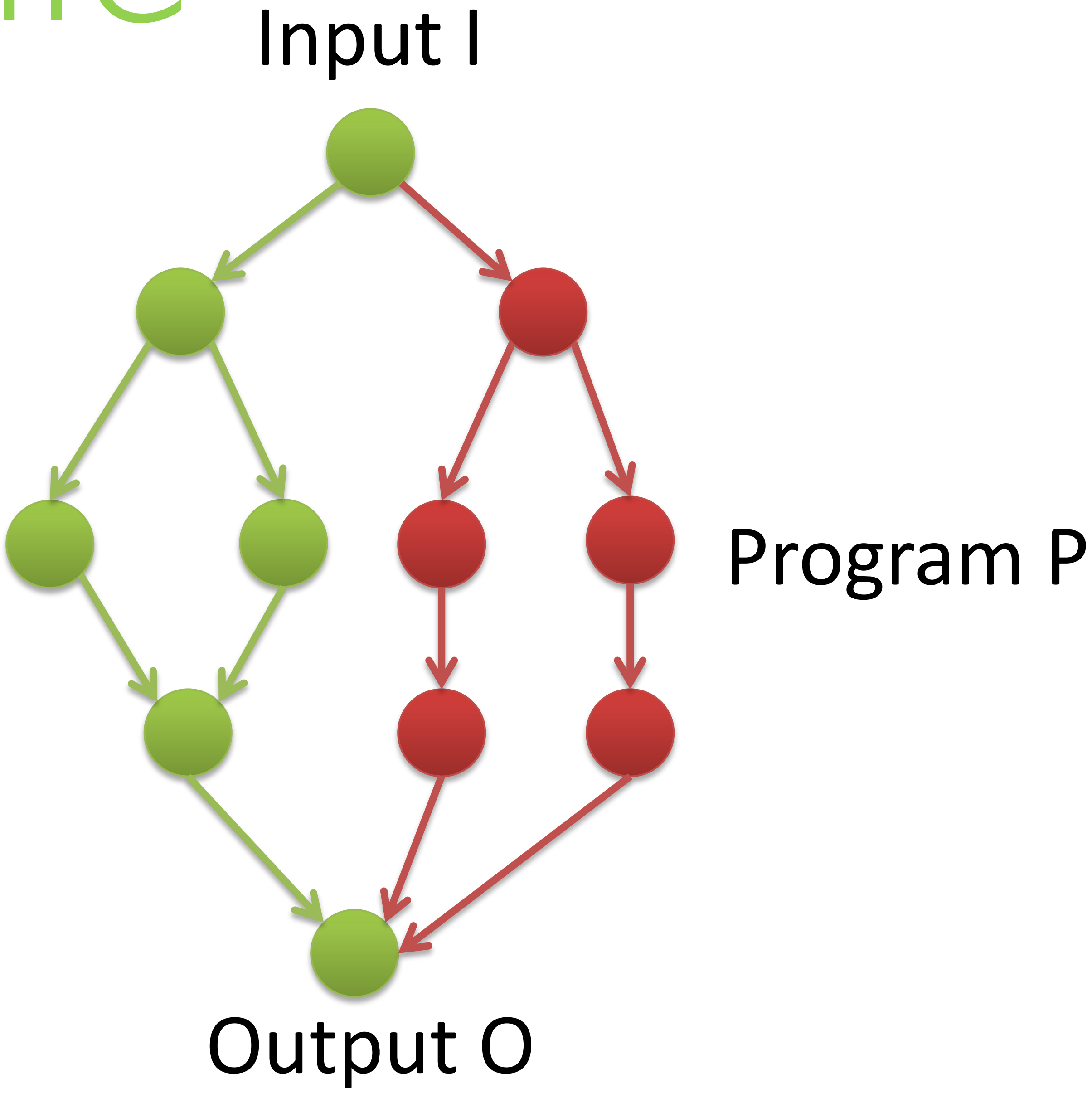
equivalent modulo input

Input I

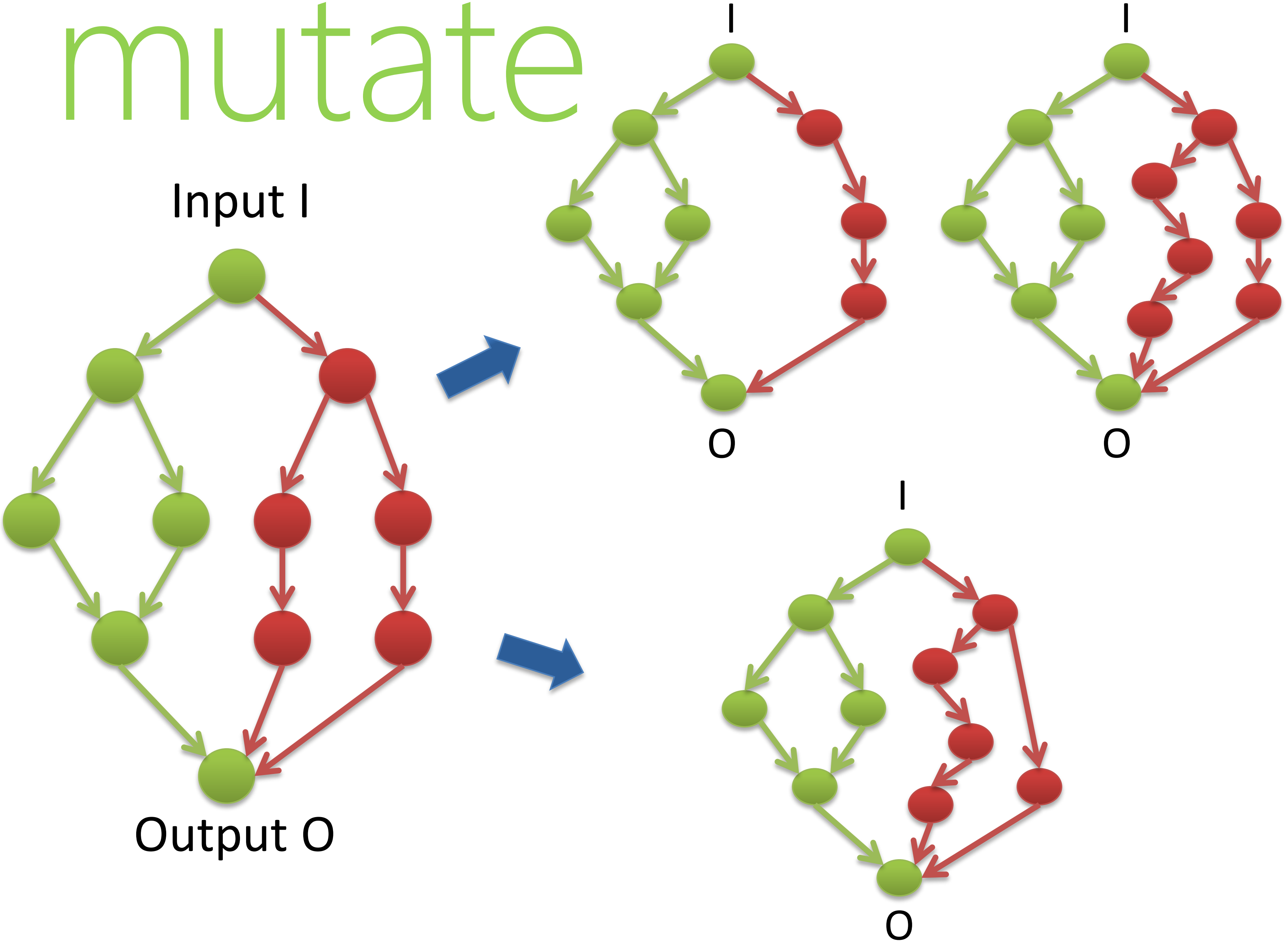


Program P

profile



mutate



equivalent modulo input

- EMI = profile + mutate
- Generate many equivalent variants w.r.t. I
- Advantages
  - General: find bugs in compilers, interpreters, analysis and transformation tools
  - Leverage existing tools
  - Generate real-world tests
  - No need different compilers

orion

- Target C compilers
- Implement pruning strategy
- Workflow
  - Extract coverage information (gcov)
  - Find expected output
  - Loop
    - Generate a variant (clang libtooling)
    - Execute the variant and compare output

evaluation

- Two machines
- Benchmarks
  - Compiler regression test suites
  - Open-source projects
  - Csmith-generated programs
- Feb 13 – Nov 13

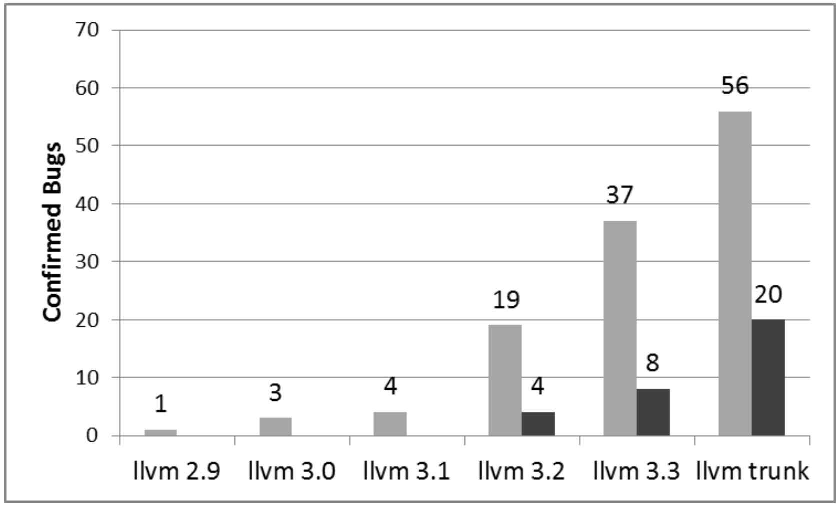
bug counts

	GCC	LLVM	TOTAL
Reported	94	61	155
Confirmed	65	56	121
Fixed	37	20	57

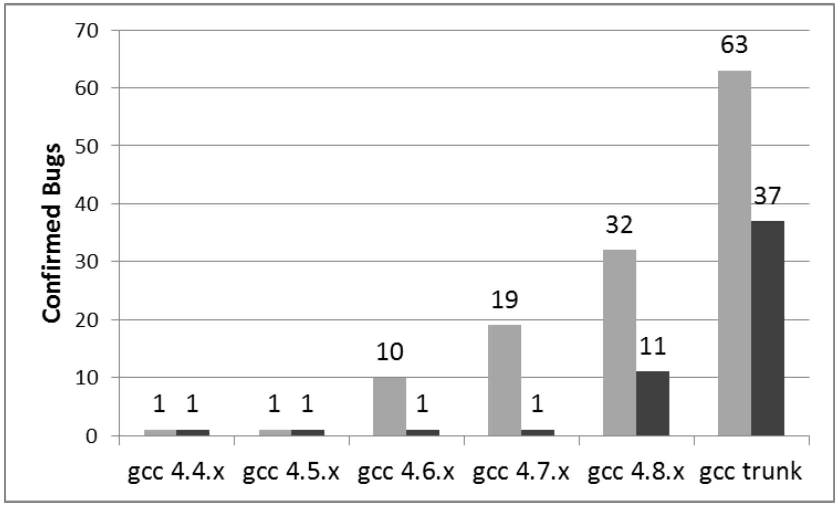
bug types

	GCC	LLVM	TOTAL
Wrong code	40	42	82
Crash	15	4	19
Performance	10	10	20

versions

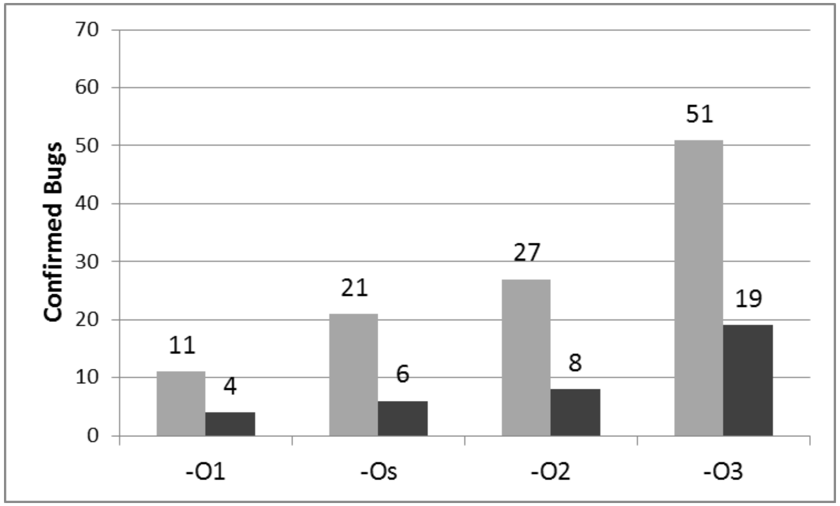


LLVM

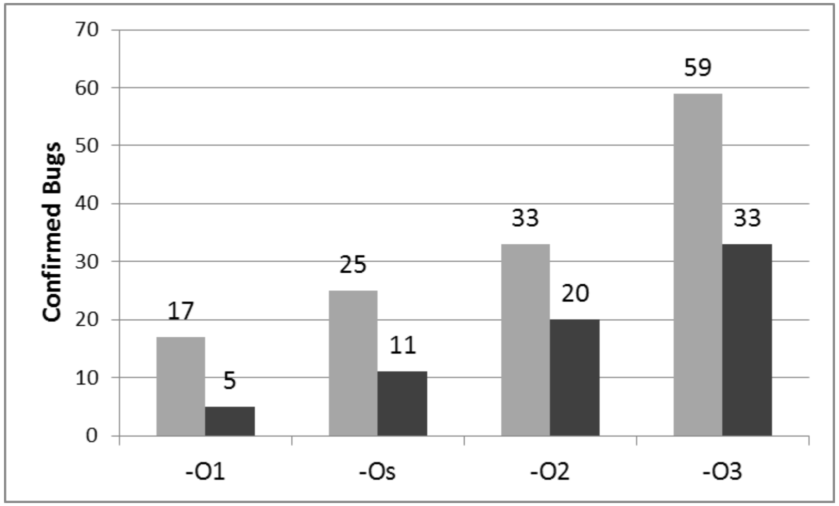


GCC

opt. level

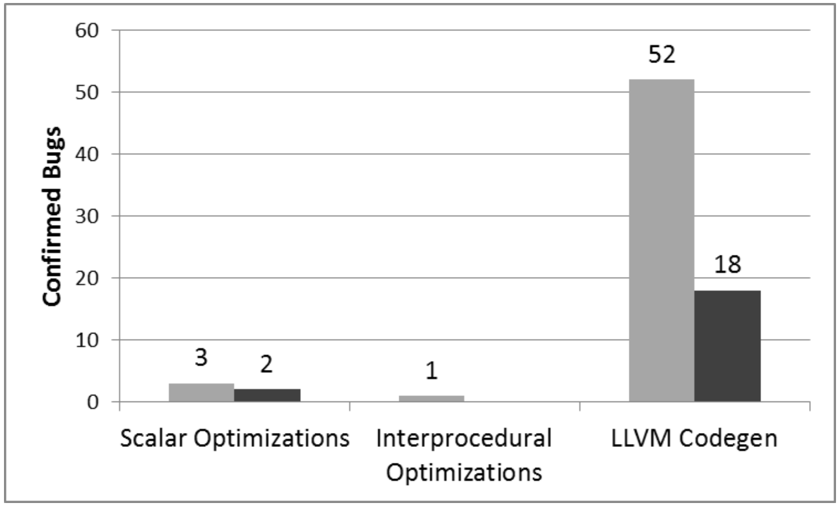


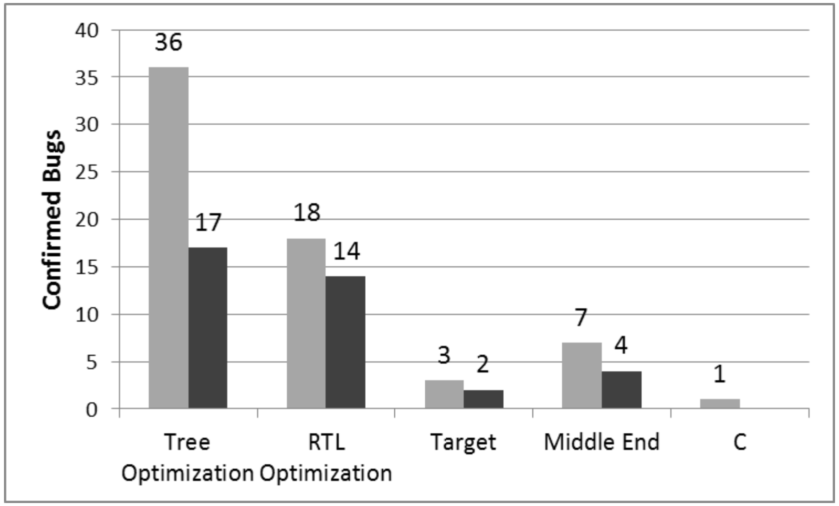
LLVM



GCC

components





LLVM

GCC

thank you