Static Program Analysis

Lecture 9: Dataflow Analysis VIII (Narrowing & DFA with Conditional Branches)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/spa11/

Summer Semester 2011

Outline

Widening Operators

Definition (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 stabilizes where
$$d_0^\nabla := d_0 \text{ and } d_{i+1}^\nabla := d_i^\nabla \nabla d_{i+1} \text{ for } i \in \mathbb{N}$$

Remarks:

- $(d_i^\nabla)_{i \in \mathbb{N}}$ 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 $(d \nabla d = d)$
- The requirement $d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$ guarantees soundness of widening

Applying Widening to Interval Analysis

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

$$\emptyset \nabla I := I \nabla \emptyset := I$$

$$[x_1, x_2] \nabla [y_1, y_2] := [z_1, z_2] \quad \text{where}$$

$$z_1 := \begin{cases} x_1 & \text{if } x_1 \leq y_1 \\ -\infty & \text{otherwise} \end{cases}$$

$$z_2 := \begin{cases} x_2 & \text{if } y_2 \leq x_2 \\ +\infty & \text{otherwise} \end{cases}$$

- Widening turns infinite ascending chain

$$I_0 = \emptyset \subseteq I_1 = [1,1] \subseteq I_2 = [1,2] \subseteq I_3 = [1,3] \subseteq \ldots$$

into a finite one:

$$\begin{aligned} I_0^\nabla &= I_0 = \emptyset \\ I_1^\nabla &= I_0^\nabla \nabla I_1 = \emptyset \nabla [1,1] = [1,1] \\ I_2^\nabla &= I_1^\nabla \nabla I_2 = [1,1] \nabla [1,2] = [1,+\infty] \\ I_3^\nabla &= I_2^\nabla \nabla I_3 = [1,+\infty] \nabla [1,3] = [1,+\infty] \end{aligned}$$

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

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

Worklist Algorithm with Widening

Goal: extend Algorithm 5.2 by widening to ensure termination

Algorithm (Worklist algorithm with widening)

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

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

Procedure:

$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := W \cdot (l, l')$; *% Initialize W*
for $l \in L$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') := \mathbf{head}(W)$; $W := \mathbf{tail}(W)$;
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \nabla \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do**
      if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$;

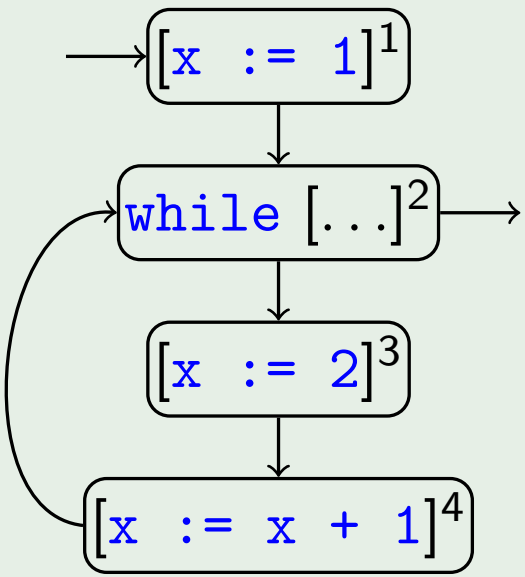
Output: $\{\mathsf{AI}_l \mid l \in L\}$, *denoted by* $\mathrm{fix}^{\nabla}(\Phi_S)$

Remark: due to widening, only $\mathrm{fix}(\Phi_S) \sqsubseteq \mathrm{fix}^{\nabla}(\Phi_S)$ is guaranteed (cf. Thm. 5.4)

Outline

Another Widening Example

Example 9.1



Transfer functions (for $\delta(\mathtt{x}) = I$):

$$\begin{aligned} \varphi_1(I) &= [1,1] \\ \varphi_2(I) &= I \\ \varphi_3(I) &= [2,2] \\ \varphi_4(\emptyset) &= \emptyset \\ \varphi_4([x_1,x_2]) &= [x_1+1, x_2+1] \end{aligned}$$

Application of worklist algorithm

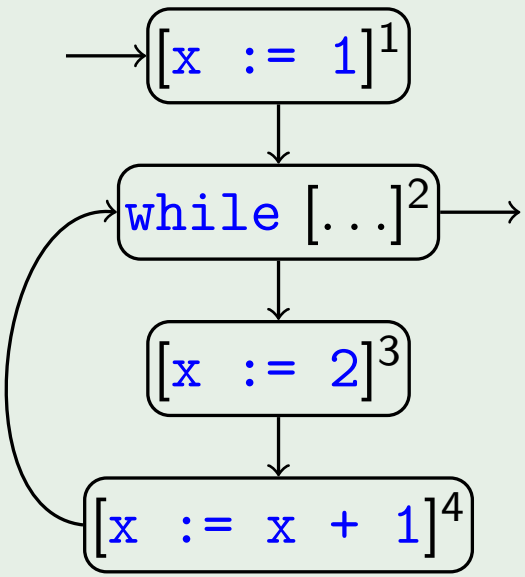
1. without widening (omitted):
   terminates with expected result for AI_2 ($[1,3]$)
2. with widening (on the board):
   terminates with unexpected result for AI_2 ($[1,+\infty]$)

Idea of Narrowing

- **Observation:** widening can lead to unnecessarily imprecise results
- **Solution:** improvement by iterating again from the result obtained by widening (i.e., from $\text{fix}^{\nabla}(\Phi_S)$)
  $\Longrightarrow$ compute $\Phi_S^k(\text{fix}^{\nabla}(\Phi_S))$ for $k = 1, 2, \ldots$
- **Soundness:** $\text{fix}(\Phi_S) \sqsubseteq \text{fix}^{\nabla}(\Phi_S)$
  $\Longrightarrow$ $\text{fix}(\Phi_S) = \Phi_S^k(\text{fix}(\Phi_S)) \sqsubseteq \Phi_S^k(\text{fix}^{\nabla}(\Phi_S))$
  (since Φ_S and thus Φ_S^k monotonic)

Narrowing Example

Example 9.2 (cf. Example 9.1)



Transfer functions (for $\delta(\mathtt{x}) = I$):

$$\begin{aligned} \varphi_1(I) &= [1,1] \\ \varphi_2(I) &= I \\ \varphi_3(I) &= [2,2] \\ \varphi_4(\emptyset) &= \emptyset \\ \varphi_4([x_1,x_2]) &= [x_1+1, x_2+1] \end{aligned}$$

Narrowing:

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

Narrowing in Practice

- **Problem:** narrowing may not terminate
  (due to infinite descending chains)
- **But:** possible to stop after every step without losing soundness
- **In practice:** termination often ensured by using narrowing operators
  ($\approx$ counterpart of widening operator; definition omitted)

Outline

Taking Conditional Branches into Account I

- **So far:** values of conditions have been ignored in analysis
- Essentially: `if` and `while` statements treated as nondeterministic choice between the two branches

Example 9.3

```
y := 0;
z := 0;
while [x > 0]' do
  if y < 17 then
    y := y + 1
  z := z + x;
  x := x - 1;
```

- Interval analysis (with widening) yields for l:

$$\begin{aligned} x &\in [-\infty, +\infty] \\ y &\in [0, +\infty] \\ z &\in [-\infty, +\infty] \end{aligned}$$

- Too pessimistic! In fact,

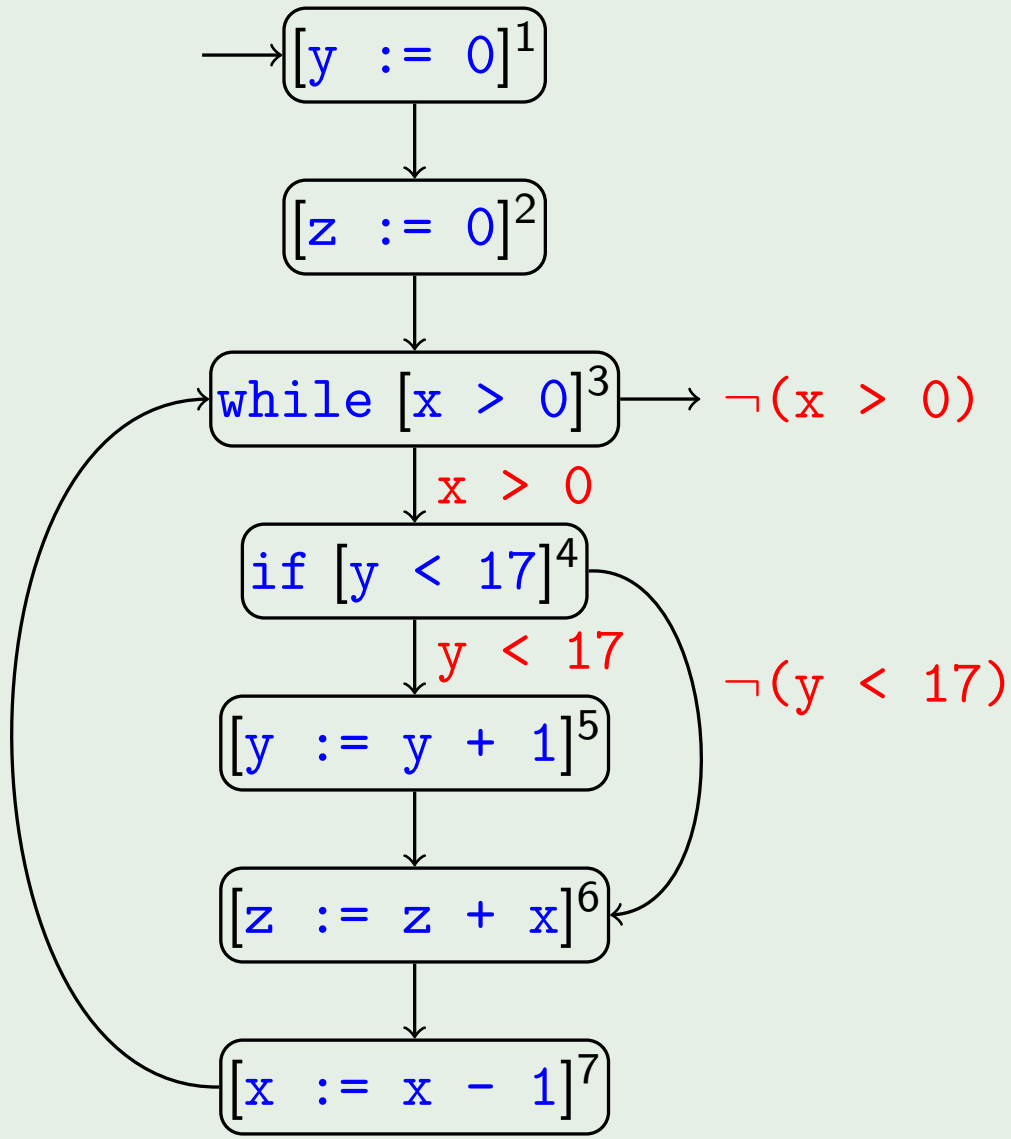
$$\begin{aligned} x &\in [-\infty, +\infty] \\ y &\in [0, 17] \\ z &\in [0, +\infty] \end{aligned}$$

Taking Conditional Branches into Account II

- **Solution:** introduce transfer functions for branches
- **First approach:** attach (negated) conditions as labels to control flow edges
  - advantage: no language modification required
  - disadvantage: entails extension of DFA framework
  - will not be considered here
- **Second approach:** encode conditions as assertions (statements)
  - advantage: DFA framework can be reused
  - disadvantage: entails extension of WHILE language
  - the way we will follow

First Approach: Conditions as Edge Labels

Example 9.4 (cf. Example 9.3)



Second Approach: Conditions as Assertions

Example 9.5 (cf. Example 9.3)

```
y := 0;
z := 0;
while x > 0 do
  assert x > 0;
  if y < 17 then
    assert y < 17;
    y := y + 1;
  z := z + x;
  x := x - 1;
assert ¬(x > 0);
```

Extending the Syntax of WHILE Programs

Definition 9.6 (Labeled WHILE programs with assertions)

The syntax of labeled WHILE programs with assertions is defined by the following context-free grammar:

$$
\begin{aligned}
a &::= z \mid x \mid a_1 \texttt{+} a_2 \mid a_1 \texttt{-} a_2 \mid a_1 \texttt{*} a_2 \in \mathit{AExp} \\
b &::= t \mid a_1 \texttt{=} a_2 \mid a_1 \texttt{>} a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in \mathit{BExp} \\
c &::= [\texttt{skip}]^l \mid [x \texttt{ := } a]^l \mid c_1 \texttt{;} c_2 \mid \\
&\quad\ \texttt{if } [b]^l \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } [b]^l \texttt{ do } c \mid [\texttt{assert } b]^l \in \mathit{Cmd}
\end{aligned}
$$

To be done:

- Definition of transfer functions for `assert` blocks (depending on analysis problem)
- Idea: assertions as filters that let only "valid" information pass

Outline

Constant Propagation Analysis with Assertions I

So far:

- Complete lattice $(D, \sqsubseteq)$ where
  - $D := \{\delta \mid \delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}\}$
    - $\delta(x) = z \in \mathbb{Z}$: x has constant value z
    - $\delta(x) = \bot$: x undefined
    - $\delta(x) = \top$: x overdefined (i.e., different possible values)
  - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\bot \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)
- Transfer functions $\{\varphi_l \mid l \in L\}$ defined by

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

where

$$val_\delta(x) := \delta(x) \qquad val_\delta(z) := z$$

$$val_\delta(a_1 \; op \; a_2) := \begin{cases} z_1 \; op \; z_2 & \text{if } z_1, z_2 \in \mathbb{Z} \\ \bot & \text{if } z_1 = \bot \text{ or } z_2 = \bot \\ \top & \text{otherwise} \end{cases}$$

for $z_1 := val_\delta(a_1)$ and $z_2 := val_\delta(a_2)$

Constant Propagation Analysis with Assertions II

Additionally for $B^l = (\texttt{assert}\ b)$, $\delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}$ and $x \in Var_c$:

$$\varphi_l(\delta)(x) := \begin{cases} \bot & \text{if } \nexists \sigma \in \Sigma_\delta : val_\sigma(b) = \text{true} \\ z & \text{if } \forall \sigma \in \Sigma_\delta : val_\sigma(b) = \text{true} \implies \sigma(x) = z \\ \top & \text{otherwise} \end{cases}$$

where

- the set of δ-assignments is given by

$$\Sigma_\delta := \left\{ \sigma : Var_c \to \mathbb{Z} \;\middle|\; \forall y \in Var_c : \sigma(y) \in \begin{cases} \emptyset & \text{if } \delta(y) = \bot \\ \{z\} & \text{if } \delta(y) = z \\ \mathbb{Z} & \text{if } \delta(y) = \top \end{cases} \right\}$$

  (and thus $\Sigma_\delta = \emptyset$ iff $\delta(y) = \bot$ for some $y \in Var_c$)

- the evaluation function $val_\sigma : BExp \to \mathbb{B}$ is defined by

$$val_\sigma(t) := t$$
$$val_\sigma(a_1 = a_2) := (val_\sigma(a_1) = val_\sigma(a_2))$$
$$val_\sigma(\neg b) := \begin{cases} \text{true} & \text{if } val_\sigma(b) = \text{false} \\ \text{false} & \text{otherwise} \end{cases}$$
$$val_\sigma(b_1 \wedge b_2) := \begin{cases} \text{true} & \text{if } val_\sigma(b_1) = val_\sigma(b_2) = \text{true} \\ \text{false} & \text{otherwise} \end{cases}$$

  etc.

Constant Propagation Analysis with Assertions III

Example 9.7

1. $Var_c = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$, $\delta = (\underbrace{\bot}_{\mathtt{x}}, \underbrace{1}_{\mathtt{y}}, \underbrace{\top}_{\mathtt{z}})$

   $\Longrightarrow \Sigma_\delta = \emptyset$

   $\Longrightarrow \varphi_{\mathtt{assert}\ b}(\delta) = (\bot, \bot, \bot)$ for every $b \in BExp$

2. $Var_c = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$, $\delta = (\underbrace{1}_{\mathtt{x}}, \underbrace{2}_{\mathtt{y}}, \underbrace{\top}_{\mathtt{z}})$

   $\Longrightarrow \Sigma_\delta = \{(1, 2, z) \mid z \in \mathbb{Z}\}$

   $\Longrightarrow$
   $$\varphi_{\mathtt{assert\ x=y}}(\delta) = (\bot, \bot, \bot)$$
   $$\varphi_{\mathtt{assert\ y=z}}(\delta) = (1, 2, 2)$$
   $$\varphi_{\mathtt{assert\ y<z}}(\delta) = (1, 2, \top)$$
   $$\varphi_{\mathtt{assert\ x<=z \wedge y>z}}(\delta) = (1, 2, 1)$$

Remarks:

- Note that for $B^l = (\mathtt{assert}\ b)$ and $\delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}$, $\varphi_l(\delta) \sqsubseteq \delta$ and hence $\Sigma_{\varphi_l(\delta)} \subseteq \Sigma_\delta$ ("filter")
- If $CP_l(x) = \bot$ for some $l \in L_c$ and $x \in Var_c$, then l is unreachable

Outline

Interval Analysis with Assertions I

So far:

- The domain $(\mathit{Int}, \subseteq)$ of intervals over $\mathbb{Z}$ is defined by
$$\mathit{Int} := \{[z_1, z_2] \mid z_1 \in \mathbb{Z} \cup \{-\infty\}, z_2 \in \mathbb{Z} \cup \{+\infty\}, z_1 \leq z_2\} \cup \{\emptyset\}$$
where
  - $-\infty \leq z$, $z \leq +\infty$, and $-\infty \leq +\infty$ (for all $z \in \mathbb{Z}$)
  - $\emptyset \subseteq I$ (for all $I \in \mathit{Int}$)
  - $[y_1, y_2] \subseteq [z_1, z_2]$ iff $z_1 \leq y_1$ and $y_2 \leq z_2$
- Transfer functions $\{\varphi_l \mid l \in L\}$ are defined by
$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \texttt{skip} \text{ or } B^l \in \mathit{BExp} \\ \delta[x \mapsto \mathit{val}_\delta(a)] & \text{if } B^l = (x \texttt{ := } a) \end{cases}$$
where
$$\mathit{val}_\delta(x) := \delta(x) \qquad \mathit{val}_\delta(a_1 \texttt{+} a_2) := \mathit{val}_\delta(a_1) \oplus \mathit{val}_\delta(a_2)$$
$$\mathit{val}_\delta(z) := [z, z] \qquad \mathit{val}_\delta(a_1 \texttt{-} a_2) := \mathit{val}_\delta(a_1) \ominus \mathit{val}_\delta(a_2)$$
$$\mathit{val}_\delta(a_1 \texttt{*} a_2) := \mathit{val}_\delta(a_1) \odot \mathit{val}_\delta(a_2)$$
with
$$\emptyset \oplus I := I \oplus \emptyset := \emptyset \ominus I := \ldots := \emptyset$$
$$[y_1, y_2] \oplus [z_1, z_2] := [y_1 + z_1, y_2 + z_2]$$
$$[y_1, y_2] \ominus [z_1, z_2] := [y_1 - z_2, y_2 - z_1]$$
$$[y_1, y_2] \odot [z_1, z_2] := [\min_{y \in [y_1, y_2], z \in [z_1, z_2]} y \cdot z, \max_{y \in [y_1, y_2], z \in [z_1, z_2]} y \cdot z]$$

Interval Analysis with Assertions II

Additionally for $B^l = (\texttt{assert}\ b)$, $\delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}$ and $x \in Var_c$:

$$\varphi_l(\delta)(x) := \begin{cases} \emptyset & \text{if } \nexists \sigma \in \Sigma_\delta : val_\sigma(b) = \text{true} \\ \left[\bigsqcap_{\mathbb{Z}\cup\{-\infty\}} Z, \bigsqcup_{\mathbb{Z}\cup\{+\infty\}} Z\right] & \text{otherwise} \end{cases}$$

where

- $Z := \{\sigma(x) \mid \sigma \in \Sigma_\delta, val_\sigma(b) = \text{true}\}$
- $\Sigma_\delta := \{\sigma : Var_c \to \mathbb{Z} \mid \forall y \in Var_c : \sigma(y) \in \delta(y)\}$
  (and thus $\Sigma_\delta = \emptyset$ iff $\delta(y) = \emptyset$ for some $y \in Var_c$)
- $val_\sigma : BExp \to \mathbb{B}$ as before

Interval Analysis with Assertions III

Example 9.8

$Var_c = \{\texttt{x}, \texttt{y}\}$, $\delta = (\underbrace{[-\infty, 2]}_{\texttt{x}}, \underbrace{[0, +\infty]}_{\texttt{y}})$

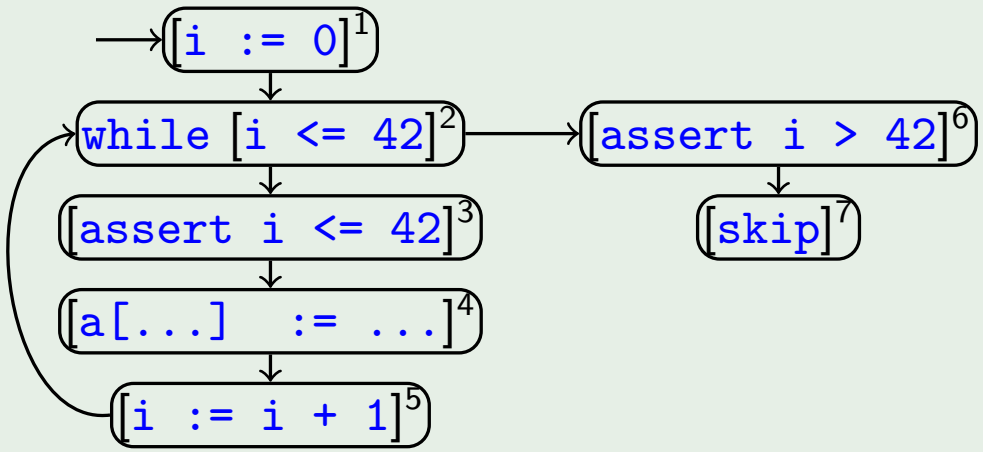
$$
\begin{aligned}
\Longrightarrow \quad & \varphi_{\texttt{assert x>0}}(\delta) = ([1, 2], [0, +\infty]) \\
& \varphi_{\texttt{assert x=y}}(\delta) = ([0, 2], [0, 2]) \\
& \varphi_{\texttt{assert x>y}}(\delta) = ([1, 2], [0, 1]) \\
& \varphi_{\texttt{assert x<y}}(\delta) = ([-\infty, 2], [0, +\infty])
\end{aligned}
$$

Remarks:

- Again for $B^l = (\texttt{assert}\ b)$ and $\delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}$, $\varphi_l(\delta) \sqsubseteq \delta$ and hence $\Sigma_{\varphi_l(\delta)} \subseteq \Sigma_\delta$ ("filter")
- Again if $\mathsf{AI}_l(x) = \emptyset$ for some $l \in L_c$ and $x \in Var_c$, then l is unreachable

Interval Analysis with Assertions IV

Example 9.9 (Interval analysis for i; cf. Example 7.7)



$$
\begin{aligned}
\varphi_1(I) &= [0,0] \\
\varphi_2(I) &= I \\
\varphi_3(I) &= I \cap [-\infty, 42] \\
\varphi_4(I) &= I \\
\varphi_5(\emptyset) &= \emptyset \\
\varphi_5([i_1, i_2]) &= [i_1 + 1, i_2 + 1] \\
\varphi_6(I) &= I \cap [43, +\infty]
\end{aligned}
$$

W	AI_1	AI_2	AI_3	AI_4	AI_5	AI_6	AI_7
$12, 23, 34, 45, 52, 26, 67$	$[-\infty, +\infty]$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$23, 34, 45, 52, 26, 67$	$[-\infty, +\infty]$	$[0, 0]$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$34, 45, 52, 26, 67$	$[-\infty, +\infty]$	$[0, 0]$	$[0, 0]$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$45, 52, 26, 67$	$[-\infty, +\infty]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$\emptyset$	$\emptyset$	$\emptyset$
$52, 26, 67$	$[-\infty, +\infty]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$\emptyset$	$\emptyset$
$23, 26, 67$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$\emptyset$	$\emptyset$
$34, 26, 67$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 0]$	$[0, 0]$	$\emptyset$	$\emptyset$
$45, 26, 67$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 0]$	$\emptyset$	$\emptyset$
$52, 26, 67$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$\emptyset$	$\emptyset$
$26, 67$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$\emptyset$	$\emptyset$
67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$\emptyset$
ε	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$