Value Range Propagation
in
LLVM

Adam Wiggins
Patryk Zadarnowski
{awiggins,patrykz}@cse.unsw.edu.au

17 June 2003

University of New South Wales
Sydney

MOTIVATION

The Problem:

- ✗ Layout of basic blocks in LLVM is brain-dead.

The Solution:

- Calculate the probability of each branch being taken.
- Arrange the most probably control flow path in a straight line.

Possible Approaches:

- ✗ Code annotations
- ✗ Static heuristics
- ✗ Profiling
- ☑ Value range propagation

Value Range Propagation

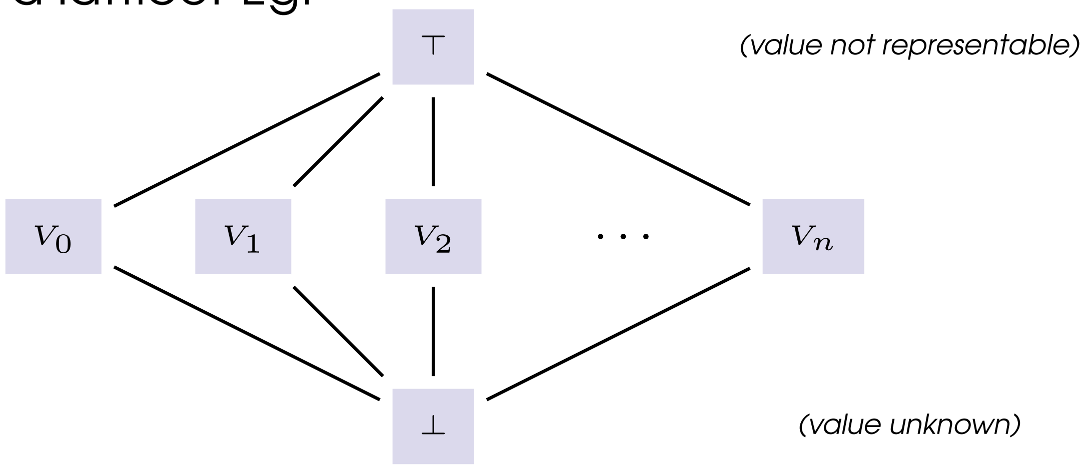
```
for (x = 0; x < 10; ++x)
{
    if (x > 7)
      y = 1;
    else
      y = x;
    // ...
    if (y == 1)
      // SOME CODE
    // ...
  }
```

Need information on possible ranges for variable values!

Data-Flow Analysis by Abstract Interpretation

➜ Interpret the program sequentially as if executing it, but:

➜ Instead of run-time values, operates over abstract values arranged into a lattice. Eg:



➜ When interpretation terminates, the environment contains information for all variables in the program encoded as abstract values.

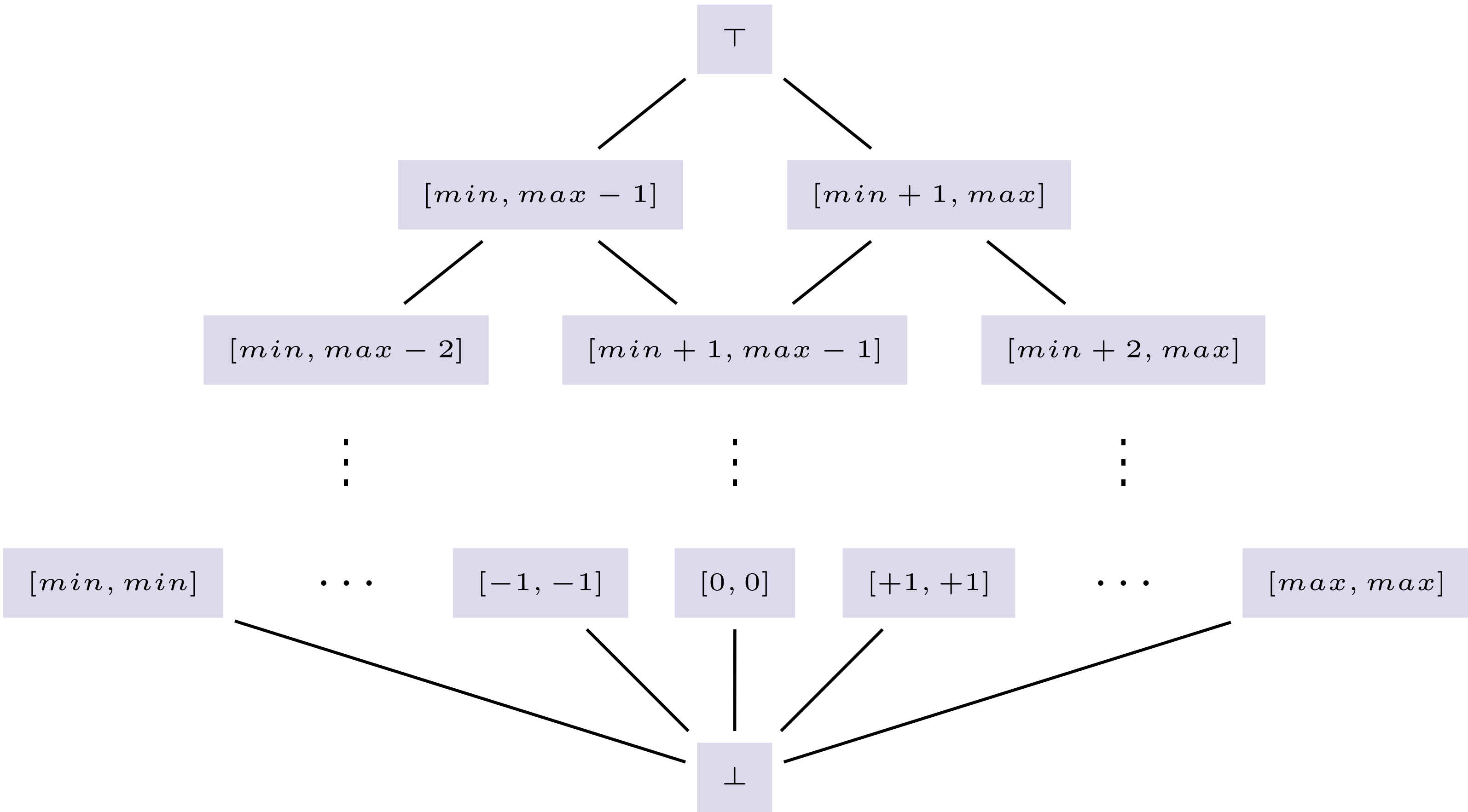
➜ But what do we do about the loops?

FIXPOINT CALCULATION

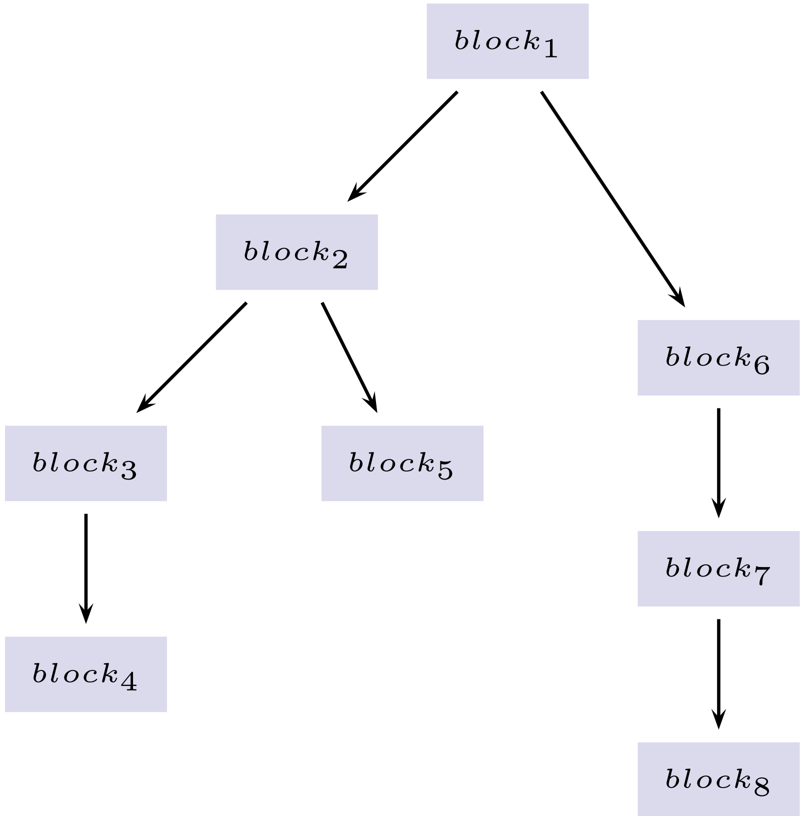
- ➜ Interpret the program along every possible path through the control flow graph (CFG).
- ➜ For ϕ nodes, merge the values calculated along each path using a suitable $\sqcap$ (meet) function.
- ➜ If $\sqcap$ changed the abstract value, re-evaluate the block.
- ➜ To ensure termination, define $\sqcap$ so that an abstract value can change only a finite number of times:
  - For each change, make sure that $\sqcap$ raises the value to a higher level in the lattice.
- ➜ When nothing changes, we've reached the fixpoint!

Infinite Lattices

Problem: value range propagation lattices are TALL!

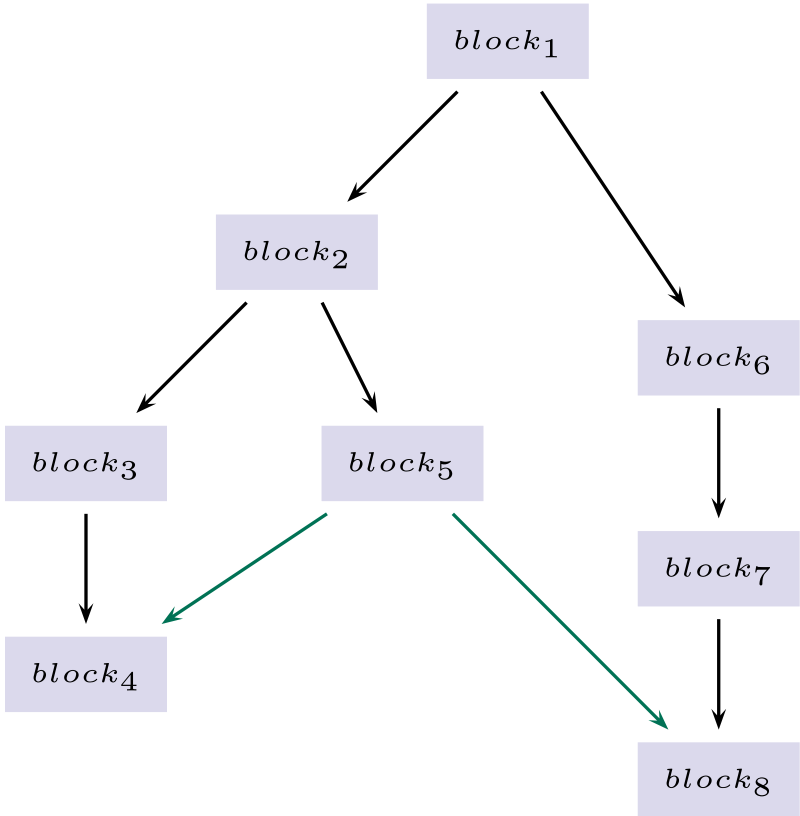


Reducing the Problem I



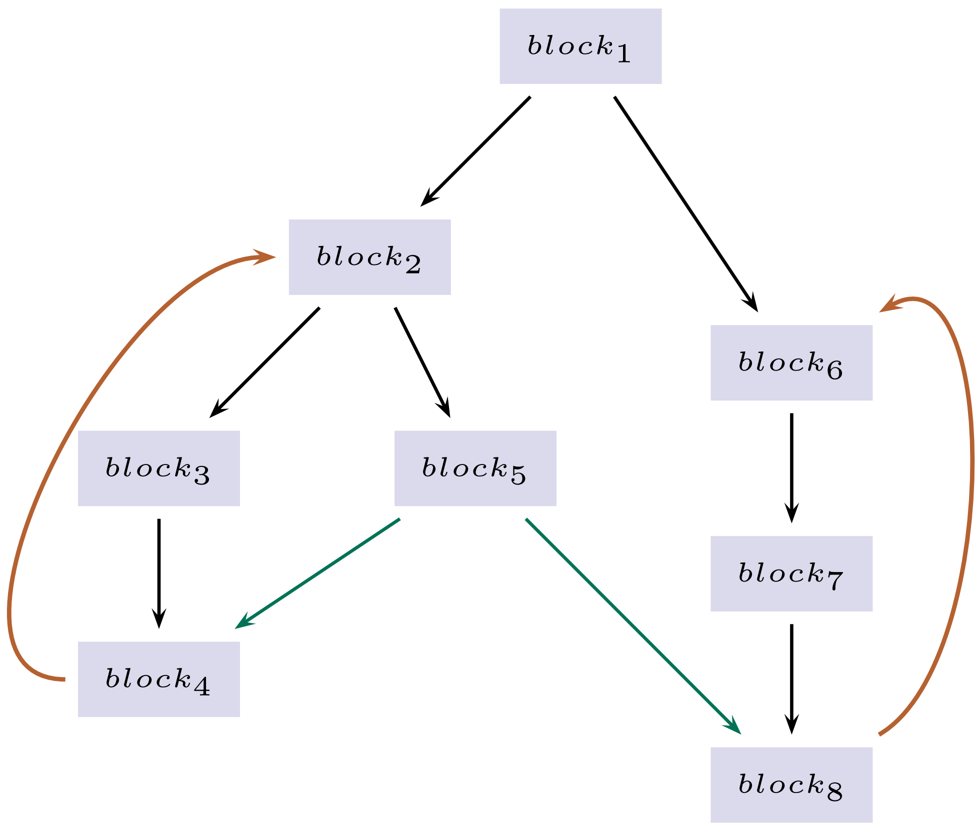
No merging, no problem.

Reducing the Problem II



With cross-edges, number of merges is bound.

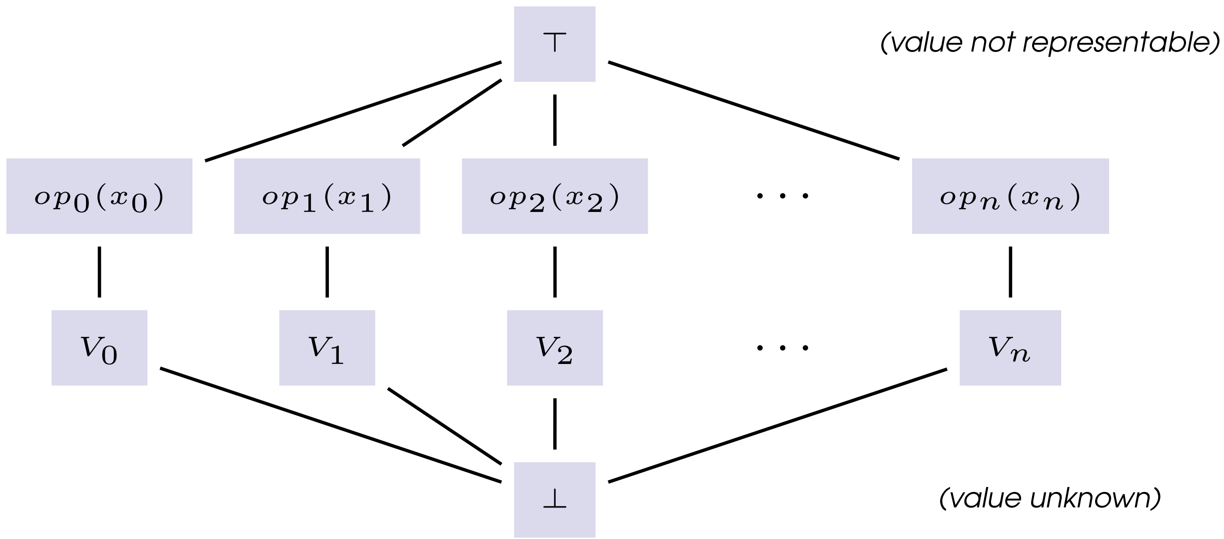
Reducing the Problem III



But with back-edges (i.e. loops) we hit the brick wall!

Symbolic Evaluation

- ➜ Flatten the matrix to a single level for all “constant cells”.
- ➜ Allow non-raising $\sqcap$ when following normal edges.
- ➜ To handle back-edges, introduce a symbolic level into the lattice:



- ➜ Always raise the level when merging along back-edges.
- ➜ Computes symbolic expressions instead of constant values for loop-indexed variables.
- ➜ Symbolic values can be folded once the fixpoint is reached.

Our Iterative Approach I

- ➜ Similar to SCC$_{ANF}$ (see: (CKZ2003))
- ➜ We adopt it for SSA and extend to support value numbering.
- ➜ Numerous advantages compared to other published approaches:
  - ☑ An iterative approach, therefore works with our symbolic lattice.
  - ☑ Good asymptotic complexity.
  - ☑ Supports inter-procedural analysis transparently.
  - ☑ Rigorous formal definition and soundness proof.
  - ☑ Can be extended to handle higher-order programs (function pointer variables.)

Our Iterative Approach II

- ➜ Collects rich information about integers:
  - minimum, maximum, stride
  - bits known to be set and cleared
  - lists of the above qualified by boolean predicates
- ➜ Collects probabilities for boolean conditions.
- ➜ Supports multiple predicated ranges in each lattice cell.
- ➜ The $\sqcap$ function:

$$
\begin{array}{rl}
 & \{0.7[32:256:1], 0.3[3:21:3]\} \\
+ & \{0.6[16:100:4], 0.4[8:8:0]\} \\
\hline
= & \{0.42[48:356:1], 0.28[40:264:1], \\
 & \quad 0.18[19:121:1], 0.12[11:29:3]\}
\end{array}
$$

Our Iterative Approach III

- Data Structures:
  - A work list Ω of blocks to be processed.
  - Environment Γ mapping variables to abstract values.
  - Environment R mapping blocks to abstract values.
- Operation:
  - ϕ nodes treated as formal parameters to blocks in which they appear.
  - Start with the entry block of `main` in Ω.
  - Iterate until Ω empty.
  - Update items in Γ and R using $\sqcap$ whenever recomputing a variable already there. For jumps and function calls simply use the current information from R.
  - If $\sqcap$ changes Γ or R, add all blocks (functions) referencing that variable back to Ω.

CONCLUSIONS

➜ Collects a lot of inter-procedural data-flow information:

- value ranges
- value probabilities
- flow of data across conditional branches

➜ Many different optimizations can utilize range information:

- basic block layout
- branch prediction hints
- inter-procedural conditional constant propagation
- variable retyping and resizing
- data packing for vector instructions
- data speculation on IA-64.

➜ The statically-computed results of the analysis can complement profiling information in the same analysis group.

➜ The algorithm is not restricted to collecting a particular kind of information.

```
letrec f(x) =


   let x’ = sub(x, x)


   in
     if x’ then
       f(x’)
     else
       x


in
   f(7)
```

```
letrec f(x) =


  let x' = sub(x, x)


  in
    if x' then
      f(x')
    else
      x


in
  f(7)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{f\}$$

```
letrec f(x) =
```

$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$

```
  let x' = sub(x, x)

  in
    if x' then
      f(x')
    else
      x

in
  f(7)
```

$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{f\}$

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

```
   let x' = sub(x, x)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

```
   in
     if x' then
       f(x')
     else
       x

in
   f(7)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{f\}$$

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

```
  let x' = sub(x, x)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

```
  in
    if x' then
      f(x')
    else
      x
```

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{f\}$$

```
in
  f(7)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{f\}$$

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

```
   let x' = sub(x, x)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

```
   in
     if x' then
       f(x')
     else
       x
```

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{f\}$$

```
in
   f(7)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{f\}$$

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{f\}$$

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \perp, x \mapsto 7\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

```
   let x’ = sub(x, x)
```

$$\Gamma = \{f \mapsto \perp, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

```
   in
     if x’ then
       f(x’)
     else
       x
```

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{f\}$$

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{\}$$

```
in
   f(7)
```

$$\Gamma = \{f \mapsto \perp, x \mapsto 7\}, \Omega = \{f\}$$

$$\Gamma = \{f \mapsto 7, x \mapsto 7, x' \mapsto 0\}, \Omega = \{f\}$$

```
letrec f(x) =


   let x' = sub(x, 1)


   in
     if x' then
       f(x')


     else
       x

in
   f(7)
```

```
letrec f(x) =

   let x' = sub(x, 1)

   in
     if x' then
       f(x')

     else
       x

in
   f(7)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \perp, x \mapsto 7\}, \Omega = \{\}$$

```
  let x' = sub(x, 1)
```

$$\Gamma = \{f \mapsto \perp, x \mapsto 7, x' \mapsto 6\}, \Omega = \{\}$$

```
  in
    if x' then
      f(x')


    else
      x

in
  f(7)
```

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

```
  let x' = sub(x, 1)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 6\}, \Omega = \{\}$$

```
  in
    if x' then
      f(x')
```

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{f\}$$

```
    else
      x
in
  f(7)
```

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{\}$$

```
   let x' = sub(x, 1)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 6\}, \Omega = \{\}$$

```
   in
     if x' then
       f(x')
```

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{f\}$$

```
     else
       x

in
   f(7)
```

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{\}$$

```
  let x’ = sub(x, 1)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 6\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto \top\}, \Omega = \{\}$$

```
  in
    if x’ then
      f(x’)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{f\}$$

```
    else
      x
in
  f(7)
```

```
letrec f(x) =
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{\}$$

```
  let x' = sub(x, 1)
```

$$\Gamma = \{f \mapsto \bot, x \mapsto 7, x' \mapsto 6\}, \Omega = \{\}$$

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto \top\}, \Omega = \{\}$$

```
  in
    if x' then
      f(x')
```

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto 6\}, \Omega = \{f\}$$

$$\Gamma = \{f \mapsto \bot, x \mapsto \top, x' \mapsto \top\}, \Omega = \{\}$$

```
    else
      x
```

$$\Gamma = \{f \mapsto \top, x \mapsto \top, x' \mapsto \top\}, \Omega = \{f\}$$

```
in
  f(7)
```