Static Program Analysis

Lecture 7: Dataflow Analysis VI (Widening and Narrowing)

Summer Semester 2018

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-18/spa/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Interval Analysis

Interval Analysis

The goal of Interval Analysis is to determine, for each (interesting) program point, a safe interval for the values of the (interesting) program variables.

Interval analysis is actually a generalisation of constant propagation
($\approx$ interval analysis with one-element intervals)

Example (Interval Analysis)

```
var a[100]: int;
i := 0;
while i <= 42 do
  if i >= 0 ∧ i < 100 then      ⟸ redundant array bounds check
    a[i] := i
  end;
  i := i + 1;
end;
```

The Domain of Interval Analysis

- The domain $(Int, \subseteq)$ of intervals over $\mathbb{Z}$ is defined by

$$Int := \{[z_1, z_2] \mid z_1 \in \mathbb{Z} \cup \{-\infty\}, z_2 \in \mathbb{Z} \cup \{+\infty\}\}, z_1 \leq z_2\} \cup \{\emptyset\}$$

  where
  - $-\infty \leq z \leq +\infty$ (for all $z \in \mathbb{Z}$)
  - $\emptyset \subseteq J$ (for all $J \in Int$)
  - $[y_1, y_2] \subseteq [z_1, z_2]$ iff $z_1 \leq y_1$ and $y_2 \leq z_2$
- $(Int, \subseteq)$ is a complete lattice with (for every $\mathcal{I} \subseteq Int$)

$$\bigsqcup \mathcal{I} = \begin{cases} \emptyset & \text{if } \mathcal{I} = \emptyset \text{ or } \mathcal{I} = \{\emptyset\} \\ [Z_1, Z_2] & \text{otherwise} \end{cases}$$

  where

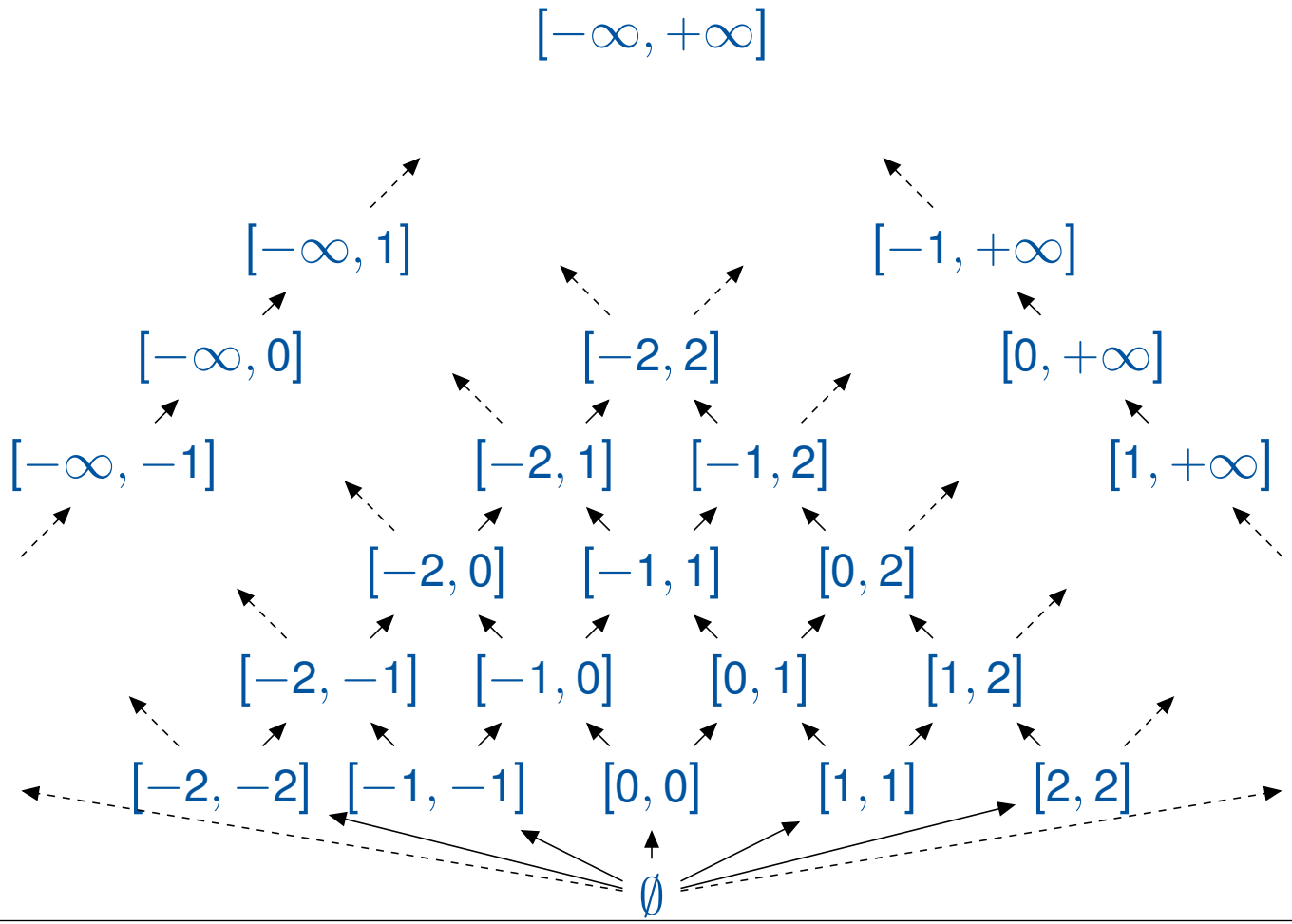
$$Z_1 := \bigsqcap_{\mathbb{Z} \cup \{-\infty\}} \{z_1 \mid [z_1, z_2] \in \mathcal{I}\}$$
$$Z_2 := \bigsqcup_{\mathbb{Z} \cup \{+\infty\}} \{z_2 \mid [z_1, z_2] \in \mathcal{I}\}$$

  (and thus $\bot = \emptyset$, $\top = [-\infty, +\infty]$)
- Clearly $(Int, \subseteq)$ has infinite ascending chains, such as

$$\emptyset \subseteq [1,1] \subseteq [1,2] \subseteq [1,3] \subseteq \ldots$$

Recap: Interval Analysis

The Complete Lattice of Interval Analysis



Formalising Interval Analysis I

The dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ is given by

- set of labels $Lab := Lab_c$
- extremal labels $E := \{\text{init}(c)\}$ (forward problem)
- flow relation $F := \text{flow}(c)$ (forward problem)
- complete lattice $(D, \sqsubseteq)$ where
  - $D := \{\delta \mid \delta : Var_c \to Int\}$
  - $\delta_1 \sqsubseteq \delta_2$ iff $\delta_1(x) \subseteq \delta_2(x)$ for every $x \in Var_c$
- $\iota := \top_D : Var_c \to Int : x \mapsto \top_{Int}$ (with $\top_{Int} = [-\infty, +\infty]$)
- φ: see next slide

Formalising Interval Analysis II

Transfer functions $\{\varphi_l \mid l \in Lab\}$ are defined by

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \texttt{skip} \text{ or } B^l \in BExp \\ \delta[x \mapsto val_\delta(a)] & \text{if } B^l = (x := a) \end{cases}$$

where

$$val_\delta(x) := \delta(x) \qquad val_\delta(a_1 + a_2) := val_\delta(a_1) \oplus val_\delta(a_2)$$
$$val_\delta(z) := [z, z] \qquad val_\delta(a_1 - a_2) := val_\delta(a_1) \ominus val_\delta(a_2)$$
$$val_\delta(a_1 * a_2) := val_\delta(a_1) \odot val_\delta(a_2)$$

with

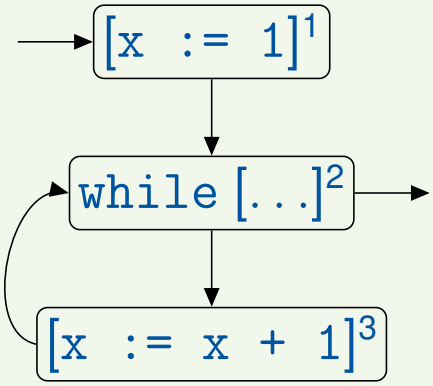
$$\emptyset \oplus J := J \oplus \emptyset := \emptyset \ominus J := \ldots := \emptyset$$
$$[y_1, y_2] \oplus [z_1, z_2] := [y_1 + z_1, y_2 + z_2]$$
$$[y_1, y_2] \ominus [z_1, z_2] := [y_1 - z_2, y_2 - z_1]$$
$$[y_1, y_2] \odot [z_1, z_2] := [\bigsqcap\{y_1 z_1, y_1 z_2, y_2 z_1, y_2 z_2\}, \bigsqcup\{y_1 z_1, y_1 z_2, y_2 z_1, y_2 z_2\}]$$

Remarks

- Important: soundness and optimality of abstract operations, e.g., $\oplus$:
  - soundness: $z_1 \in J_1, z_2 \in J_2 \implies z_1 + z_2 \in J_1 \oplus J_2$
  - optimality: $J_1 \oplus J_2$ as precise (i.e., small) as possible
  - cf. “Abstract Interpretation” (later)
- Possible refinement of DFA to take conditional blocks b^l ($b \in BExp$) into account
  - essentially: transfer functions for branches
  - b/$\neg b$ as “filter” operation
  - $\varphi_b(\delta)(x) = \text{least interval covering } \delta(x) \setminus \{z \in \mathbb{Z} \mid x = z \implies \neg b\}$
  - cf. “DFA with Conditional Branches” (later)

Interval Analysis without Widening

Example 7.1



Transfer functions (for $\delta(x) = J$):

$$\varphi_1(J) = [1, 1]$$
$$\varphi_2(J) = J$$
$$\varphi_3(\emptyset) = \emptyset$$
$$\varphi_3([a, b]) = [a + 1, b + 1]$$

Application of worklist algorithm without widening: does not terminate (on the board)

Widening Operators

Definition 7.2 (Widening operator)

Let $(D, \sqsubseteq)$ be a complete lattice. A mapping $\nabla : D \times D \to D$ is called widening operator if

- for every $d_1, d_2 \in D$,
$$d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$$
and
- for all ascending chains $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$, the ascending chain $d_0^\nabla \sqsubseteq d_1^\nabla \sqsubseteq \ldots$ eventually stabilises where
$$d_0^\nabla := d_0 \text{ and } d_{i+1}^\nabla := d_i^\nabla \nabla d_{i+1} \text{ for each } i \in \mathbb{N}$$

Remarks:

- The requirement $d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$ guarantees soundness of widening
- $(d_i^\nabla)_{i \in \mathbb{N}}$ is clearly an ascending chain as $d_{i+1}^\nabla = d_i^\nabla \nabla d_{i+1} \sqsupseteq d_i^\nabla \sqcup d_{i+1} \sqsupseteq d_i^\nabla$
- In contrast to $\sqcup$, ∇ does *not* have to be commutative, associative, monotonic, nor absorptive ($d \nabla d = d$)

Applying Widening to Interval Analysis

- A widening operator: $\nabla : \mathit{Int} \times \mathit{Int} \to \mathit{Int}$ with

$$\emptyset \nabla J := J \nabla \emptyset := J$$
$$[x_1, x_2] \nabla [y_1, y_2] := [z_1, z_2] \quad \text{where}$$
$$z_1 := \begin{cases} x_1 & \text{if } x_1 \leq y_1 \\ -\infty & \text{otherwise} \end{cases}$$
$$z_2 := \begin{cases} x_2 & \text{if } x_2 \geq y_2 \\ +\infty & \text{otherwise} \end{cases}$$

- Widening turns infinite ascending chain

$$J_0 = \emptyset \subseteq J_1 = [1,1] \subseteq J_2 = [1,2] \subseteq J_3 = [1,3] \subseteq \ldots$$

  into a finite one:

$$\begin{aligned} J_0^\nabla &= J_0 = \emptyset \\ J_1^\nabla &= J_0^\nabla \nabla J_1 = \emptyset \nabla [1,1] = [1,1] \\ J_2^\nabla &= J_1^\nabla \nabla J_2 = [1,1] \nabla [1,2] = [1,+\infty] \\ J_3^\nabla &= J_2^\nabla \nabla J_3 = [1,+\infty] \nabla [1,3] = [1,+\infty] \\ &\vdots \end{aligned}$$

- In fact, the maximal chain size arising with this operator is 4:

$$\emptyset \subseteq [3,7] \subseteq [3,+\infty] \subseteq [-\infty,+\infty]$$

Worklist Algorithm with Widening I

Goal: extend Algorithm 4.9 by widening to ensure termination

Algorithm 7.3 (Worklist algorithm with widening)

Input: *dataflow system* $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure:

$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := W \cdot (l, l')$; *% Initialise W*

  for $l \in Lab$ **do**

    if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$; *% Initialise* AI

  while $W \neq \varepsilon$ **do**

    $(l, l') := \mathbf{head}(W)$; $W := \mathbf{tail}(W)$; *% Next control-flow edge*

    if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*

      $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \nabla \varphi_l(\mathsf{AI}_l)$; *% Update analysis information*

      for $(l', l'') \in F$ **do**

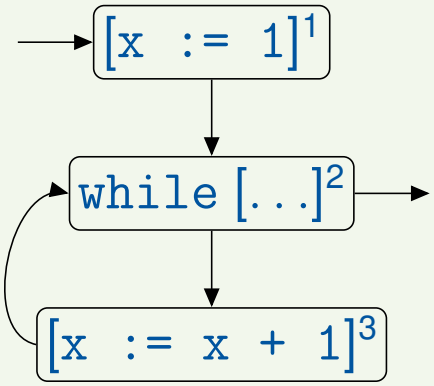
        if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$; *% Propagate modification*

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$, *denoted by* $\mathrm{fix}^{\nabla}(\Phi_S)$

Remark: due to widening, only $\mathrm{fix}^{\nabla}(\Phi_S) \sqsupseteq \mathrm{fix}(\Phi_S)$ is guaranteed (cf. Thm. 4.12)

Worklist Algorithm with Widening II

Example 7.4 (cf. Example 7.1)



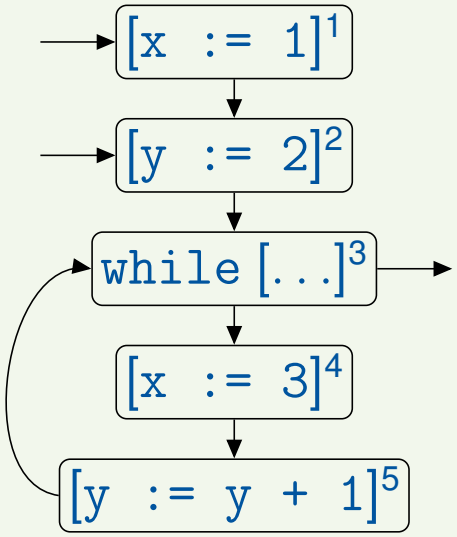
Transfer functions (for $\delta(x) = J$):

$$\varphi_1(J) = [1, 1]$$
$$\varphi_2(J) = J$$
$$\varphi_3(\emptyset) = \emptyset$$
$$\varphi_3([a, b]) = [a + 1, b + 1]$$

Application of worklist algorithm with widening:
terminates with expected result for AI_2 ($[1, +\infty]$) (on the board)

Another Widening Example

Example 7.5



Transfer functions (for $\delta = (J_x, J_y)$):

$$\begin{aligned}
\varphi_1(J_x, J_y) &= ([1,1], J_y) \\
\varphi_2(J_x, J_y) &= (J_x, [2,2]) \\
\varphi_3(J_x, J_y) &= (J_x, J_y) \\
\varphi_4(J_x, J_y) &= ([3,3], J_y) \\
\varphi_5(J_x, \emptyset) &= (J_x, \emptyset) \\
\varphi_5(J_x, [a,b]) &= (J_x, [a+1, b+1])
\end{aligned}$$

Application of worklist algorithm

1. without widening (omitted): diverges (for y) with expected result for x: $AI_3(x) = [1,3]$
2. with widening (on the board): terminates with unexpected result for x: $AI_3(x) = [1, +\infty]$

Idea of Narrowing

- **Observation:** widening can "shoot above the target", i.e., lead to unnecessarily imprecise results
- **Solution:** improvement by iterating again from the result obtained by widening (i.e., from $\text{fix}^{\nabla}(\Phi_S)$)

$$\Longrightarrow \text{ compute } \Phi_S^k(\text{fix}^{\nabla}(\Phi_S)) \text{ for } k = 1, 2, \ldots$$

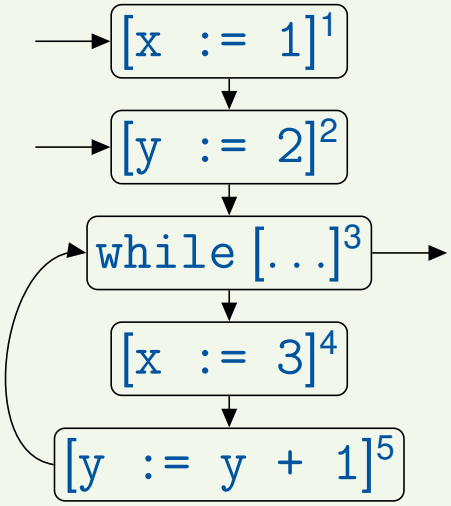
- **Soundness:** $\text{fix}^{\nabla}(\Phi_S) \sqsupseteq \text{fix}(\Phi_S)$ (cf. Algorithm 7.3)

$$\Longrightarrow \Phi_S^k(\text{fix}^{\nabla}(\Phi_S)) \sqsupseteq \Phi_S^k(\text{fix}(\Phi_S)) = \text{fix}(\Phi_S)$$

(since Φ_S and thus Φ_S^k monotonic)

Narrowing Example

Example 7.6 (cf. Example 7.5)



Transfer functions (for $\delta = (J_x, J_y)$):

$$\begin{aligned}
\varphi_1(J_x, J_y) &= ([1,1], J_y) \\
\varphi_2(J_x, J_y) &= (J_x, [2,2]) \\
\varphi_3(J_x, J_y) &= (J_x, J_y) \\
\varphi_4(J_x, J_y) &= ([3,3], J_y) \\
\varphi_5(J_x, \emptyset) &= (J_x, \emptyset) \\
\varphi_5(J_x, [a,b]) &= (J_x, [a+1, b+1])
\end{aligned}$$

Narrowing	AI_1	AI_2	AI_3	AI_4	AI_5
$\text{fix}^{\nabla}(\Phi_S)$	$(\top, \top)$	$([1,1], \top)$	$([1,+\infty], [2,+\infty])$	$([1,+\infty], [2,+\infty])$	$([3,3], [2,+\infty])$
$\Phi_S^1(\text{fix}^{\nabla}(\Phi_S))$	$(\top, \top)$	$([1,1], \top)$	$([1,3], [2,+\infty])$	$([1,+\infty], [2,+\infty])$	$([3,3], [2,+\infty])$
$\Phi_S^2(\text{fix}^{\nabla}(\Phi_S))$	$(\top, \top)$	$([1,1], \top)$	$([1,3], [2,+\infty])$	$([1,3], [2,+\infty])$	$([3,3], [2,+\infty])$
$\Phi_S^3(\text{fix}^{\nabla}(\Phi_S))$	$(\top, \top)$	$([1,1], \top)$	$([1,3], [2,+\infty])$	$([1,3], [2,+\infty])$	$([3,3], [2,+\infty])$

Narrowing in Practice

- **Problem:** narrowing may not terminate (due to infinite descending chains)
- **But:** possible to stop after each step without losing soundness ($\Phi_S^k(\text{fix}^\nabla(\Phi_S)) \sqsupseteq \text{fix}(\Phi_S)$)
- **In practice:** termination often ensured by using narrowing operators
  ($\approx$ counterpart of widening operator; definition omitted)