CS477 Formal Software Dev Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

Slides based in part on previous lectures
by Mahesh Vishwanathan, and by Gul Agha

April 25, 2018

Domain for Standard Interpretation

- Given (N, I, K, E) a control flow graph with labels using variables from Var
- Let $Val = values \cup \{\top, \bot\}$, the extended set of values, ordered as before
  - Val is a complete lattice.
- Let $Env = \{\rho \mid \rho : Var \to Val\}$
  - Env is a complete lattice
  - An env used to be a partial function; now map undefined to $\bot$
  - $val : (Exp \times Env) \to Val$
  - Will assume $\{\text{true, false}\} \subseteq \text{values}$
  - $bval : (BExp \times Env) \to \{\text{true, false}\} \cup \{\top, \bot\} \subseteq Val$
- Let $States = (E \cup \{\top, \bot\}) \times Env$
  - $States$ is a complete lattice assuming the order $((e, \rho) \leq (e', \rho')) \equiv ((e \leq e') \wedge (\rho \leq \rho'))$.

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $I(n)$:

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - next_state$((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq \text{Enter}$
  - $l(n) = \text{Exit} \Rightarrow \text{next\_state}((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - $\text{next\_state}((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$
  - $l(n) = (if\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - next_state$((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$
  - $l(n) = (if\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
    - if $bval(b, \rho) = \bot$ then next_state$((m, k, n), \rho) = (\bot, \rho)$

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq \text{Enter}$
  - $l(n) = \text{Exit} \Rightarrow \text{next\_state}((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - $\text{next\_state}((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$
  - $l(n) = (\mathit{if}\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
    - if $bval(b, \rho) = \bot$ then $\text{next\_state}((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then $\text{next\_state}((m, k, n), \rho) = (\top, \rho)$

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - next_state$((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$
  - $l(n) = (if\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
    - if $bval(b, \rho) = \bot$ then next_state$((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then next_state$((m, k, n), \rho) = (\top, \rho)$
    - $bval(b, \rho) =$ true then next_state$((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$

Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - next_state$((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$
  - $l(n) = (if\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
    - if $bval(b, \rho) = \bot$ then next_state$((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then next_state$((m, k, n), \rho) = (\top, \rho)$
    - $bval(b, \rho) =$ true then next_state$((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$
    - $bval(b, \rho) =$ false then next_state$((m, k, n), \rho) = ((n, \text{suc}, q), \rho)$

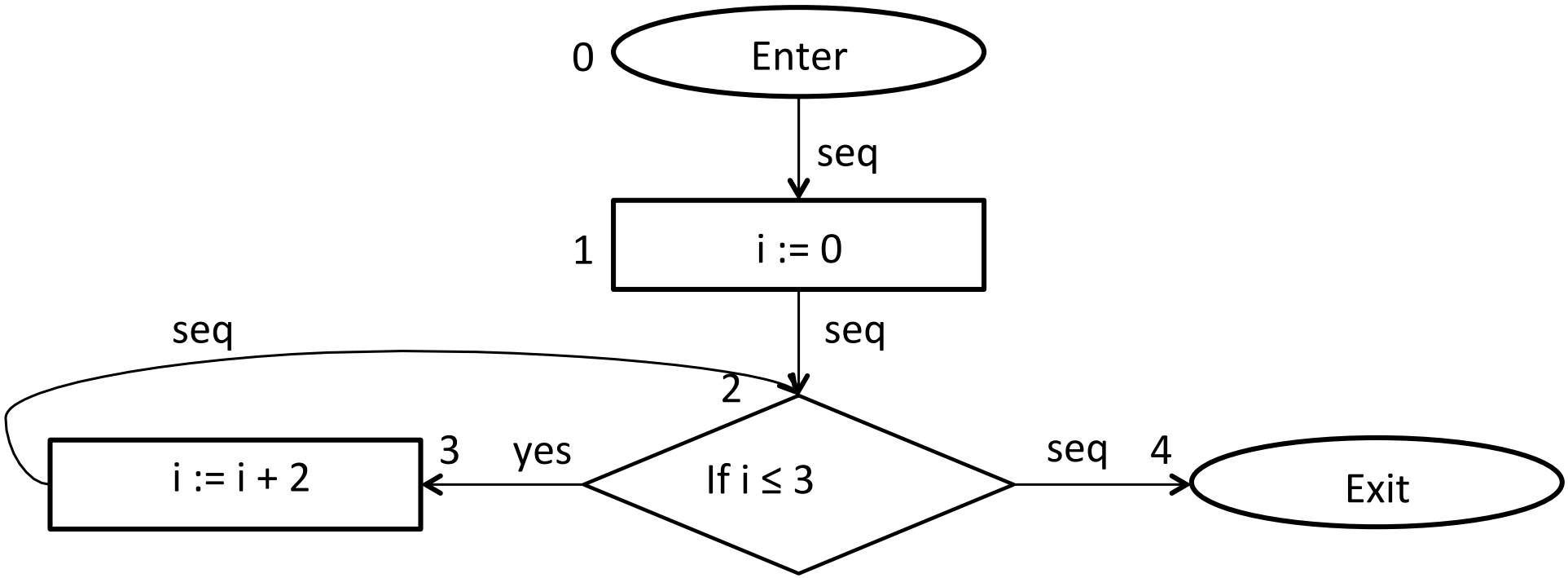
Transitions in Control Flow Graphs

- next_state : $States \rightarrow States$
- next_state$(\top, \rho) = (\top, \rho)$; next_state$(\bot, \rho) = (\bot, \rho)$
- next_state$((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ Enter
  - $l(n) =$ Exit $\Rightarrow$ next_state$((m, k, n), \rho) = ((m, k, n), \rho)$
  - $l(n) = (i := e)$, then n has unique successor node p, $(n, \text{suc}, p) \in E$.
    - next_state$((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto val(e, \rho)])$
  - $l(n) = (\mathit{if}\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
    - if $bval(b, \rho) = \bot$ then next_state$((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then next_state$((m, k, n), \rho) = (\top, \rho)$
    - $bval(b, \rho) =$ true then next_state$((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$
    - $bval(b, \rho) =$ false then next_state$((m, k, n), \rho) = ((n, \text{suc}, q), \rho)$
- next_state is transition semantics for control flow graphs

Example

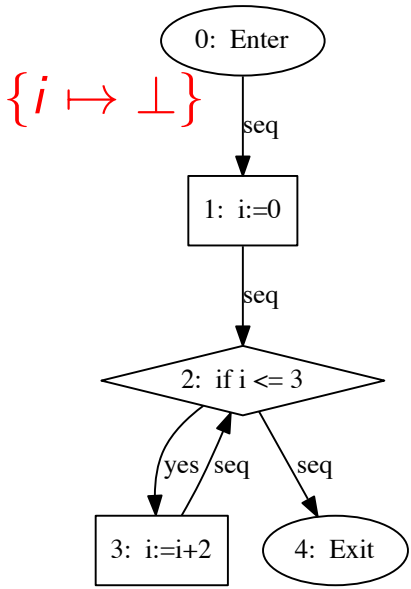
Consider the following control flow graph (N, I, K, E) where:

- $Var = \{i\}, \quad values = \mathbb{Z}$
- $N = \{0, 1, 2, 3, 4, 5, 6\}$
- $I(0) = \text{Enter}, \ I(1) = \texttt{i:=0}, \ I(2) = \texttt{if i} \leq 3,$
  $I(3) = \texttt{i:=i+2}, \ I(4) = \text{Exit}$
- $K = \{\text{yes}, \text{seq}\}$
- $E = \left\{ \begin{array}{ll} (0, \text{seq}, 1), & (1, \text{seq}, 2), \\ (2, \text{yes}, 3), & (2, \text{seq}, 4), \\ (3, \text{seq}, 2) & \end{array} \right\}$


0
Enter
seq
1
i := 0
seq
seq
2
3
yes
If i ≤ 3
seq
4
i := i + 2
Exit

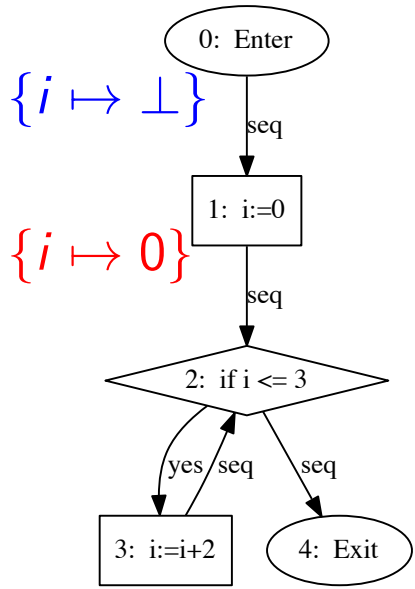
Example: next_state

- next_state$((0, \text{seq}, 1), \{i \mapsto \bot\}) =$



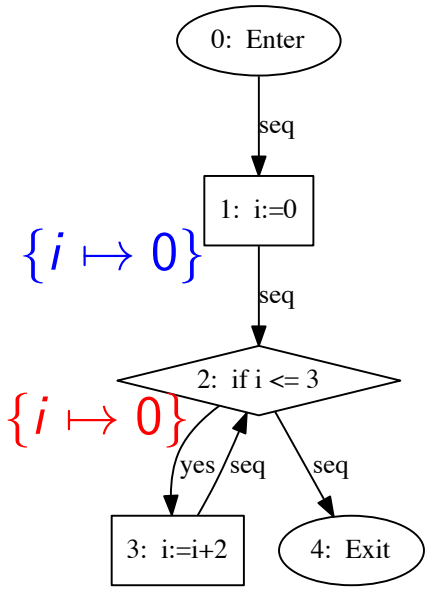
Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$



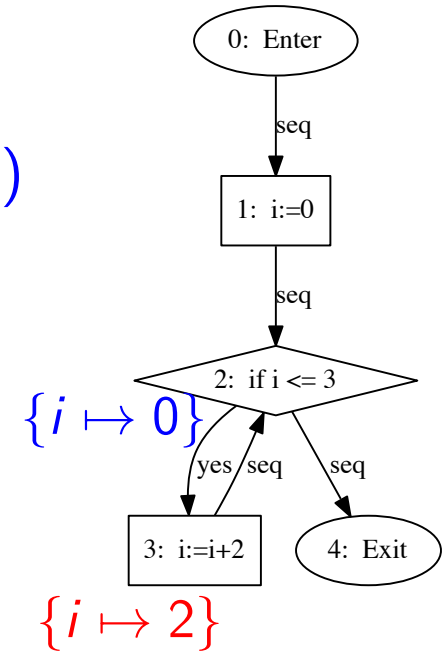
Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next\_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$



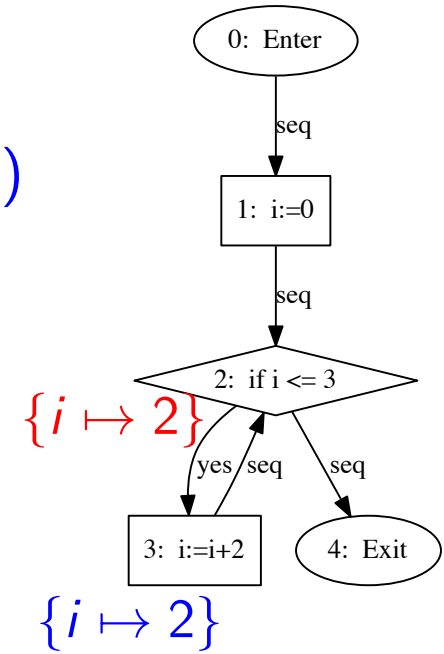
Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next\_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 0\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 0\}[i \mapsto 0 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 2\})$



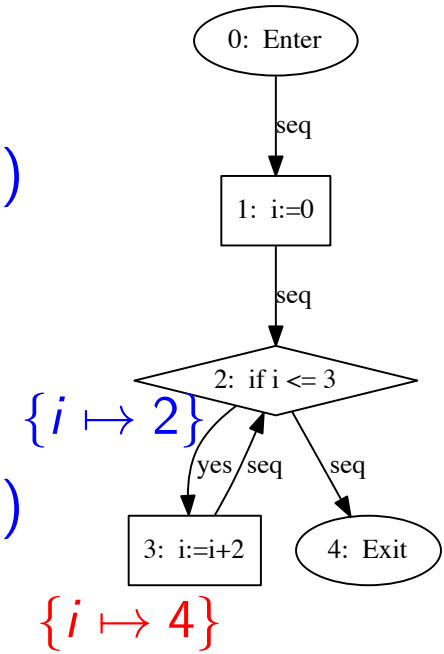
Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next\_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 0\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 0\}[i \mapsto 0 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 2\})$
- Since $\{i \mapsto 2\}(i) = 2 \leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 2\}) = ((2, \text{yes}, 3), \{i \mapsto 2\})$



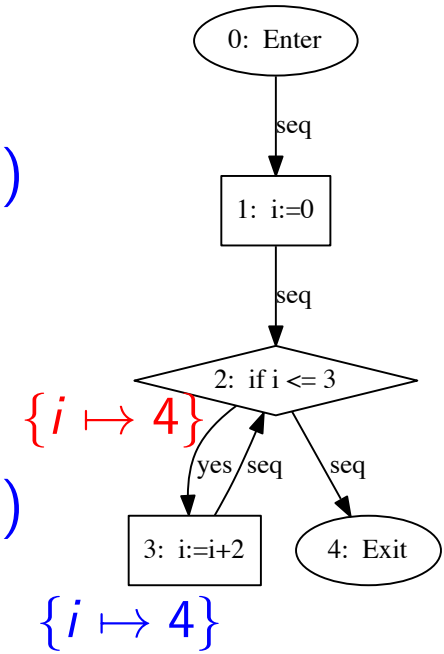
Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next\_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 0\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 0\}[i \mapsto 0 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 2\})$
- Since $\{i \mapsto 2\}(i) = 2 \leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 2\}) = ((2, \text{yes}, 3), \{i \mapsto 2\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 2\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 2\}[i \mapsto 2 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 4\})$



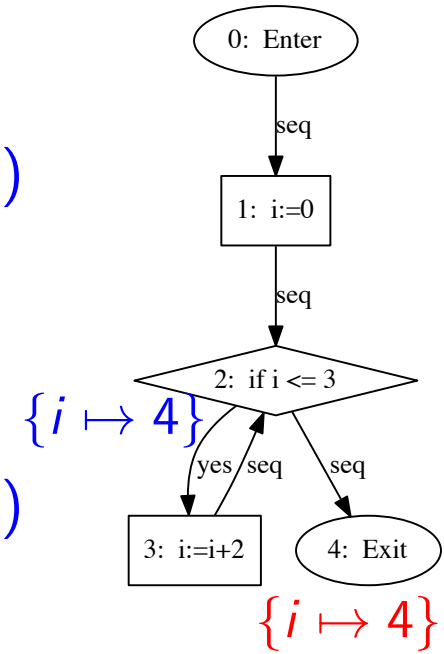
Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next\_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 0\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 0\}[i \mapsto 0 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 2\})$
- Since $\{i \mapsto 2\}(i) = 2 \leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 2\}) = ((2, \text{yes}, 3), \{i \mapsto 2\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 2\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 2\}[i \mapsto 2 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 4\})$
- Since $\{i \mapsto 4\}(i) = 4 \not\leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 4\}) = ((2, \text{seq}, 4), \{i \mapsto 4\})$



Example: next_state

- $\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next\_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 0\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 0\}[i \mapsto 0 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 2\})$
- Since $\{i \mapsto 2\}(i) = 2 \leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 2\}) = ((2, \text{yes}, 3), \{i \mapsto 2\})$
- $\text{next\_state}((2, \text{yes}, 3), \{i \mapsto 2\}) =$
  $((3, \text{seq}, 2), \{i \mapsto 2\}[i \mapsto 2 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 4\})$
- Since $\{i \mapsto 4\}(i) = 4 \not\leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 4\}) = ((2, \text{seq}, 4), \{i \mapsto 4\})$
- Since $l(4) = \text{Exit}$ we have
  $\text{next\_state}((2, \text{seq}, 4), \{i \mapsto 4\}) = ((2, \text{seq}, 4), \{i \mapsto 4\})$



Standard Interpretation and Semantics

- Let $Interp(\theta, (m, k, n))$ be the lifting of next_state to sets of environments (contexts)
- If θ tells all the environments we might come into our edge with, $Interp(\theta, (m, k, n))$ tells us the set of environments we may leave with
  - Note: $l(m) \neq \text{Exit}$
  - If $l(m) = \text{Enter}$ there are no inbound edges to m, so $Interp(\theta, (m, k, n)) = \{\text{next\_state}((m, k, n), \{v \mapsto \perp | v \in Var\})\}$
  - $l(m) \neq \text{Enter} \Rightarrow$
    $Interp(\theta, (m, k, n)) =$
    $\{\rho \mid \exists m',\ k',\ \rho' \mid (m', k', m) \in E \wedge \rho' \in \theta((m', k', m)) \wedge$
    $\text{next\_state}((m', k', m), \rho') = ((m, k, n), \rho)\}$

Standard Interpretation and Semantics

- Let $Contexts = \mathcal{P}(Env)$
  - $Contexts$ is a complete lattice
  - A context corresponds to a formula in predicate logic over the program variables (recall how we encoded Hoare Logic)
- If for all $e \in E$ we have $\theta(e) \subseteq \phi(e)$, then for all $e' \in E$ we have $Interp(\theta, e') \subseteq Interp(\phi, e')$
- **Result:** $(Contexts, Interp)$ is an abtract interpretation
- Recall: $Interp : ((E \to Contexts) \times E) \to Contexts$ so $\overline{Interp} : (E \to Contexts) \to (E \to Contexts)$
- $\overline{Interp}(\theta)(e) = Interp(\theta, e) = \overline{Interp}^1(\theta)(e)$

- $\overline{Interp}^{n+1}(\theta)(e) = \overline{Interp}(\overline{Interp}^n(\theta))(e)$
- $\mu\, \overline{Interp}(e) = \bigcup_{n \in \mathbb{N}} \overline{Interp}^n\{e' \mapsto \{\} \mid e' \in E\}(e)$
- $\mu\, \overline{Interp}$ tells us the best knowledge we can know about our program
- Problem: May take an unbounded amount of computation; as informative as transition semantics

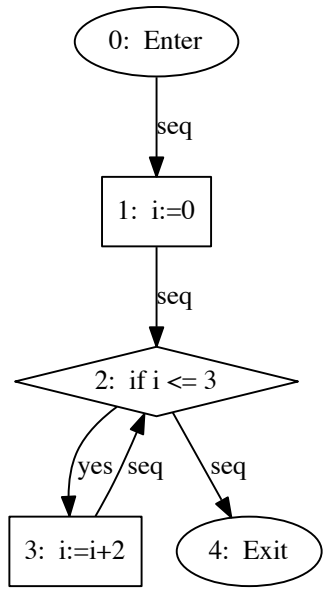
Example: *Interp*

Let θ map edges to sets of environments. *Interp* will tell us the set of environments next_state will associate with each edge assuming θ gives a set of (possibly) possible environments for each predecessor edge:

- Since $\mathit{Var} = \{i\}$,
  $\mathit{Interp}(\theta, (0, \text{seq}, 1)) = \{\text{next\_state}((0, \text{seq}, 1), \{i \mapsto \bot\})\} = \{\{i \mapsto 0\}\}$
- Since $I(2) =$ `if i <= 3` we have $\mathit{Interp}(\theta, (1, \text{seq}, 2)) = \theta((0, \text{seq}, 1))$
- $\mathit{Interp}(\theta, (2, \text{yes}, 3)) =$
  $\{\rho[i \mapsto \rho(i) + 2] \mid \rho \in (\theta(1, \text{seq}, 2) \cup \theta(3, \text{seq}, 2)) \wedge \rho(i) \leq 3\}$
- $\mathit{Interp}(\theta, (3, \text{seq}, 2)) = \theta(2, \text{yes}, 3)$
- $\mathit{Interp}(\theta, (2, \text{no},)) = \{\rho \mid \rho \in (\theta(1, \text{seq}, 2) \cup \theta(3, \text{seq}, 2)) \wedge \rho(i) > 3\}$

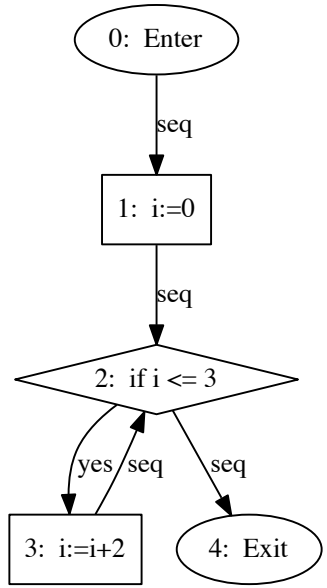
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\,\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\qquad\qquad\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\qquad\qquad\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\qquad\qquad\qquad\qquad\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\qquad\qquad\qquad\qquad\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\qquad\qquad\}$



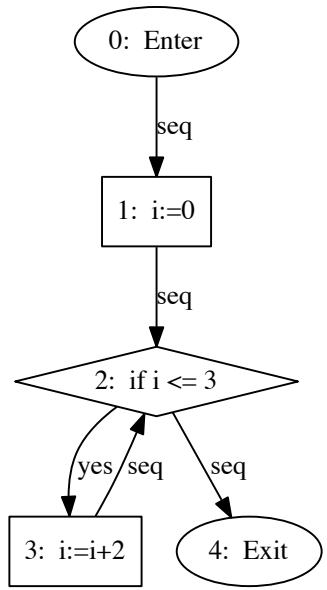
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\,\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\qquad\qquad\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\qquad\qquad\qquad\qquad\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\qquad\qquad\qquad\qquad\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\qquad\qquad\}$



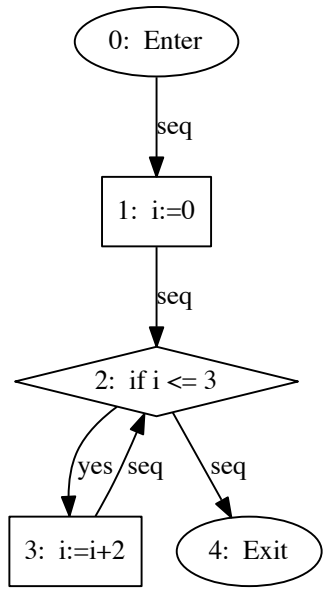
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\quad\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\quad\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\quad\}$



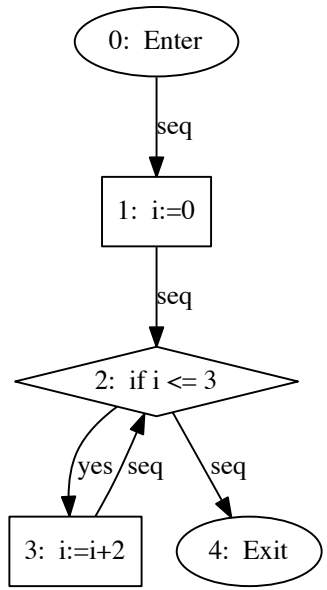
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 2\},\ \ \}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\ \ \}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\ \ \}$



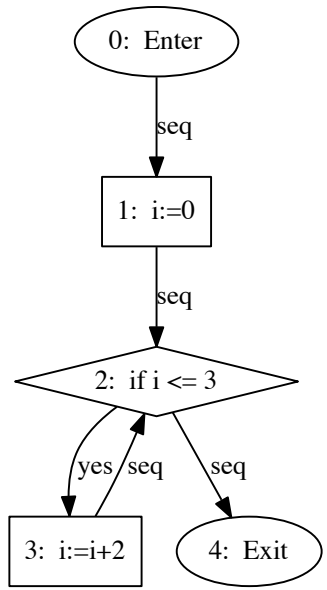
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\,\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 2\},\quad\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\{i \mapsto 2\},\quad\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\quad\}$



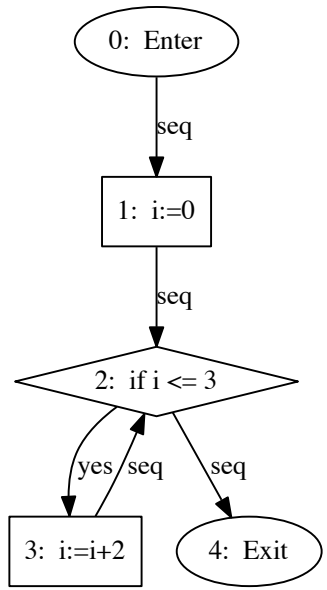
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\,\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\{i \mapsto 2\},\ \ \}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\ \ \}$



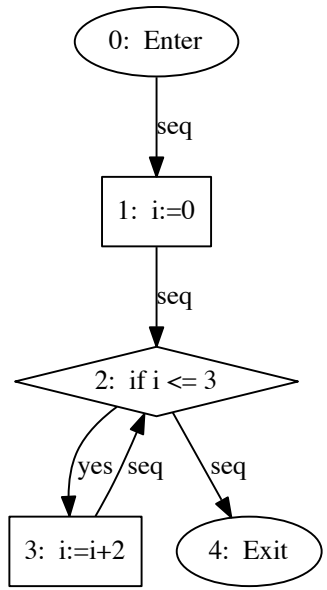
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\,\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\qquad\qquad\}$



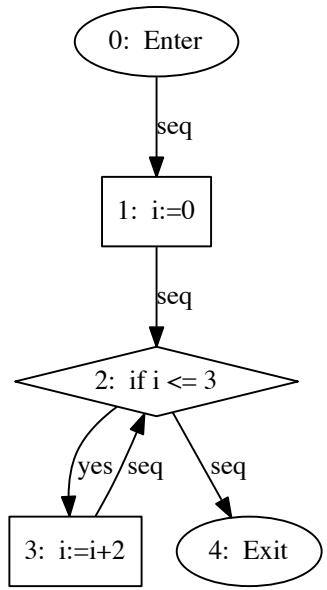
Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\{i \mapsto 4\}\}$



Example: $\mu\,\overline{Interp}$

- $\mu\,\overline{Interp} : E \to Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\}$
- $\mu\,\overline{Interp}(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\,\overline{Interp}(0, \text{seq}, 1) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$
- $\mu\,\overline{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}(3, \text{seq}, 2) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$
- $\mu\,\overline{Interp}((2, \text{no}, 4)) = \{\{i \mapsto 4\}\}$



Soundness of Abstract Semantics

Fact: An abstract interpretation $(A, \mathcal{I})$ is sound (or consistent) with respect to $(Env, Interp)$ if and only if there exist $\alpha,\ \beta$ such that

- $\alpha : Contexts \to A$, $\beta : A \to Contexts$
- $\alpha,\ \beta$ order preserving
- For all $a \in A$ have $\alpha(\beta(a)) = a$
- For all $S \in Contex$, have $S \subseteq \beta(\alpha(S))$
  - The abtract interpretation gives us more possibilities, is less precise
- For all $e \in E$, $\alpha(\mu\, \overline{Interp}(e)) = \mu\, \overline{\mathcal{I}}(e)$
- The pair (α, β) is called a *Galois Insertion*.

- Game: find useful A where we can compute $\mu\, \overline{\mathcal{I}}$, usually in time proportional to size of program

Composing Abstract Interpretations

- Observe: Can abstract an abstraction:
  - Given $\alpha : \mathit{Context} \rightarrow A$, $\beta : A \rightarrow \mathit{Contexts}$ such that for all $a \in A$ have $\alpha(\beta(a)) = a$
  - and for all $S \in \mathit{Contex}$, have $S \subseteq \beta(\alpha(S))$
  - Given $\gamma : A \rightarrow B$, $\delta : B \rightarrow A$ such that for all $b \in b$ have $\gamma(\delta(b)) = b$
  - and for all $a \in A$, have $\delta(\gamma(a)) \leq a$
  - Then $(B, \gamma \circ \mu\overline{\mathcal{I}})$ is another abstract interpretation.

Some Abstract Interpretations

- Replace *Contexts* with $A = Var \rightarrow \mathcal{P}(Val)$. Let $S \in Contexts$. Define $\alpha(S)(x) = \{v \mid \exists s \in S.v = s(x) \wedge s(x) \neq \bot\}$
- Chain with $B = Var \rightarrow \{int, bool, \bot, \top\}$ and
  - $\gamma(a)(x) = int$ if $a(x)$ contains only integers (and $\bot$);
  - $\gamma(a)(x) = int$ if $a(x)$ contains only integers (and $\bot$);
  - $\gamma(a)(x) = \bot$ if $a(x) = \{\}$;
  - $gamma(a)(x) = \top$ otherwise
  - Can be used for type checking
- Chain with $B = Var \rightarrow \{\bot, \top\}$ and
  - $\gamma(a)(x) = \top$ if $a(x)$ contains something other than $\bot$;
  - $\gamma(a)(x) = bot$ if $a(x) = \{\bot\}$ or $a(x) = \{\}$
  - Can be used for checking variables are initialized before they are used.