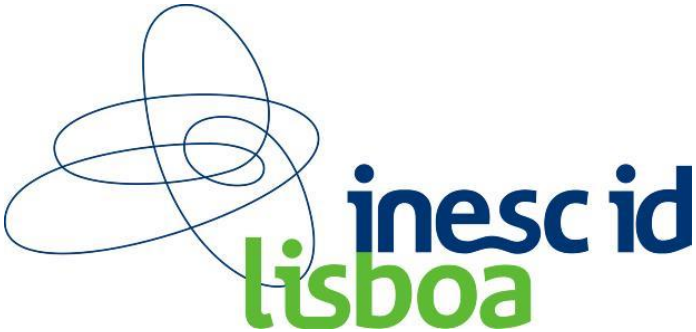
technology
from seed

Weakest Precondition Synthesis for Compiler Optimizations

Nuno Lopes and José Monteiro



TÉCNICO LISBOA

Why WP Synthesis for Compiler Optimizations?

- Deriving preconditions by hand is hard; WPs are often non-trivial
- WPs derived by hand are often wrong!
- Weaker preconditions expose more optimization opportunities

Motivation: Compilers are Full of Bugs

- Yang, Chen, Eide, Regehr. Finding and Understanding Bugs in C Compilers, PLDI'12:
  - 79 bugs in GCC (25 P1)
  - 202 bugs in LLVM
  - 2 wrong-code bugs in CompCert

- 32 open P1 bug reports in GCC (as of last week)
- 403 open wrong-code bug reports in GCC
- 16 open wrong-code bug reports in LLVM

Verification to the Rescue: LLVM PR17827

lib/Transforms/InstCombine/InstCombineCompares.cpp

```
// For a logical right shift, we can fold if the comparison is not
// signed. We can also fold a signed comparison if the shifted mask
// value and the shifted comparison value are not negative.
// These constraints are not obvious, but we can prove that they are
// correct using an SMT solver such as "Z3" :
// http://rise4fun.com/Z3/Tslfh
```

```
if (ShiftOpcode == Instruction::AShr) {
    // There may be some constraints that make this possible,
    // but nothing simple has been discovered yet.
    CanFold = false;
}
```

Outline

- Preliminaries
- Language of Preconditions
- Example
- Algorithm
- Evaluation: PSyCO

Outline

- Preliminaries
- Language of Preconditions
- Example
- Algorithm
- Evaluation: PSyCO

Compiler Optimizations

- Compiler optimization
  - Transformation function
  - Precondition
  - Profitability heuristic

Loop Unswitching


```
while I < N do
    if B then
        S1
    else
        S2
    I := I + 1
```

→

```
if B then
    while I < N do
        S1
        I := I + 1
else
    while I < N do
        S2
        I := I + 1
```

S_1, **S_2** are template statements
B is a template Boolean expression

Loop Unswitching: Example Instantiation


```
…
while I < N do
    if N > 5 then
        A := A + N
    else
        A := A + 1
    I := I + 1
…
```

→

```
if N > 5 then
    while I < N do
        A := A + N
        I := I + 1
else
    while I < N do
        A := A + 1
        I := I + 1
```

Instantiation:

$B \longmapsto N > 5$

$S_1 \longmapsto A := A + N$

$S_2 \longmapsto A := A + 1$

Loop Unswitching: Weakest Precondition


```
while I < N do
    if B then
        S1
    else
        S2
    I := I + 1
```

→

```
if B then
    while I < N do
        S1
        I := I + 1
else
    while I < N do
        S2
        I := I + 1
```

Precondition:

$I \notin R(B) \wedge$
$W(S_1) \cap R(B) = \emptyset \wedge$
$W(S_2) \cap R(B) = \emptyset$

Outline

- Preliminaries
- Language of Preconditions
- Example
- Algorithm
- Evaluation: PSyCO

Language of Preconditions

- Read and Write sets for each template statement/expression
- Arbitrary constraints over read/write sets
- In practice constraints are only over R/W and W/W intersection
  - $v \notin R(B)$
  - $W(S_1) \cap R(B) = \emptyset$
  - $W(S_1) \cap W(S_2) = \emptyset$

Language of Preconditions: Suitability

- Books and developers already informally speak about read and write sets
- Can be efficiently discharged using current compiler technology:
  - Memory dependence analysis
  - Alias/pointer analysis
  - Loop analysis
  - Range analysis
  - …

Outline

- Preliminaries
- Language of Preconditions
- Example
- Algorithm
- Evaluation: PSyCO

Synthesizing WP for Loop Unswitching


```
while I < N do
    if B then
        S1
    else
        S2
    I := I + 1
```

→

```
if B then
    while I < N do
        S1
        I := I + 1
else
    while I < N do
        S2
        I := I + 1
```

1) Find counterexample


```
while I < N do
    if B then
        S1
    else
        S2
    I := I + 1
```

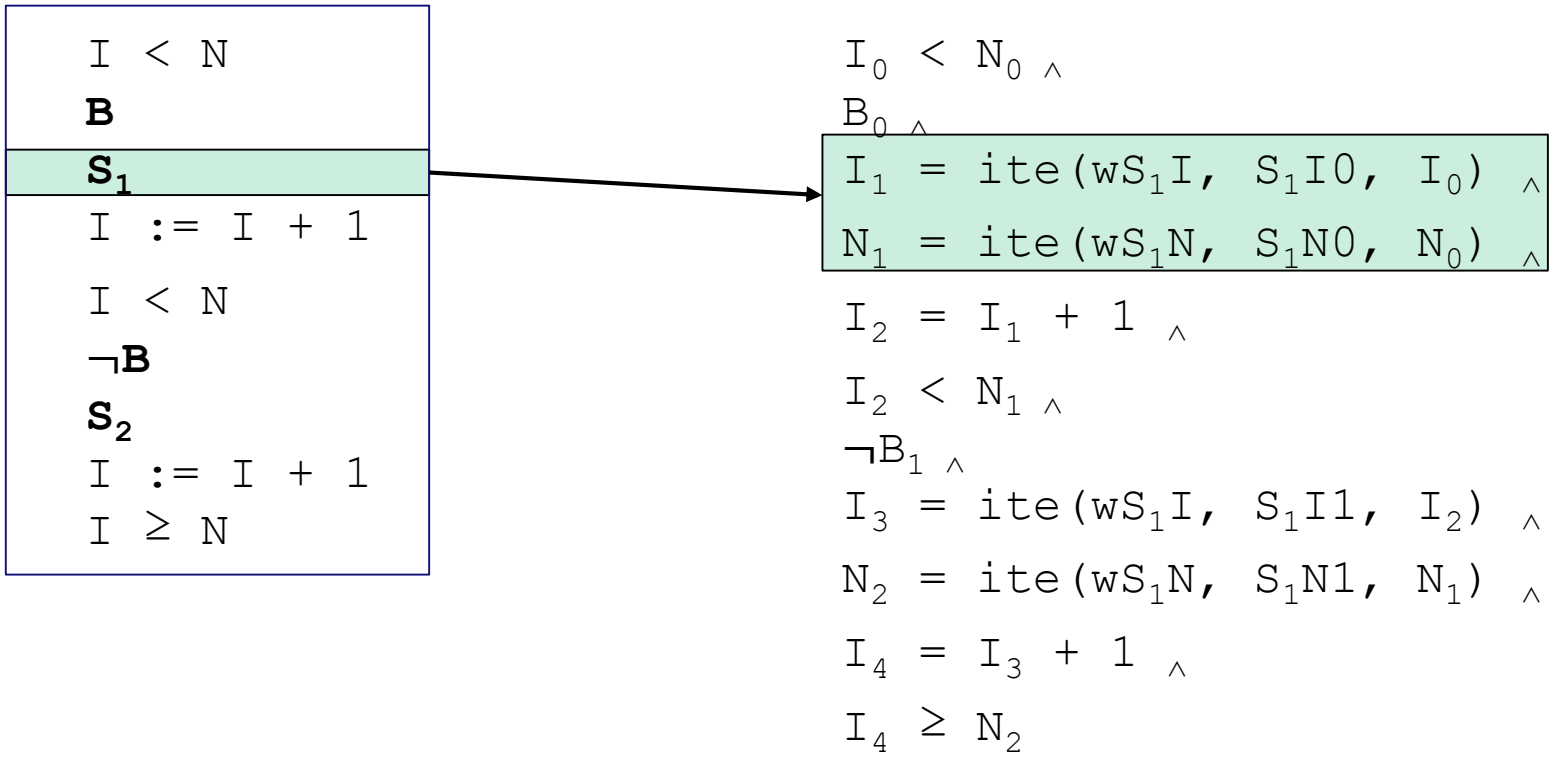
→

```
if B then
    while I < N do
        S1
        I := I + 1
else
    while I < N do
        S2
        I := I + 1
```

Pre = true

I < N	**B**
B	I < N
S_1	**S_1**
I := I + 1	I := I + 1
I < N	I < N
¬B	**S_1**
S_2	I := I + 1
I := I + 1	I ≥ N
I ≥ N	

2) Synthesize WP for counterexample: VC Gen



2) Synthesize WP for counterexample: Conditional Ackermannization

$I_0 < N_0 \wedge$
$B_0 \wedge$
$I_1 = ite(wS_1I, S_1I0, I_0) \wedge$
$N_1 = ite(wS_1N, S_1N0, N_0) \wedge$
$I_2 = I_1 + 1 \wedge$
$I_2 < N_1 \wedge$
$\neg B_1 \wedge$
$I_3 = ite(wS_1I, S_1I1, I_2) \wedge$
$N_2 = ite(wS_1N, S_1N1, N_1) \wedge$
$I_4 = I_3 + 1 \wedge$
$I_4 \geq N_2$

B_0 and B_1 are equal if the values of the variables in R(B) are equal

$$\begin{pmatrix}(I \in R(B) \rightarrow I_0 = I_2) \wedge \\ (N \in R(B) \rightarrow N_0 = N_1)\end{pmatrix} \rightarrow B_0 = B_1$$

2) Synthesize WP for counterexample: Must-write vs may-write

$$
\begin{aligned}
& I_0 < N_0 \wedge \\
& B_0 \wedge \\
& I_1 = \text{ite}(wS_1I, S_1I0, I_0) \wedge \\
& N_1 = \text{ite}(wS_1N, S_1N0, N_0) \wedge \\
& I_2 = I_1 + 1 \wedge \\
& I_2 < N_1 \wedge \\
& \neg B_1 \wedge \\
& I_3 = \text{ite}(wS_1I, S_1I1, I_2) \wedge \\
& N_2 = \text{ite}(wS_1N, S_1N1, N_1) \wedge \\
& I_4 = I_3 + 1 \wedge \\
& I_4 \geq N_2
\end{aligned}
$$

If a variable is in the write set of a statement, it may or may not be written.

$$wS_1I \rightarrow I \in W(S_1)$$
$$wS_1N \rightarrow N \in W(S_1)$$

2) Synthesize WP for counterexample: Final constraint

inesc id lisboa

```
I < N                    B
B                        I < N
S1                       S1
I := I + 1               I := I + 1
-------------------------------------------
I < N                    I < N
¬B                       S1
S2                       I := I + 1
I := I + 1               I ≥ N
I ≥ N
```

$$\exists S \; \forall V \; Path \wedge Ackermann \wedge MustWrite \wedge \ldots \rightarrow PathIsCorrect$$

S = Read/Write sets

V = Vars from VCGen, Must-write vars

A possible model:

$W(S_1) = \emptyset$

$R(S_1) = \emptyset$

$R(B) = \emptyset$

2) Synthesize WP for counterexample: Disjunction of all models

`I < N`	**B**
B	`I < N`
S_1	**S_1**
`I := I + 1`	`I := I + 1`
`I < N`	`I < N`
¬B	**S_1**
S_2	`I := I + 1`
`I ≥ N`	`I ≥ N`

Precondition:

$I \notin R(B) \wedge$

$W(S_1) \cap R(B) = \emptyset$

3) Iterate until no more counterexamples can be found


```
while I < N do
    if B then
        S1
    else
        S2
    I := I + 1
```

→

```
if B then
    while I < N do
        S1
        I := I + 1
else
    while I < N do
        S2
        I := I + 1
```

Precondition:

$I \notin R(B) \wedge$
$W(S_1) \cap R(B) = \emptyset \wedge$
$W(S_2) \cap R(B) = \emptyset$

Outline

- Preliminaries
- Language of Preconditions
- Example
- Algorithm
- Evaluation: PSyCO

Algorithm

1) Find counterexample
2) Generate WP that rules out the counterexample
3) Iterate until no more counterexamples can be found

Optimizations

- Model generalization
- Exploit UNSAT cores
- Bias towards R/W and W/W intersections

Outline

- Preliminaries
- Language of Preconditions
- Example
- Algorithm
- Evaluation: PSyCO

PSyCO: Precondition Synthesizer for Compiler Optimizations

- About 1,400 lines of Python
- Uses Z3 for constraint solving
- Source code and benchmarks available from http://goo.gl/7K02H9

PSyCO: Results

Optimization	# Counterexamples	# Models	WP Time	Total Time
Code hoisting	1	1	0.07s	0.23s
Constant propagation	1	1	0.04s	0.16s
Copy propagation	0	0	0s	0.11s
If-conversion	0	0	0s	0.11s
Partial redundancy elimin.	1	1	0.10s	0.30s
Loop fission	6	36	1.28s	2.18s
Loop flattening	1	1	0.07s	3.31s
Loop fusion	6	36	1.26s	2.19s
Loop interchange	11	25	1.42s	23.8s
Loop invariant code motion	3	3	0.22s	0.55s
Loop peeling	0	0	0s	0.27s
Loop reversal	4	7	0.25s	0.54s
Loop skewing	1	1	0.06s	163s
Loop strength reduction	1	2	1.14s	1.41s
Loop tiling	1	1	0.07s	4.60s
Loop unrolling	2	4	0.13s	0.50s
Loop unswitching	2	2	0.15s	0.77s
Software pipelining	1	2	0.13s	0.58s

Example of Synthesized WP: Software Pipelining

inesc id lisboa

$$\textbf{while } V_1 < V_2 \textbf{ do} \quad \Rightarrow \quad \textbf{if } V_1 < V_2 \textbf{ then}$$

S_1	S_1
S_2	**while** $V_1 < (V_2 - 1)$ **do**
$V_1 := V_1 + 1$	S_2
	$V_1 := V_1 + 1$
	S_1
	S_2
	$V_1 := V_1 + 1$

Precondition:

$$\begin{aligned}
&V_2 \notin \mathsf{W}(S_2) \wedge \\
&\big((\mathsf{R}(S_1) \cap \mathsf{W}(S_2) = \emptyset \wedge \\
&\ \mathsf{R}(S_1) \cap \mathsf{W}(S_1) = \emptyset \wedge \\
&\ \mathsf{R}(S_2) \cap \mathsf{W}(S_2) = \emptyset) \vee \\
&\ V_1 \notin \mathsf{W}(S_2)\big)
\end{aligned}$$

(Weaker than PEC's [PLDI'09])

Conclusion

- Deriving WPs by hand is hard and error-prone
- Weaker preconditions enable more optimization opportunities
- Presented the first algorithm for the automatic synthesis of WPs for compiler optimizations

inesc id
lisboa