Static Program Analysis
Part 5 – widening and narrowing

http://cs.au.dk/~amoeller/spa/

Anders Møller & Michael I. Schwartzbach
Computer Science, Aarhus University
Interval analysis

• Compute upper and lower bounds for integers
• Possible applications:
  – array bounds checking
  – integer representation
  – ...
• Lattice of intervals:

\[
\text{Interval} = \text{lift}(\{ [l, h] \mid l,h \in N \land l \leq h \})
\]

where

\[
N = \{-\infty, \ldots, -2, -1, 0, 1, 2, \ldots, \infty\}
\]

and intervals are ordered by inclusion:

\[
[l_1, h_1] \subseteq [l_2, h_2] \text{ iff } l_2 \leq l_1 \land h_1 \leq h_2
\]
The interval lattice

The bottom element here interpreted as “not an integer”
Interval analysis lattice

• The total lattice for a program point is

\[ L = \text{Vars} \rightarrow \text{Interval} \]

that provides bounds for each (integer) variable

• If using the worklist solver that initializes the worklist
  with only the entry node, use the lattice \( \text{lift}(L) \)
  — bottom value of \( \text{lift}(L) \) represents “unreachable program point”
  — bottom value of \( L \) represents “maybe reachable, but all variables are non-integers”

• This lattice has \textit{infinite height}, since the chain

\[ [0,0] \subseteq [0,1] \subseteq [0,2] \subseteq [0,3] \subseteq [0,4] \ldots \]

occurs in \textit{Interval}
Interval constraints

• For assignments:

\[
\llbracket x = E \rrbracket = JOIN(v)[x \mapsto eval(JOIN(v), E)]
\]

• For all other nodes:

\[
\llbracket v \rrbracket = JOIN(v)
\]

where \( JOIN(v) = \bigsqcup \llbracket w \rrbracket \) for \( w \in pred(v) \)
Evaluating intervals

- The `eval` function is an abstract evaluation:
  - `eval(\sigma, x) = \sigma(x)`
  - `eval(\sigma, \text{intconst}) = [\text{intconst}, \text{intconst}]`
  - `eval(\sigma, E_1 \text{ op } E_2) = \text{op}(eval(\sigma,E_1),eval(\sigma,E_2))`

- Abstract arithmetic operators:
  - `\text{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]`

- Abstract comparison operators (could be improved):
  - `\overline{\text{op}}([l_1, h_1], [l_2, h_2]) = [0, 1]`
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...
Widening

• Introduce a \textit{widening} function $\omega: L^n \rightarrow L^n$ so that

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

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

• i.e. the function $\omega$ coarsens the information
Turbo charging the iterations
Widening for intervals

- The function $\omega$ is defined pointwise on $L^n$
- Parameterized with a fixed finite subset $B \subseteq \mathbb{N}$
  - must contain $-\infty$ and $\infty$ (to retain the $\top$ element)
  - typically seeded with all integer constants occurring in the given program
- Idea: Find the nearest enclosing allowed interval
- On single elements from $Interval$:
  \[
  \omega([a, b]) = [\max\{i \in B | i \leq a\}, \min\{i \in B | b \leq i\}]
  \]
  \[
  \omega(\bot) = \bot
  \]
Divergence in action

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

- \([x \rightarrow \bot, y \rightarrow \bot]\)
- \([x \rightarrow [8,8], y \rightarrow [0,1]]\)
- \([x \rightarrow [8,8], y \rightarrow [0,2]]\)
- \([x \rightarrow [8,8], y \rightarrow [0,3]]\)
...
Widening in action

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

\[ B = \{-\infty, 0, 1, 7, \infty\} \]
Correctness of widening

• Widening works when:
  – $\omega$ is an extensive and monotone function, and
  – $\omega(L)$ is a finite-height lattice

• Safety: $\forall i: f_i(\bot, \bot, ..., \bot) \subseteq (\omega \circ f)_i(\bot, \bot, ..., \bot)$
since $f$ is monotone and $\omega$ is extensive

• $\omega \circ f$ is a monotone function $\omega(L) \rightarrow \omega(L)$
so the fixed-point exists

• 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:

\[
\text{fix} = \bigsqcup f^i(\bot, \bot, \ldots, \bot) \quad \text{fix}^\omega = \bigsqcup (\omega \circ f)^i(\bot, \bot, \ldots, \bot)
\]

then \( \text{fix} \sqsubseteq \text{fix}^\omega \)
• But we also have that

\[
\text{fix} \sqsubseteq f(\text{fix}^\omega) \sqsubseteq \text{fix}^\omega
\]

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

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

\[ B = \{-\infty, 0, 1, 7, \infty\} \]
Correctness of (repeated) narrowing

- \( f(fix\omega) \sqsubseteq \omega(f(fix\omega)) = (\omega \circ f)(fix\omega) = fix\omega \) since \( \omega \) 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

```plaintext
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 $\nabla : L \times L \rightarrow L$ that is extensive in both arguments and satisfies the following property:
  for all increasing chains $z_0 \subseteq z_1 \subseteq \ldots$,
  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
More powerful widening for interval analysis

Extrapolates unstable bounds to $B$:

\[
\begin{align*}
\bot \nabla y &= y \\
x \nabla \bot &= x \\
[a_1, b_1] \nabla [a_2, b_2] &= \\
&[\text{if } a_1 \leq a_2 \text{ then } a_1 \text{ else } \max\{i \in B \mid i \leq a_2\}, \\
&\text{if } b_2 \leq b_1 \text{ then } b_1 \text{ else } \min\{i \in B \mid b_2 \leq i\}] 
\end{align*}
\]

The $\nabla$ operator on $L$ is then defined pointwise down to individual intervals

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