Widening and Narrowing

Yu Zhang

Most content comes from http://cs.au.dk/~amoeller/spa/

1

Interval Analysis

- Compute upper and lower bounds for integers
- Possible applications:
  - array bounds checking
  - integer representation
  - …
- Latticeof intervals:

  Interval = *lift* ({ [l,h] | l,h ∈ *N* ∧ l ≤ h })

  where

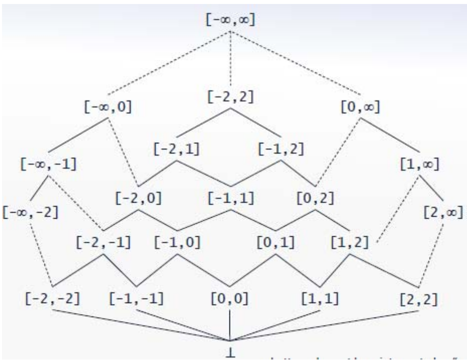
  N= {-∝, ..., -2, -1, 0, 1, 2, ..., ∝}

  and intervals are ordered by inclusion:

  $[l_1,h_1] \sqsubseteq [l_2,h_2]$ iff $l_2 \le l_1 \wedge h_1 \le h_2$

2

The Interval Lattice



3

Interval Analysis Lattice

- The total lattice for a program point is

  $L = Vars \rightarrow Interval$

  that provides bounds for each (integer) variable

- If using the worklist solver that initializes the worklist with only the *entry*node, use the lattice *lift*(L)
  - bottom value of *lift*(L) represents "unreachable program point"
  - bottom value of L represents "maybe reachable, but all variables are non-integers"

- This lattice has *infinite height*, since the chain

  [0,0] ⊑[0,1] ⊑[0,2] ⊑[0,3] ⊑[0,4]...

  occurs in *Interval*

4

Interval Constraints

- For assignments:

  $[\![x = E]\!] = JOIN(v)[x \rightarrow eval(JOIN(v), E)]$

- For all other nodes:

  $[\![v]\!] = JOIN(v)$

where

$$JOIN(v) = \bigsqcup_{w \in pred(v)} [\![w]\!]$$

Least upper bound

5

Evaluating Intervals

- The *eval* functionis an *abstract evaluation*:
  - *eval*(σ, *x*) = σ(*x*)
  - *eval*(σ, *intconst*) = [*intconst*,*intconst*]
  - *eval*(σ, E_1 op E_2) = op(*eval*(σ,E_1),*eval*(σ,E_2))
- Abstract arithmetic operators:

  $\overline{op}([l_1, h_1],[l_2, h_2])$

  $= [\min_{x \in [l_1,h_1], y \in [l_2,h_2]} x \text{ op } y, \max_{x \in [l_1,h_1], y \in [l_2,h_2]} x \text{ op } y]$

  Not trivial to implement

- Abstract comparison operators (could be improved):

  $\overline{op}([l_1, h_1],[l_2, h_2]) = [0,1]$

6

Fixed-point Problems

- The lattice has infinite height, so the fixed-point algorithm does not work ☹
- In L^n, the sequence of approximants

  $f^i(\bot, \bot, ..., \bot)$

  is not guaranteed to converge
- (Exercise: give an example of a program where this happens)

- Restricting to 32 bit integers is not a practical solution
- *Widening* gives a useful solution …

7

Widening

- Introduce a *widening* function $\omega: L^n \to L^n$ so that

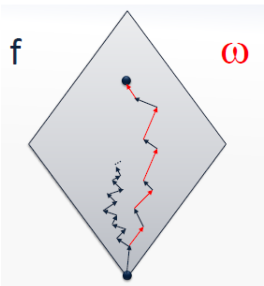
  $(\omega \circ f)^i(\bot, \bot, ..., \bot)$

  converges on a fixed-point that is a safe approximation of each $f^i(\bot, \bot, ..., \bot)$

- i.e. the function ω coarsens the information

8

Turbo Charging the Iterations



9

Widening for Intervals

- The function ω is defined pointwise on L^n
- Parameterized with a fixed finite subset $B \subset N$
  - must contain $-\infty$ and ∞ (to retain the T element)
  - typically seeded with all integer constants occurring in the given program
- Idea: Find the nearest enclosing allowed interval
- On single elements from *Interval* :

  $\omega([a,b]) = [\ max\{i \in B \mid i \leq a\},\ min\{i \in B \mid b \leq i\}\ ]$

  $\omega(\bot) = \bot$

10

Divergence in Action

```
y = 0;
x = 7;
x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \to \bot, y \to \bot]$
$[x \to [8,8], y \to [0,1]]$
$[x \to [8,8], y \to [0,2]]$
$[x \to [8,8], y \to [0,3]]$
…

在while循环之后的程序点的状态

11

Divergence in Action

```
y = 0;
x = 7;
x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \to \bot, y \to \bot]$
$[x \to [7,\infty], y \to [0,1]]$
$[x \to [7,\infty], y \to [0,7]]$
$[x \to [7,\infty], y \to [0,\infty]]$

8

$B = \{-\infty, 0, 1, 7, \infty\}$

12

Correctness of Widening

- Widening works when:
  - ω is an *extensive* and *monotone* function, and
  - $\omega(L)$ is a *finite-height* lattice
- Safety: $\forall i: f^i(\bot, \bot, ..., \bot) \sqsubseteq (\omega \circ f)^i(\bot, \bot, ..., \bot)$
  since f is monotone and ω is extensive
- $\omega \circ f$ is a monotone function $\omega(L) \rightarrow \omega(L)$
  so the fixed-point exists

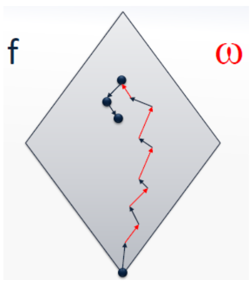
> **Exercise 4.16:** A function $f: L \rightarrow L$ where L is a lattice is *extensive* when $\forall x \in L: x \sqsubseteq f(x)$. Assume L is the powerset lattice $2^{\{0,1,2,3,4\}}$ Give examples

- Almost "correct by definition"!
- When used in the worklist algorithm, it suffices to apply widening on back-edges in the CFG

Narrowing

- Widening generally shoots over the target
- *Narrowing* may improve the result by applying f
- Define:
  $fix = \bigsqcup f^i(\bot, \bot, ..., \bot)$ $\quad fix\omega = \bigsqcup (\omega \circ f)^i(\bot, \bot, ..., \bot)$
  then $fix \sqsubseteq fix\ \omega$
- But we also have that
  $fix \sqsubseteq f(fix\omega) \sqsubseteq fix\ \omega$
  so applying f again may improve the result and remain sound!
- This can be iterated arbitrarily many times
  - may diverge, but safe to stop anytime

Backing up



Narrowing in Action

```
y = 0;
x = 7;
x = x+1;
while (input) {
   x = 7;
   x = x+1;
   y = y+1;
}
```

$[x \rightarrow \bot, y \rightarrow \bot]$
$[x \rightarrow [7,\infty], y \rightarrow [0,1]]$
$[x \rightarrow [7,\infty], y \rightarrow [0,7]]$
$[x \rightarrow [7,\infty], y \rightarrow [0,\infty]]$
...
$[x \rightarrow [8,8], y \rightarrow [0,\infty]]$

$B = \{-\infty, 0, 1, 7, \infty\}$

Correctnessof (Repeated) Narrowing

- $f(fix\omega) \sqsubseteq \omega(f(fix\omega)) = (\omega \circ f)(fix\omega) = fix\omega$
  since ω is extensive
  - by induction we also have, for all i:
    $f^{i+1}(fix\omega) \sqsubseteq f^i(fix\omega) \sqsubseteq fix\omega$
  - i.e. $f^{i+1}(fix\omega)$ is at least as precise as $f^i(fix\omega)$
- $fix \sqsubseteq fix\omega$ hence $f(fix) = fix \sqsubseteq f(fix\omega)$
  by monotonicity of f
  - by induction we also have, for all i:
    $fix \sqsubseteq f^i(fix\omega)$
  - i.e. $f^i(fix\omega)$ is a sound approximation of *fix*

More Powerful Widening

- Defining the widening function based on constants occurring in the given program may not work

```
f(x) { // "McCarthy's 91 function"
  var r;
  if (x > 100) {
    r = x - 10;
  } else {
    r = f(f(x + 11));
  }
  return r;
}
```

https://en.wikipedia.org/wiki/McCarthy_91_function

- Note: this example requires interprocedural analysis...

More Powerful Widening

- A *widening* is a function ∇: L × L →L that is extensive in both arguments and satisfies the following property:
  for all increasing chains $z_0 \sqsubseteq z_1 \sqsubseteq \ldots$,
  the sequence $y_0 = z_0, \ldots, y_{i+1} = y_i \nabla z_{i+1}, \ldots$ converges
  (i.e. stabilizes after a finite number of steps)

- Now replace the basic fixed point solver by computing $x_0 = \bot, \ldots, x_{i+1} = x_i \nabla F(x_i), \ldots$ until convergence

19

More Powerful Widening for Interval Analysis

- Extrapolates unstable bounds to B:

$\bot \nabla y = y$
$x \nabla \bot = x$
$[a_1, b_1] \nabla [a_2, b_2] =$
$[$if $a_1 \leq a_2$ then a_1 else $max\{i \in B \mid i \leq a_2\}$,
if $b_2 \leq b_1$ then b_1 else $min\{i \in B \mid b_2 \leq i\}]$

The ∇ operator on L is then defined pointwise down individual intervals

For the small example program, we now get the same result as with simple widening plus narrowing (but now without using narrowing)

20