



Static Program Analysis

**Lecture 9: Dataflow Analysis VIII
(Conditional Interval Analysis & Java Virtual Machine)**

Winter Semester 2016/17

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ws-1617/spa/>

Recap: Taking Conditional Branches into Account

Outline of Lecture 9

Recap: Taking Conditional Branches into Account

Interval Analysis with Assertions

The Java Virtual Machine

The Java Bytecode Verifier

The Type-Level Abstract Interpreter

Recap: Taking Conditional Branches into Account

Extending the Syntax of WHILE Programs

Definition (Labelled WHILE programs with assertions)

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

$$a ::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in AExp$$
$$b ::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp$$
$$c ::= [\text{skip}]' \mid [x := a]' \mid c_1 ; c_2 \mid$$
$$\text{if } [b]' \text{ then } c_1 \text{ else } c_2 \text{ end} \mid \text{while } [b]' \text{ do } c \text{ end} \mid [\text{assert } b]' \in Cmd$$

To be done:

- Definition of **transfer functions** for **assert** blocks (depending on analysis problem)
- Idea: assertions as **filters** that let only “compatible” information pass

Interval Analysis with Assertions

Outline of Lecture 9

Recap: Taking Conditional Branches into Account

Interval Analysis with Assertions

The Java Virtual Machine

The Java Bytecode Verifier

The Type-Level Abstract Interpreter

Interval Analysis with Assertions

Original Interval Analysis

So far:

- 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

- $-\infty \leq z$ and $z \leq +\infty$ (for all $z \in \mathbb{Z}$)
- $\emptyset \subseteq J$ (for all $J \in Int$)
- $[y_1, y_2] \subseteq [z_1, z_2]$ iff $y_1 \geq z_1$ and $y_2 \leq z_2$
- **Transfer functions** $\{\varphi_l \mid l \in Lab\}$ are defined by

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

where

$$\begin{array}{ll} val_\delta(x) := \delta(x) & val_\delta(a_1 + a_2) := val_\delta(a_1) \oplus val_\delta(a_2) \\ val_\delta(z) := [z, z] & val_\delta(a_1 - a_2) := val_\delta(a_1) \ominus val_\delta(a_2) \\ & val_\delta(a_1 * a_2) := val_\delta(a_1) \odot val_\delta(a_2) \end{array}$$

Interval Analysis with Assertions

Transfer Functions of Assertions I

Additionally for $B^l = (\text{assert } b)$, $\delta : \text{Var}_c \rightarrow \text{Int}$ and $x \in \text{Var}_c$:

$$\varphi_l(\delta)(x) := \begin{cases} \emptyset & \text{if } Z = \emptyset \\ \left[\prod_{\mathbb{Z} \cup \{-\infty\}} Z, \bigsqcup_{\mathbb{Z} \cup \{+\infty\}} Z \right] & \text{otherwise} \end{cases}$$

where

- $Z := \{\sigma(x) \mid \sigma \in \Sigma_\delta, \text{val}_\sigma(b) = \text{true}\}$
- $\Sigma_\delta := \{\sigma : \text{Var}_c \rightarrow \mathbb{Z} \mid \forall y \in \text{Var}_c : \sigma(y) \in \delta(y)\}$
(and thus $\Sigma_\delta = \emptyset$ iff $\delta(y) = \emptyset$ for some $y \in \text{Var}_c$)
- $\text{val}_\sigma : \text{BExp} \rightarrow \mathbb{B}$: see Slide 8.25

Interval Analysis with Assertions

Transfer Functions of Assertions II

Example 9.1

$$\text{Var}_c = \{x, y\}, \delta = (\underbrace{[-\infty, 2]}_x, \underbrace{[0, +\infty]}_y)$$

$$\implies \varphi_{\text{assert } x > 0}(\delta) = ([1, 2], [0, +\infty])$$

$$\varphi_{\text{assert } x=y}(\delta) = ([0, 2], [0, 2])$$

$$\varphi_{\text{assert } x > y}(\delta) = ([1, 2], [0, 1])$$

$$\varphi_{\text{assert } x < y}(\delta) = ([-\infty, 2], [0, +\infty])$$

Interval Analysis with Assertions

Transfer Functions of Assertions II

Example 9.1

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

$$\implies \varphi_{\text{assert } x > 0}(\delta) = ([1, 2], [0, +\infty])$$

$$\varphi_{\text{assert } x=y}(\delta) = ([0, 2], [0, 2])$$

$$\varphi_{\text{assert } x > y}(\delta) = ([1, 2], [0, 1])$$

$$\varphi_{\text{assert } x < y}(\delta) = ([-\infty, 2], [0, +\infty])$$

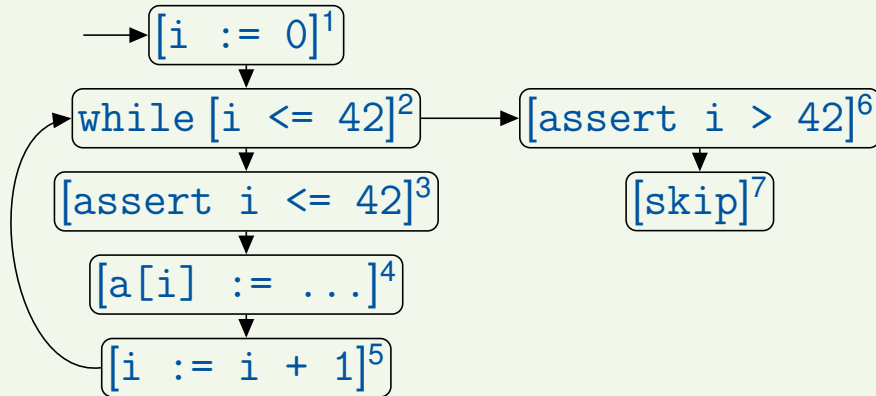
Remarks:

- Again for $B^l = (\text{assert } b)$ and $\delta : Var_c \rightarrow Int$, $\varphi_l(\delta) \sqsubseteq \delta$ and hence $\Sigma_{\varphi_l(\delta)} \subseteq \Sigma_\delta$ (“**filter**”)
- Again if $AI_l(x) = \emptyset$ for some $l \in Lab_c$ and $x \in Var_c$, then l is **unreachable** (and $AI_l(y) = \emptyset$ for all $y \in Var_c$)

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

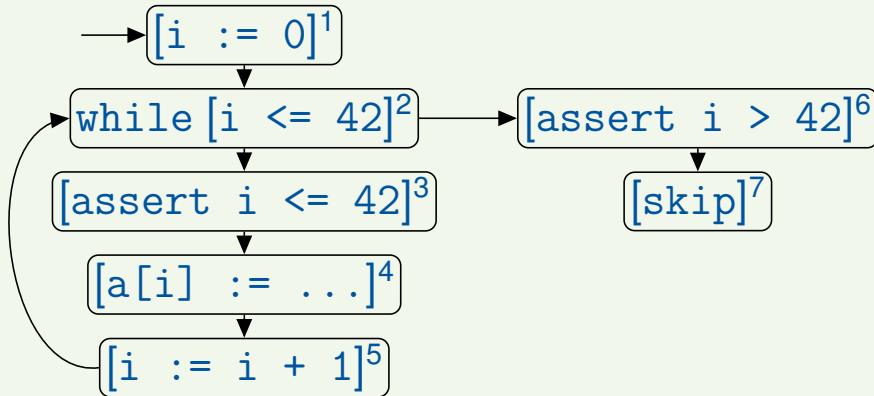
$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

$$\varphi_6(J) = J \cap [43, +\infty]$$

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

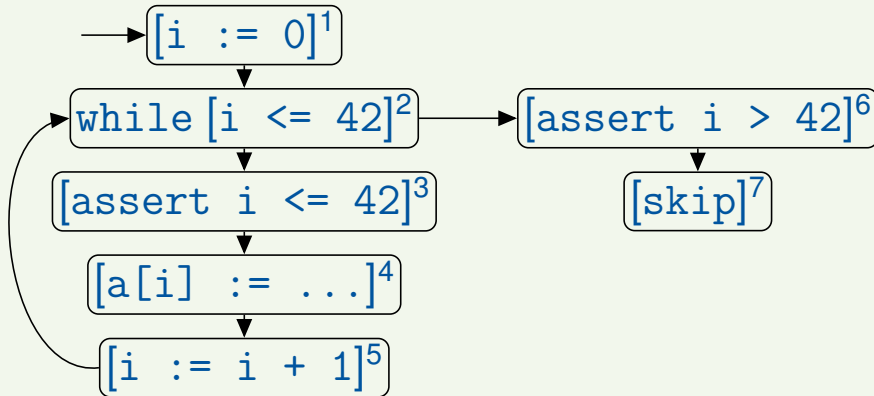
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

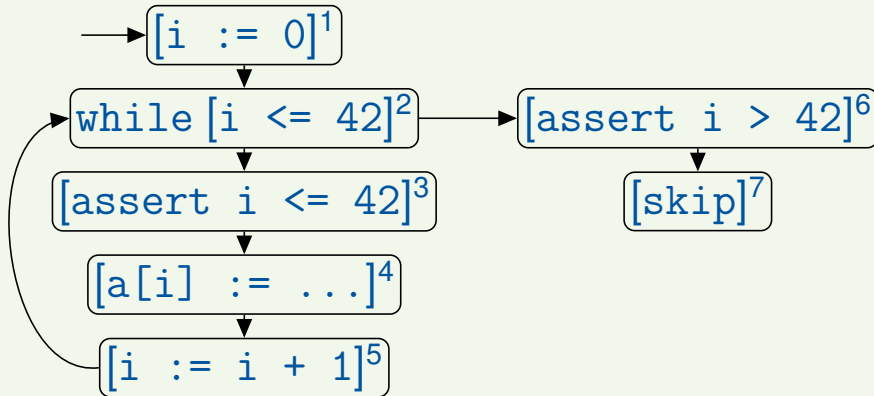
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

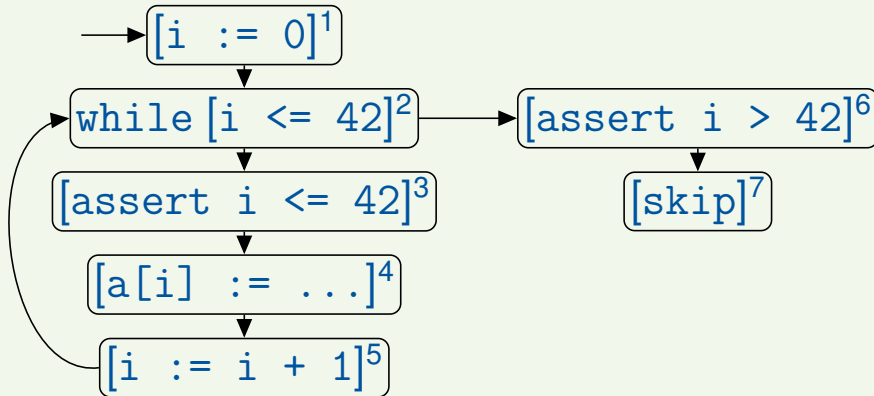
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

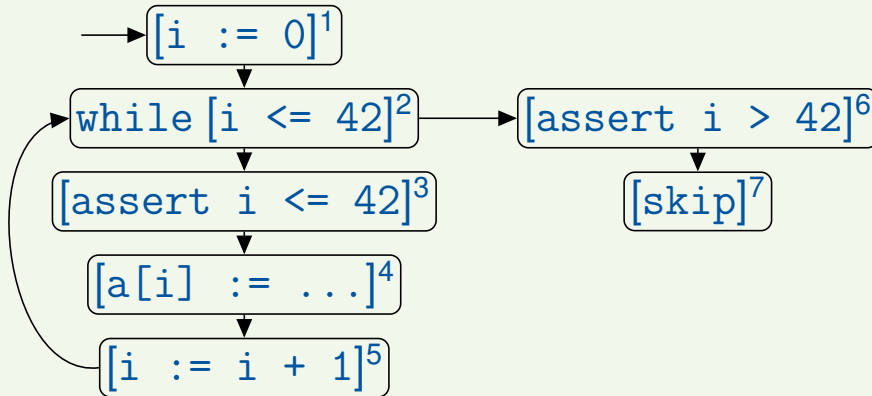
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

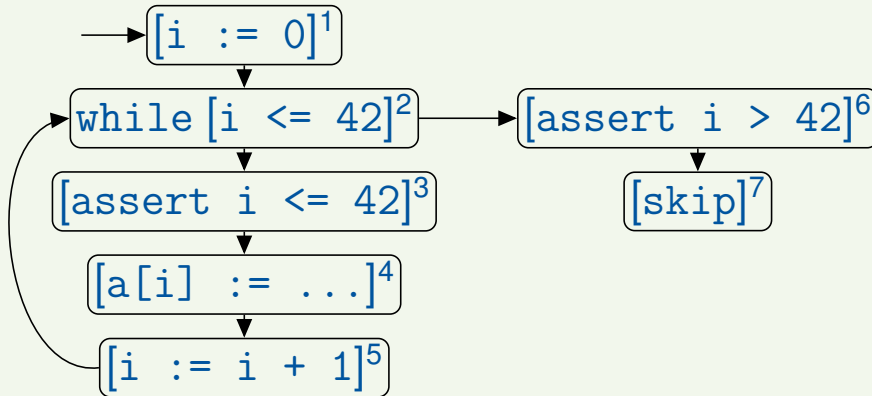
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

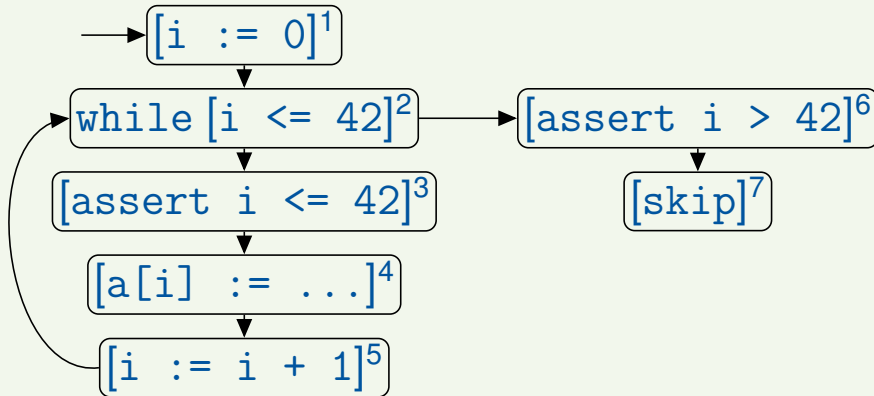
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

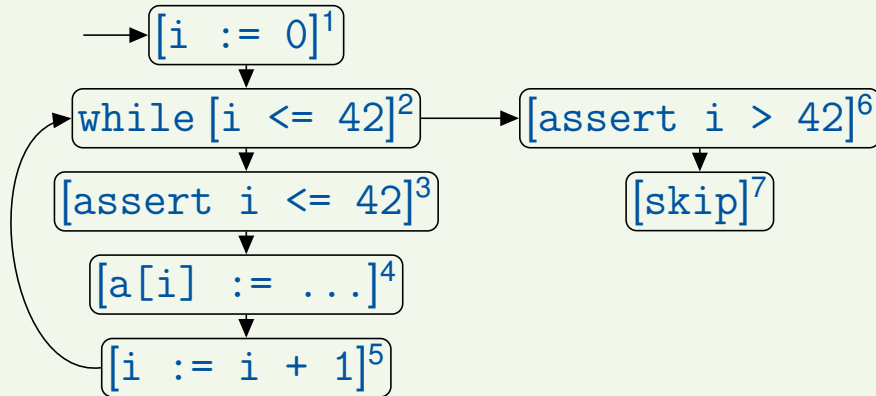
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	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

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

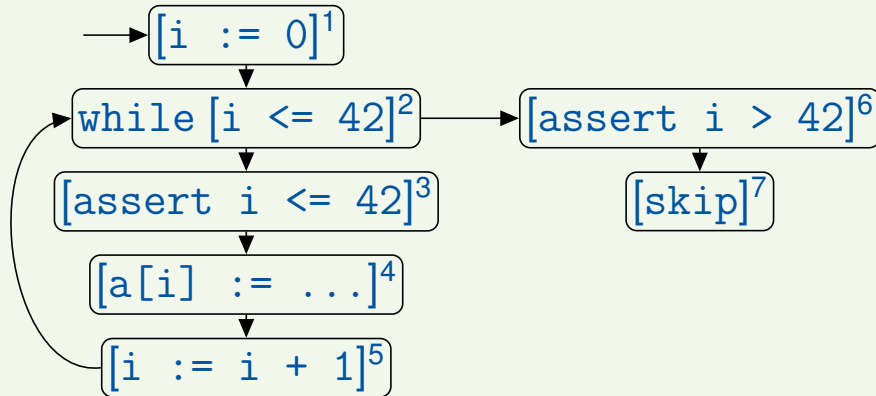
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	AI ₁	AI ₂	AI ₃	AI ₄	AI ₅	AI ₆	AI ₇
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	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 0]$	\emptyset	\emptyset

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

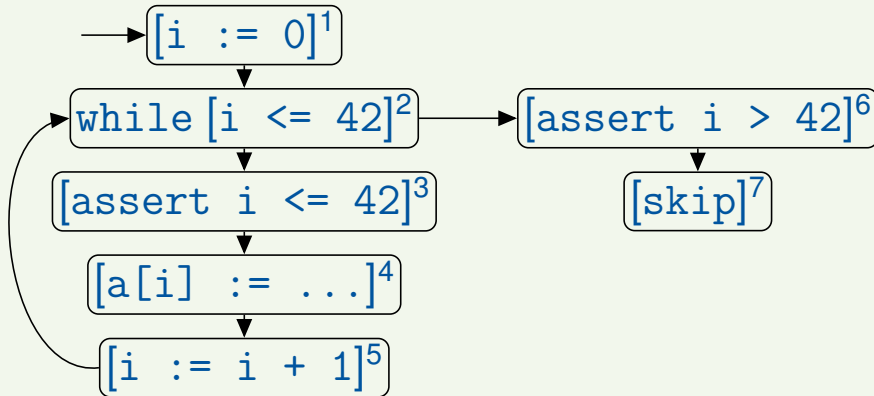
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	AI ₁	AI ₂	AI ₃	AI ₄	AI ₅	AI ₆	AI ₇
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	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 0]$	\emptyset	\emptyset
52, 26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

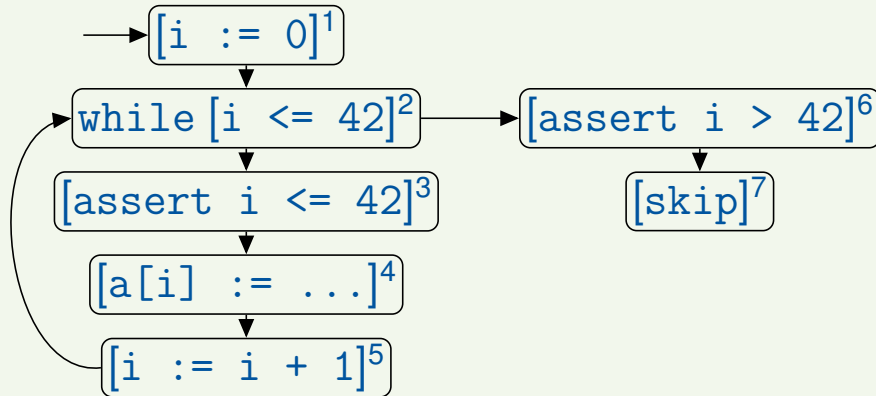
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	AI ₁	AI ₂	AI ₃	AI ₄	AI ₅	AI ₆	AI ₇
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	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 0]$	\emptyset	\emptyset
52, 26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset
26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

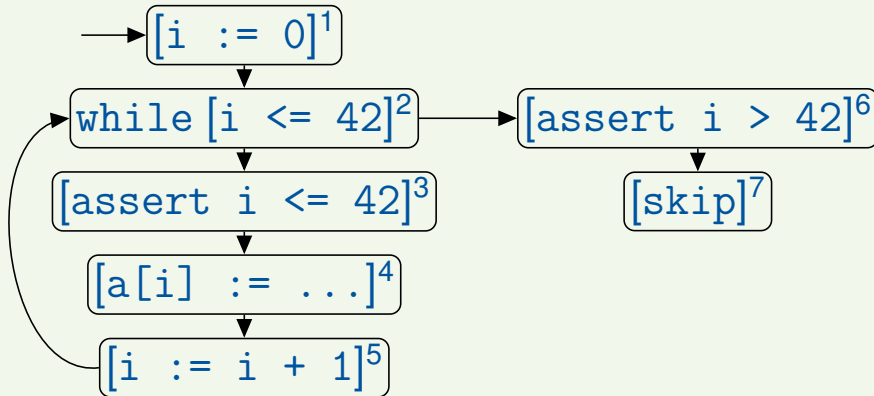
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	AI ₁	AI ₂	AI ₃	AI ₄	AI ₅	AI ₆	AI ₇
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	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 0]$	\emptyset	\emptyset
52, 26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset
26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset
67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset

Interval Analysis with Assertions

An Example

Example 9.2 (Interval analysis for array index; cf. Example 7.2)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

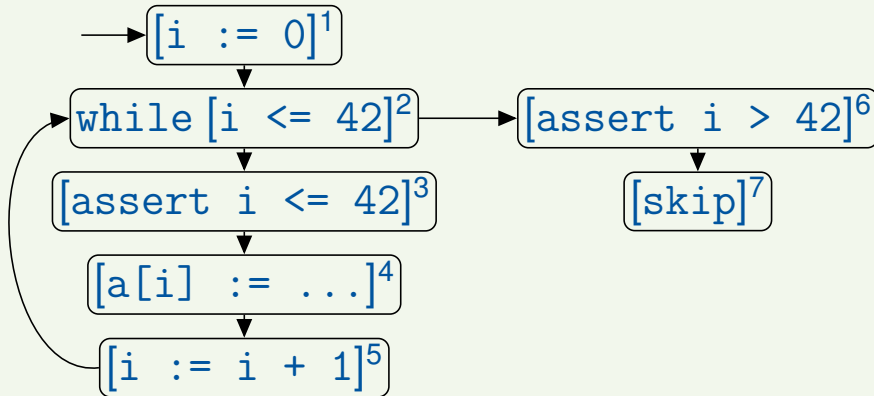
$$\varphi_6(J) = J \cap [43, +\infty]$$

Worklist w. widening	AI ₁	AI ₂	AI ₃	AI ₄	AI ₅	AI ₆	AI ₇
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	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 0]$	\emptyset	\emptyset
52, 26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset
26, 67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset	\emptyset
67	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	\emptyset
ε	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing

$\text{fix}^\nabla(\Phi_S)$

AI_1

AI_2

AI_3

AI_4

AI_5

AI_6

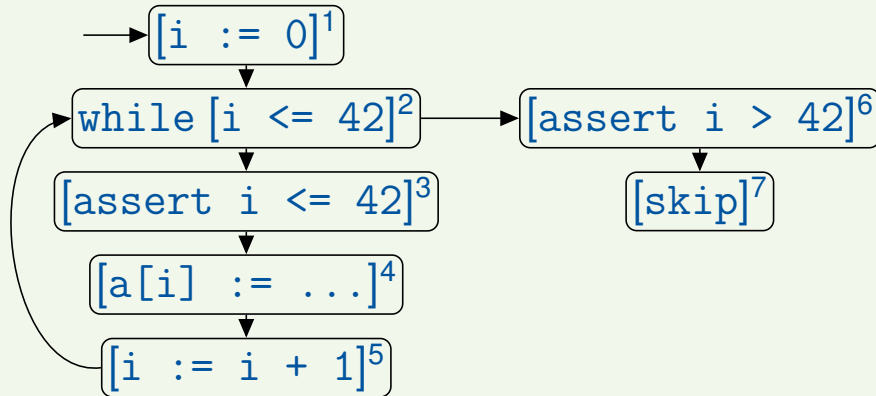
AI_7

$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
----------------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

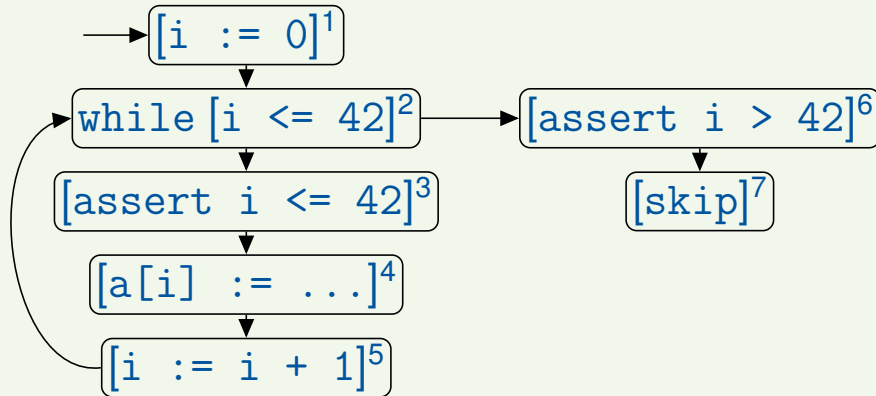
$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing	Al ₁	Al ₂	Al ₃	Al ₄	Al ₅	Al ₆	Al ₇
$\text{fix}^\nabla(\Phi_S)$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

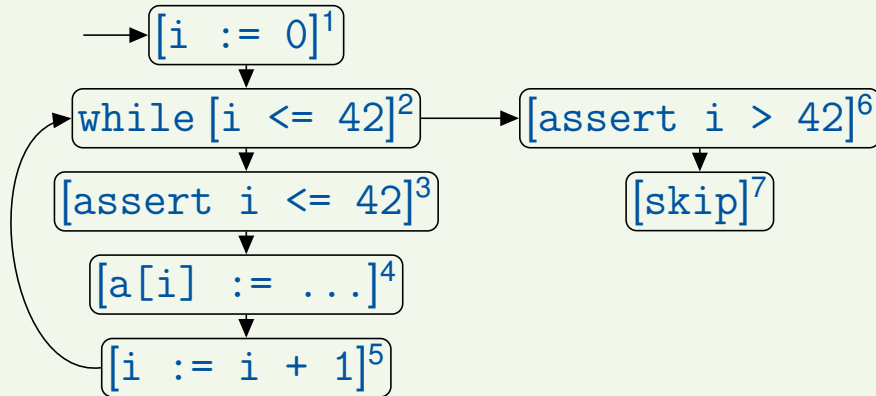
$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing	Al ₁	Al ₂	Al ₃	Al ₄	Al ₅	Al ₆	Al ₇
$\text{fix}^\nabla(\Phi_S)$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^2(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

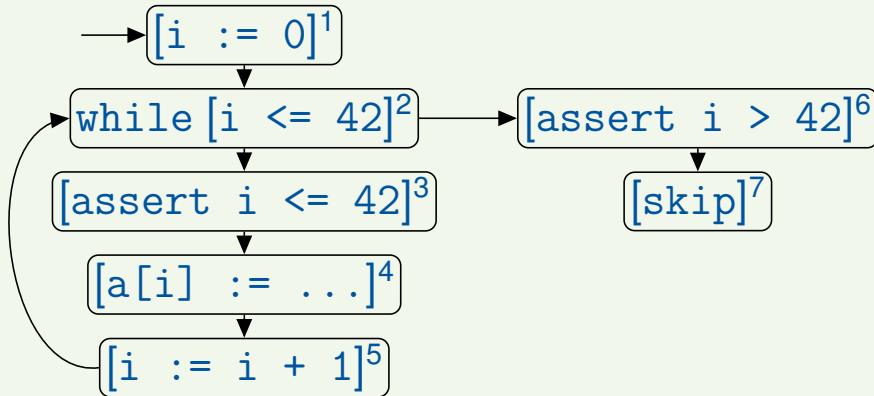
$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing	Al ₁	Al ₂	Al ₃	Al ₄	Al ₅	Al ₆	Al ₇
$\text{fix}^\nabla(\Phi_S)$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^2(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^3(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

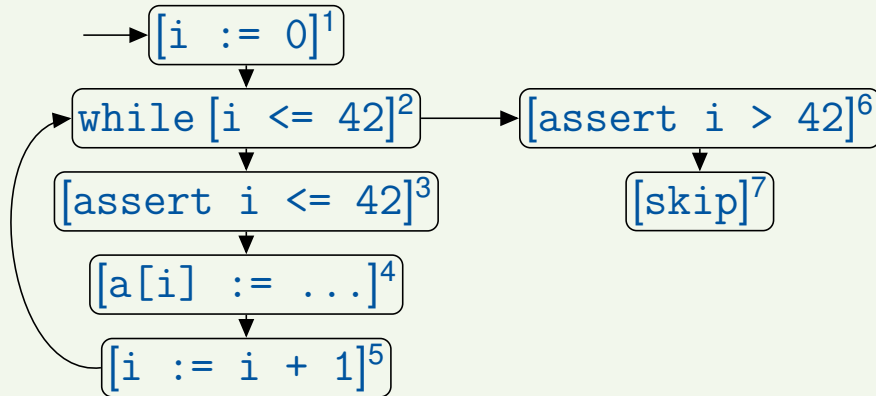
$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing	Al ₁	Al ₂	Al ₃	Al ₄	Al ₅	Al ₆	Al ₇
$\text{fix}^\nabla(\Phi_S)$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^2(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^3(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^4(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, 43]$	$[0, 42]$	$[0, 42]$	$[0, 43]$	$[43, +\infty]$

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

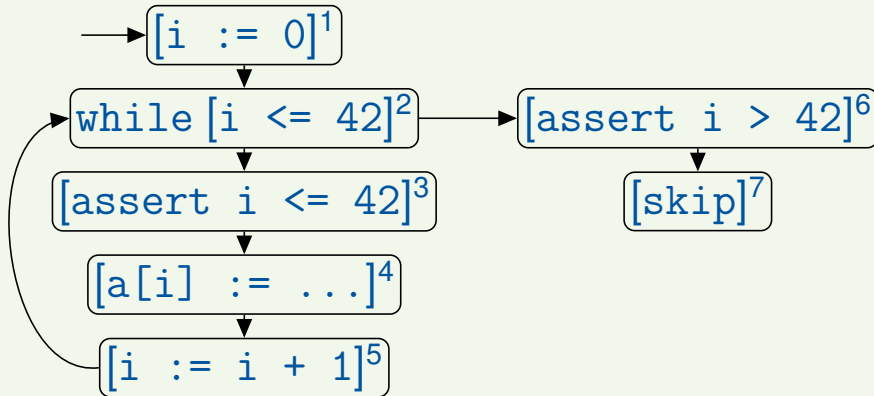
$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing	Al ₁	Al ₂	Al ₃	Al ₄	Al ₅	Al ₆	Al ₇
$\text{fix}^\nabla(\Phi_S)$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^2(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^3(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^4(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, 43]$	$[0, 42]$	$[0, 42]$	$[0, 43]$	$[43, +\infty]$
$\Phi_S^5(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, 43]$	$[0, 42]$	$[0, 42]$	$[0, 43]$	$[43, 43]$

Interval Analysis with Assertions

An Example (continued)

Example 9.2 (Interval analysis for array index; continued)



$$\varphi_1(J) = [0, 0]$$

$$\varphi_2(J) = J$$

$$\varphi_3(J) = J \cap [-\infty, 42]$$

$$\varphi_4(J) = J$$

$$\varphi_5(\emptyset) = \emptyset$$

$$\varphi_5([i_1, i_2]) = [i_1 + 1, i_2 + 1]$$

$$\varphi_6(J) = J \cap [43, +\infty]$$

Narrowing	Al ₁	Al ₂	Al ₃	Al ₄	Al ₅	Al ₆	Al ₇
$\text{fix}^\nabla(\Phi_S)$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, +\infty]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^2(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, +\infty]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^3(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, +\infty]$	$[0, 42]$	$[0, 42]$	$[0, +\infty]$	$[43, +\infty]$
$\Phi_S^4(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, 43]$	$[0, 42]$	$[0, 42]$	$[0, 43]$	$[43, +\infty]$
$\Phi_S^5(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, 43]$	$[0, 42]$	$[0, 42]$	$[0, 43]$	$[43, 43]$
$\Phi_S^6(\text{fix}^\nabla(\Phi_S))$	$[-\infty, +\infty]$	$[0, 43]$	$[0, 43]$	$[0, 42]$	$[0, 42]$	$[0, 43]$	$[43, 43]$

The Java Virtual Machine

Outline of Lecture 9

Recap: Taking Conditional Branches into Account

Interval Analysis with Assertions

The Java Virtual Machine

The Java Bytecode Verifier

The Type-Level Abstract Interpreter

The Java Virtual Machine

Java Bytecode

- **Intermediate language** between high-level language and machine code
- Execution on **Java Virtual Machine** (JVM)

The Java Virtual Machine

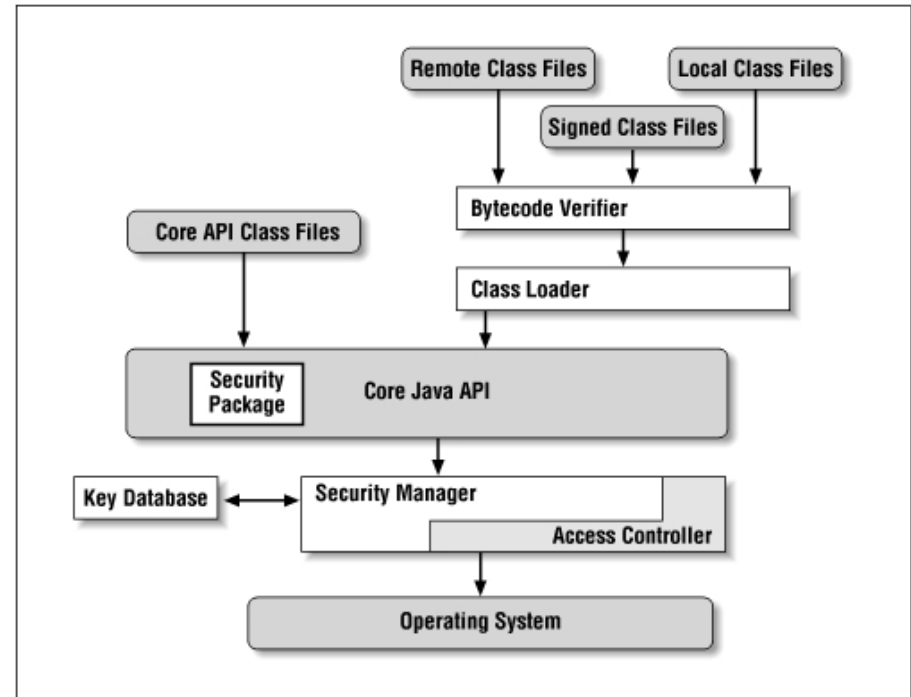
Java Bytecode

- **Intermediate language** between high-level language and machine code
- Execution on **Java Virtual Machine** (JVM)
- **Advantages:**
 - architecture independency (especially for web applications)
 - faster than pure (i.e., source code) interpretation
- **Problem: security issues**
 - destruction of data
 - modification of data
 - disclosure of personal information
 - modification of other programs

The Java Virtual Machine

Java Security: the Sandbox

- **Insulation layer** providing indirect access to system resources
- Hardware access via **API classes and methods**
- **Bytecode verification** upon loading
 - well-typedness
 - proper object referencing
 - proper control flow



The Java Virtual Machine

The Java Virtual Machine

- Conventional **stack-based abstract machine**
- Supports **object-oriented features**: classes, methods, etc.
- **Stack** for method parameters and intermediate results of expression evaluations
- **Registers** for source-level local variables
- Both part of **method activation record** (and thus preserved across method calls)
- Method entry point specifies **required number** of
 - registers (m_r)
 - stack slots (m_s ; for memory allocation)
- (Most) instructions are **typed**

The Java Virtual Machine

Example: Factorial Function

Example 9.3 (Factorial function)

Java source code:

```
static int factorial(int n)
{ int res;
  for (res = 1; n > 0; n--) res = res * n;
  return res; }
```

The Java Virtual Machine

Example: Factorial Function

Example 9.3 (Factorial function)

Java source code:

```
static int factorial(int n)
{ int res;
  for (res = 1; n > 0; n--) res = res * n;
  return res; }
```

Corresponding JVM bytecode:

```
method static int factorial(int), 2 registers, 2 stack slots
 1: istore 0    // store n in register 0
 2: iconst_1   // push constant 1
 3: istore 1    // store res in register 1
 4: iload 0    // push n
 5: ifle 12    // if <= 0, go to end
 6: iload 1    // push res
 7: iload 0    // push n
 8: imul      // res * n on top of stack
 9: istore 1    // store in res
10: iinc 0, -1 // decrement n
11: goto 4     // go to loop header
12: iload 1    // push res
13: ireturn   // return res to caller
```

The Java Virtual Machine

JVM Instruction Set (excerpt; \approx 200 instructions in total)

`iload n` : push integer from register n

`istore n` : pop integer into register n

`iconst_ z` : push integer z

`aconst_null`: push null reference

`iadd`: add two topmost integers on stack and push sum

`getfield C f τ` : pop reference to object o (of class C) and push value of field f of o (of type τ)

`putfield C f τ` : pop value v (of type τ) and reference to object o (of class C) and assign v to field f of o

`new C` : create new object (of class C) and push reference

`invoke C M $\tau_0(\tau_1, \dots, \tau_n)$` : pop values v_1, \dots, v_n (of type τ_1, \dots, τ_n) and reference to object o (of class C), call method M of o with parameters v_1, \dots, v_n , and push return value (of type τ_0)

`if_icmpeq l` : pop two topmost integers from stack and jump to line l if equal

`ireturn`: return to caller with integer result on top of stack

Malicious Bytecode

Example 9.4 (Malicious bytecode)

```
1: iconst_5
2: iconst_1
3: putfield A f int
```

interprets second stack entry (5) as reference to object of class **A** and assigns topmost stack entry (1) to field **f** of this object

The Java Bytecode Verifier

Outline of Lecture 9

Recap: Taking Conditional Branches into Account

Interval Analysis with Assertions

The Java Virtual Machine

The Java Bytecode Verifier

The Type-Level Abstract Interpreter

The Java Bytecode Verifier

Correctness of Bytecode

Conditions to ensure **proper operation**:

Type correctness: arguments of instructions always of expected type

No stack over-/underflow: never push to full stack or pop from empty stack

Code containment: PC must always point into the method code

Register initialization: load from non-parameter register only after store

Object initialization: constructor must be invoked before using class instance

Access control: operations must respect visibility modifiers
(`private/protected/public`)

The Java Bytecode Verifier

Correctness of Bytecode

Conditions to ensure **proper operation**:

Type correctness: arguments of instructions always of expected type

No stack over-/underflow: never push to full stack or pop from empty stack

Code containment: PC must always point into the method code

Register initialization: load from non-parameter register only after store

Object initialization: constructor must be invoked before using class instance

Access control: operations must respect visibility modifiers

(`private/protected/public`)

Options:

- **dynamic checking** at execution time (“defensive JVM approach”)
 - expensive, slows down execution
- **static checking** at loading time (here)
 - verified code executable at full speed without extra dynamic checks

The Java Bytecode Verifier

The Java Bytecode Verifier

Summary: **dataflow analysis** applied to **type-level abstract interpretation** of JVM

1. Association of **type information** with register and stack contents
 - set of types forms a complete lattice
2. Simulation of **execution of instructions** at type level (“**symbolic execution**”)
3. Use **dataflow analysis** to cover all concrete executions
4. **Modular analysis**: proceeds method per method

(see X. Leroy: *Java Bytecode Verification: Algorithms and Formalizations*, Journal of Automated Reasoning 30(3-4), 2003, 235–269)

The Type-Level Abstract Interpreter

Outline of Lecture 9

Recap: Taking Conditional Branches into Account

Interval Analysis with Assertions

The Java Virtual Machine

The Java Bytecode Verifier

The Type-Level Abstract Interpreter

The Type-Level Abstract Interpreter

Types

The **set of types**, *Typ*, is composed of

- **Primitive** types:
 - *int* (covering *boolean*, *byte*, *char*, *short*)
 - *long*
 - *float*
 - *double*

The Type-Level Abstract Interpreter

Types

The **set of types**, *Typ*, is composed of

- **Primitive** types:
 - *int* (covering *boolean*, *byte*, *char*, *short*)
 - *long*
 - *float*
 - *double*
- **Object reference** types: *C* for every class name *C*

The Type-Level Abstract Interpreter

Types

The **set of types**, *Typ*, is composed of

- **Primitive** types:
 - *int* (covering *boolean*, *byte*, *char*, *short*)
 - *long*
 - *float*
 - *double*
- **Object reference** types: *C* for every class name *C*
- **Array** types: $\tau[]$ for every primitive or object reference type τ

The Type-Level Abstract Interpreter

Types

The **set of types**, Typ , is composed of

- **Primitive** types:
 - int (covering $boolean$, $byte$, $char$, $short$)
 - $long$
 - $float$
 - $double$
- **Object reference** types: C for every class name C
- **Array** types: $\tau[]$ for every primitive or object reference type τ
- **Method** types: $\tau_0(\tau_1, \dots, \tau_n)$ for $n \in \mathbb{N}$, $\tau_i \in Typ$

The Type-Level Abstract Interpreter

Types

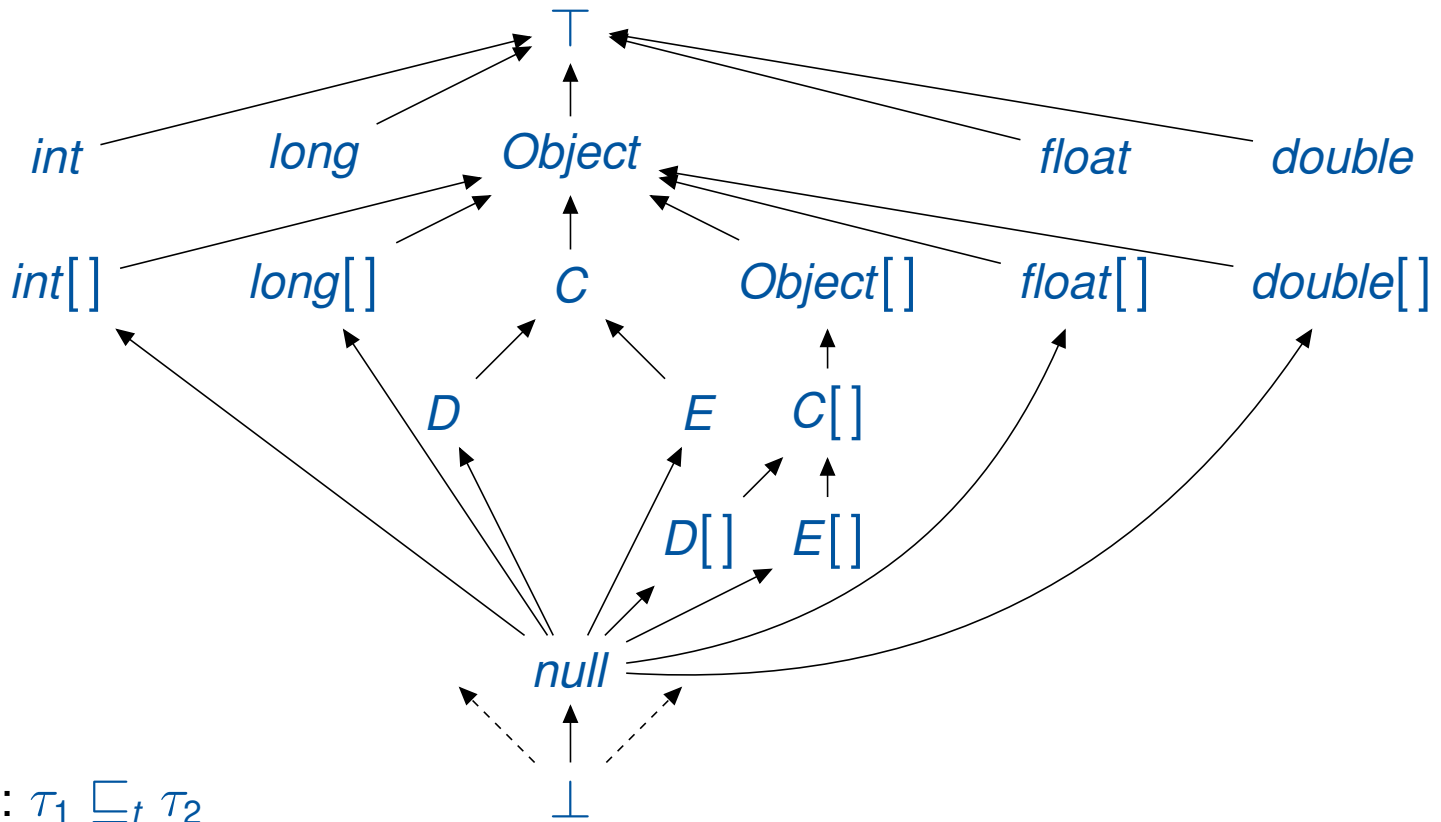
The **set of types**, Typ , is composed of

- **Primitive** types:
 - *int* (covering *boolean*, *byte*, *char*, *short*)
 - *long*
 - *float*
 - *double*
- **Object reference** types: C for every class name C
- **Array** types: $\tau[]$ for every primitive or object reference type τ
- **Method** types: $\tau_0(\tau_1, \dots, \tau_n)$ for $n \in \mathbb{N}$, $\tau_i \in Typ$
- **Special** types:
 - *null* (null reference)
 - *Object* (any object)
 - \top (contents of uninitialised registers, i.e., any value)
 - \perp (absence of any value)

The Type-Level Abstract Interpreter

The Subtyping Relation (excerpt)

(C , D , E user-defined classes; D , E extending C)



Notation: $\tau_1 \sqsubseteq_t \tau_2$

The Type-Level Abstract Interpreter

The Type-Level Abstract Interpreter

- **Idea:** execute JVM instructions on **types** (rather than concrete values)
 - **stack type** $S \in \text{Typ}^{\leq m_s}$ (top to the left)
 - **register type** $R : \{0, \dots, m_r - 1\} \rightarrow \text{Typ}$

The Type-Level Abstract Interpreter

The Type-Level Abstract Interpreter

- **Idea:** execute JVM instructions on **types** (rather than concrete values)
 - **stack type** $S \in \text{Typ}^{\leq m_s}$ (top to the left)
 - **register type** $R : \{0, \dots, m_r - 1\} \rightarrow \text{Typ}$
- Represented as **transition relation**

$$i : (S, R) \rightarrow (S', R')$$

where

- i : current instruction
- (S, R) : stack/register type before execution
- (S', R') : stack/register type after execution

The Type-Level Abstract Interpreter

The Type-Level Abstract Interpreter

- **Idea:** execute JVM instructions on **types** (rather than concrete values)
 - **stack type** $S \in \text{Typ}^{\leq m_s}$ (top to the left)
 - **register type** $R : \{0, \dots, m_r - 1\} \rightarrow \text{Typ}$
- Represented as **transition relation**

$$i : (S, R) \rightarrow (S', R')$$

where

- i : current instruction
- (S, R) : stack/register type before execution
- (S', R') : stack/register type after execution
- **Errors** (type mismatch, stack over-/underflow, ...) denoted by absence of transition

The Type-Level Abstract Interpreter

Some Transition Rules

<code>iconst_z :</code>	$(S, R) \rightarrow (int.S, R)$	if $ S < m_s$
<code>aconst_null :</code>	$(S, R) \rightarrow (null.S, R)$	if $ S < m_s$
<code>iadd :</code>	$(int.int.S, R) \rightarrow (int.S, R)$	
<code>if_icmpeq l :</code>	$(int.int.S, R) \rightarrow (S, R)$	
<code>iload n :</code>	$(S, R) \rightarrow (int.S, R)$	if $0 \leq n < m_r, R(n) = int, S < m_s$
<code>aload n :</code>	$(S, R) \rightarrow (R(n).S, R)$	if $0 \leq n < m_r, R(n) \sqsubseteq_t Object, S < m_s$
<code>istore n :</code>	$(int.S, R) \rightarrow (S, R[n \mapsto int])$	if $0 \leq n < m_r$
<code>astore n :</code>	$(\tau.S, R) \rightarrow (S, R[n \mapsto \tau])$	if $0 \leq n < m_r, \tau \sqsubseteq_t Object$
<code>getfield C f τ :</code>	$(D.S, R) \rightarrow (\tau.S, R)$	if $D \sqsubseteq_t C$
<code>putfield C f τ :</code>	$(\tau'.D.S, R) \rightarrow (S, R)$	if $\tau' \sqsubseteq_t \tau, D \sqsubseteq_t C$
<code>invoke C M σ :</code>	$(\tau'_n \dots \tau'_1.\tau'.S, R) \rightarrow (\tau_0.S, R)$	if $\sigma = \tau_0(\tau_1, \dots, \tau_n), \tau'_i \sqsubseteq_t \tau_i$ for $1 \leq i \leq n, \tau' \sqsubseteq_t C$

Some Theoretical Properties

Lemma 9.5

1. (Typ, \sqsubseteq_t) is a *complete lattice satisfying ACC*.

Some Theoretical Properties

Lemma 9.5

1. (Typ, \sqsubseteq_t) is a *complete lattice satisfying ACC*.
2. (*Determinacy*) The transitions of the abstract interpreter define a partial function:
If $i : (S, R) \rightarrow (S_1, R_1)$ and $i : (S, R) \rightarrow (S_2, R_2)$, then $S_1 = S_2$ and $R_1 = R_2$.

Some Theoretical Properties

Lemma 9.5

1. (Typ, \sqsubseteq_t) is a *complete lattice satisfying ACC*.
2. (*Determinacy*) The transitions of the abstract interpreter define a partial function:
If $i : (S, R) \rightarrow (S_1, R_1)$ and $i : (S, R) \rightarrow (S_2, R_2)$, then $S_1 = S_2$ and $R_1 = R_2$.
3. (*Soundness*) If $i : (S, R) \rightarrow (S', R')$, then for all concrete states (s, r) matching (S, R) , the defensive JVM will not stop with a run-time type exception when applying i to (s, r) (but rather change to some (s', r') matching (S', R')).

The Type-Level Abstract Interpreter

Some Theoretical Properties

Lemma 9.5

1. (Typ, \sqsubseteq_t) is a *complete lattice satisfying ACC*.
2. (*Determinacy*) The transitions of the abstract interpreter define a partial function:
If $i : (S, R) \rightarrow (S_1, R_1)$ and $i : (S, R) \rightarrow (S_2, R_2)$, then $S_1 = S_2$ and $R_1 = R_2$.
3. (*Soundness*) If $i : (S, R) \rightarrow (S', R')$, then for all concrete states (s, r) matching (S, R) , the defensive JVM will not stop with a run-time type exception when applying i to (s, r) (but rather change to some (s', r') matching (S', R')).

Proof.

see X. Leroy: *Java Bytecode Verification: Algorithms and Formalizations* □