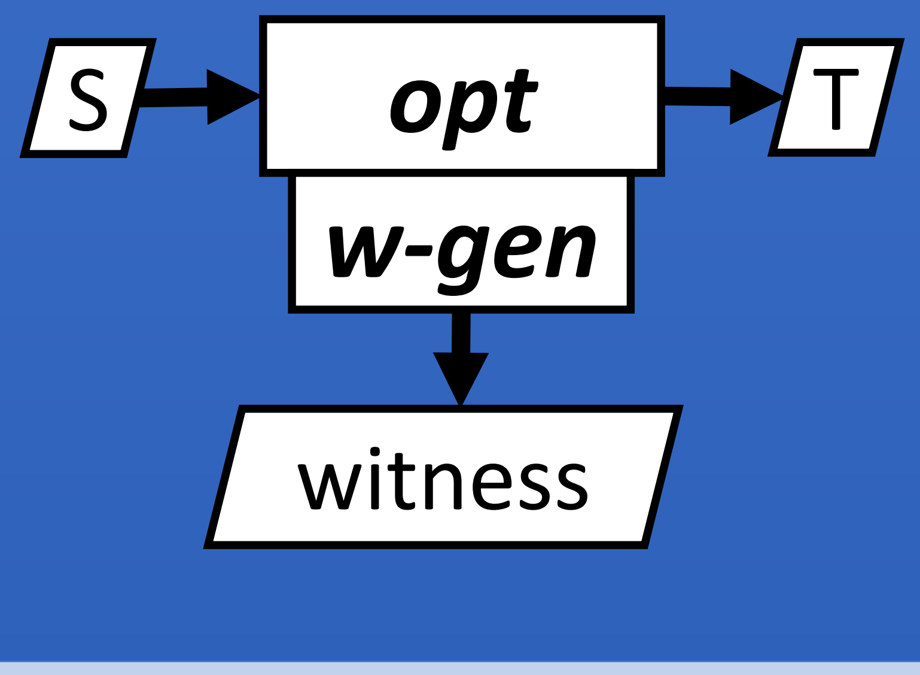
An LLVM Refinement Checker and its Applications

Yiji Zhang[1], Lenore D. Zuck[1], Kedar Namjoshi[2] UIC[1] NOKIA Bell Labs[2]

Background

- In optimizing compilers sometimes

  output $\overset{\text{sem}}{\neq}$ input
- Approaches:
  - Verified optimizers
  - Verify each (input, output)
- Our approach: witness LLVM optimizations



Witness

- Constant propagation

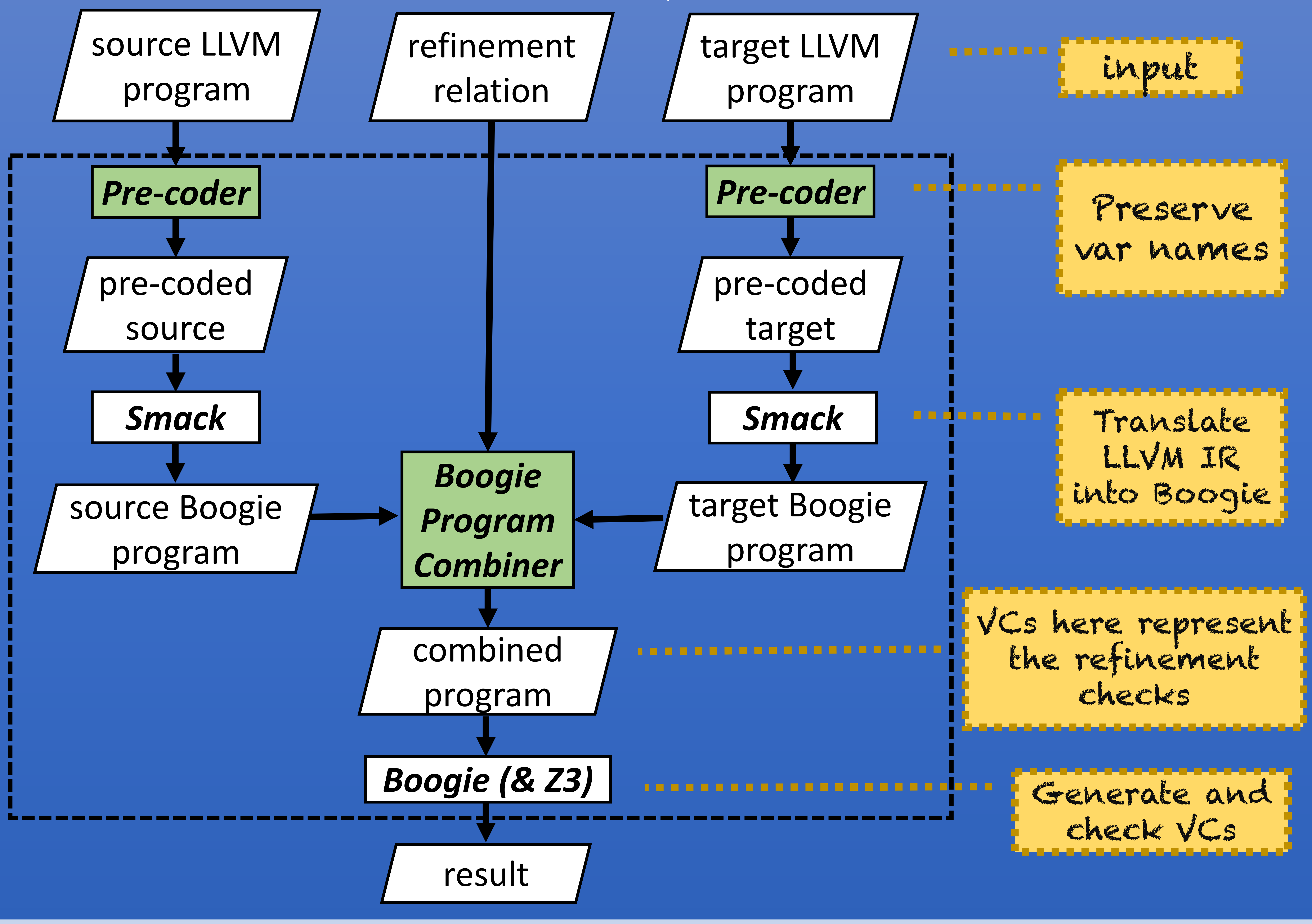
```
L1: X := 3;      L1: x := 3;
L2: Y := 5;      L2: y := 5;
L3: Z := X+4;    L3: z := 7;
L4:              L4:
```

source target

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

Eg. $at(L4)$: $x = X \wedge y = Y \wedge z = Z \wedge X = 3 \wedge Y = 5 \wedge Z = 7$

Workflow


source LLVM program
refinement relation
target LLVM program
input
Pre-coder
Pre-coder
Preserve var names
pre-coded source
pre-coded target
Smack
Smack
Translate LLVM IR into Boogie
source Boogie program
Boogie Program Combiner
target Boogie program
combined program
VCs here represent the refinement checks
Boogie (& Z3)
Generate and check VCs
result