An LLVM Refinement Checker and its Applications

Yiji Zhang¹, Lenore D. Zuck¹, Kedar Namjoshi²

Background
• In optimizing compilers sometimes output ≠ input
• Approaches:
  • Verified optimizers
  • Verify each (input, output)
• Our approach: witness LLVM optimizations

Witness
• Constant propagation
  L1: \( X := 3; \)
  L2: \( Y := 5; \)
  L3: \( Z := X+4; \)
  L4: \( \)

Witness: simulation relation between source and target at each location that describes how source and target variables map

\[ \text{Eg. at } (L4): \ x = X \land y = Y \land z = Z \land X = 3 \land Y = 5 \land Z = 7 \]

Workflow

source LLVM program

pre-coder

pre-coded source

Smack

source Boogie program

Boogie Program Combiner

combined program

Boogie (& Z3)

result

input

Preserve var names

Translate LLVM IR into Boogie

VCs here represent the refinement checks

Generate and check VCs