CS477 Formal Software Dev Methods

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

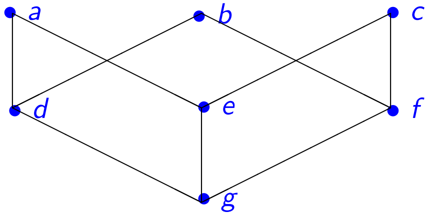
Slides based in part on previous lectures
by Mahesh Vishwanathan, and by Gul Agha

April 20, 2018

Partial Orders

A partial order on a set S is a binary relation $\leq$ on S such that

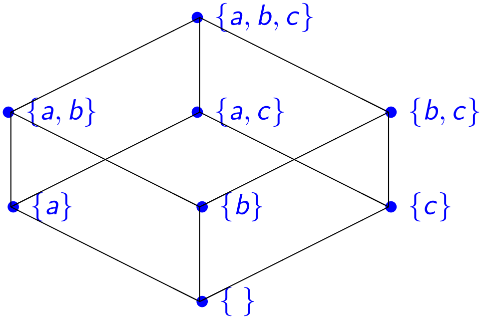
- **[Refl]** $s \leq s$ for all $s \in S$
- **[Antisym]** $s \leq t$ and $t \leq s$ impilies $s = t$, for all $s,\ t \in S$
- **[Trans]** $s \leq t$ and $t \leq u$ impilies $s \leq u$, for all $s,\ t,\ \in S$

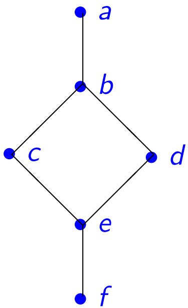


Upper Bounds and Complete Latices

- In a partial order $(S, \leq)$, given $X \subseteq S$, y is an upper bound for X if for all $x \in X$ we have $x \leq y$.
- y is a least upper bound of X, y is an upper bound of X and whenever z is an upper bound of X, $y \leq z$.
- **Note:** Least upper bounds are unique.
- A complete lattice is a partial order $(L, \leq)$ such that for all $X \subseteq S$ there exists a (unique) least upper bound.
- Write $\mathrm{lub}(X)$ or $\bigvee X$ for the least upper bound of X.
- Write $x \vee y$ for $\bigvee\{x, y\}$
- **Note:** $x \vee y = x \iff y \leq x$
- **Note:** Given a set S, $(\mathcal{P}(S), \subseteq)$ is a complete lattice.
- Write $\bot = \bigvee\{\ \}$ and $\top = \bigvee S$

Example Complete Lattices



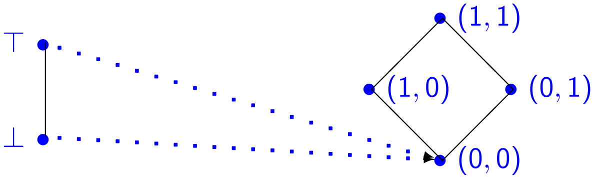


Partial Orders, Functions, and Complete Lattices

- Let X be an arbitrary set and A and B be partial orders.
- A function $f : A \to B$ is order-preserving if, for all $x, y \in A$ with $x \leq y$ we have $f(x) \leq f(y)$
- Function $f, g : X \to A$ may be ordered by pointwise comparrison:
  - Write $f \leq_{fun} g$ to mean that for all $x \in X$ we have $f(x) \leq g(x)$
  - Will leave off the subcript in general
- **Fact:** $(\{f \mid f : X \to B\}, \leq_{fun})$ is a partial order.
- **Fact:** $(\{f \mid f : X \to B\}, \leq_{fun})$ is a complete lattice if B is.
- **Fact:** $(\{f \mid f : A \to B,\ f \text{ order-preserving}\} \leq_{fun})$ is a complete lattice if B is.

Example: Monotone Function Space

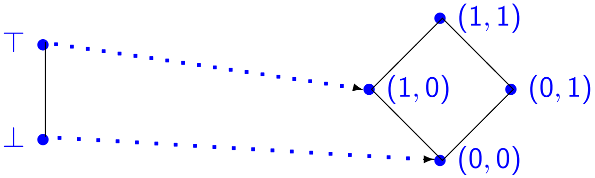
a

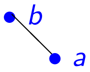


- a

Example: Monotone Function Space

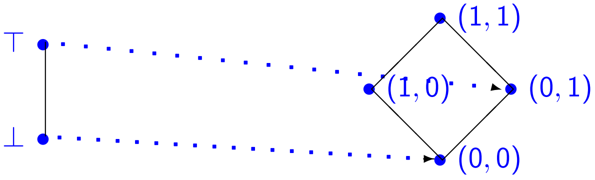
b

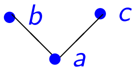




Example: Monotone Function Space

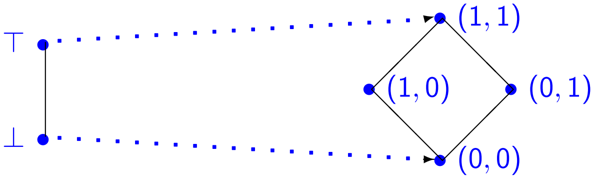
c

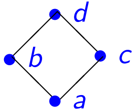




Example: Monotone Function Space

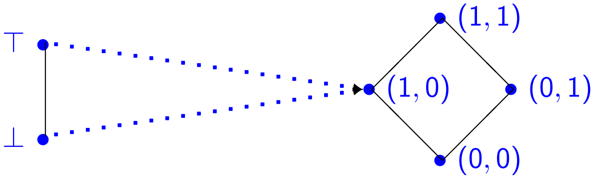
d

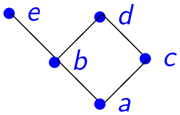




Example: Monotone Function Space

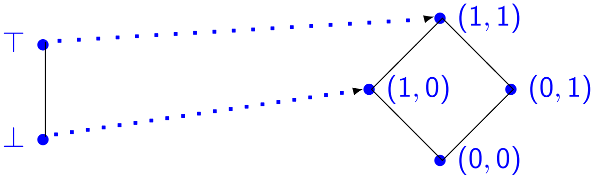
e

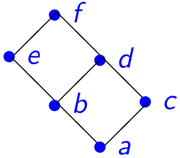




Example: Monotone Function Space

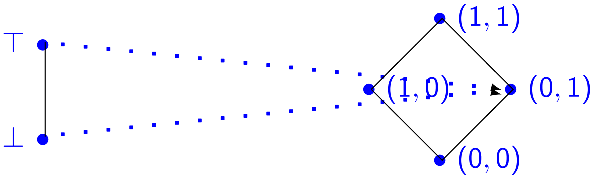
f

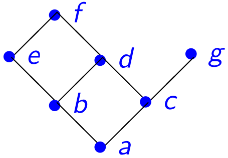




Example: Monotone Function Space

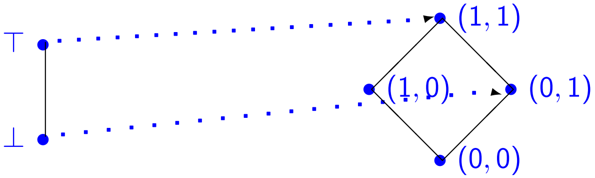
g

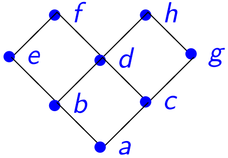




Example: Monotone Function Space

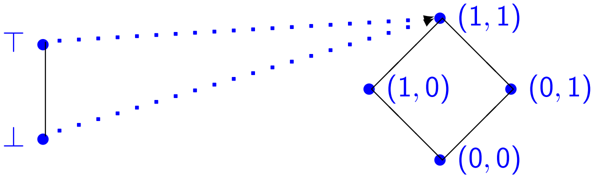
h

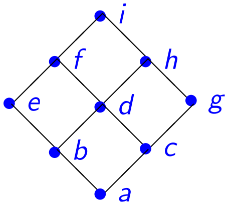




Example: Monotone Function Space

i





Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, l, K, E) where

- N is a finite set of nodes
- $l : N \rightarrow \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that

Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, l, K, E) where

- N is a finite set of nodes
- $l : N \to \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
  - for all m, n, $n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$

Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, l, K, E) where

- N is a finite set of nodes
- $l : N \to \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
  - for all $m,\ n,\ n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$
  - if $m \in N$ and $l(m) = \text{Exit}$ then $|\{n \,|\, \exists k \in K.\ \exists m \in N.\,(m, k, n) \in E\}| = 0$

Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, l, K, E) where

- N is a finite set of nodes
- $l : N \to \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
  - for all m, n, $n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$
  - if $m \in N$ and $l(m) = \text{Exit}$ then $|\{n \mid \exists k \in K.\ \exists m \in N.\,(m, k, n) \in E\}| = 0$
  - if $m \in N$ and $l(m) = \text{Entry}$ or $l(m) = i := e$ for some identifier i and expression e, then

Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, I, K, E) where

- N is a finite set of nodes
- $I : N \to \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
  - for all m, n, $n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$
  - if $m \in N$ and $I(m) = \text{Exit}$ then $|\{n \,|\, \exists k \in K.\ \exists m \in N.\,(m, k, n) \in E\}| = 0$
  - if $m \in N$ and $I(m) = \text{Entry}$ or $I(m) = i := e$ for some identifier i and expression e, then
    - $|\{n \,|\, \exists k \in K.\ \exists m \in N.\,(m, k, n) \in E\}| = 1$

Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, I, K, E) where

- N is a finite set of nodes
- $l : N \to \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
  - for all $m,\ n,\ n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$
  - if $m \in N$ and $l(m) = \text{Exit}$ then $|\{n \mid \exists k \in K.\ \exists m \in N.\, (m, k, n) \in E\}| = 0$
  - if $m \in N$ and $l(m) = \text{Entry}$ or $l(m) = i := e$ for some identifier i and expression e, then
    - $|\{n \mid \exists k \in K.\ \exists m \in N.\, (m, k, n) \in E\}| = 1$
    - for all $k \in K$ and $m \in N$, if $(m, k, n) \in E$ then $k = \text{seq}$

Control-Flow Graphs

A Control-Flow Graph (for a SIMPL-like language) is a tuple (N, l, K, E) where

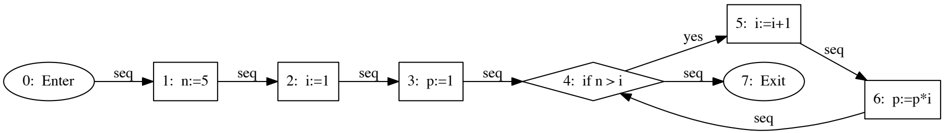
- N is a finite set of nodes
- $l : N \to \{\text{Entry}, \text{Exit}, \text{i:=e}, \text{if b}, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
  - for all m, n, $n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$
  - if $m \in N$ and $l(m) = \text{Exit}$ then $|\{n \,|\, \exists k \in K.\ \exists m \in N.\, (m, k, n) \in E\}| = 0$
  - if $m \in N$ and $l(m) = \text{Entry}$ or $l(m) = i := e$ for some identifier i and expression e, then
    - $|\{n \,|\, \exists k \in K.\ \exists m \in N.\, (m, k, n) \in E\}| = 1$
    - for all $k \in K$ and $m \in N$, if $(m, k, n) \in E$ then $k = \text{seq}$
  - if $m \in N$ and $l(m) = \text{if } b$ for some boolean expression b, then $|\{n \,|\, \exists k \in K.\, (m, k, n) \in E\}| = 2$

Example

n:=5; i:=1; p:=1; while n>i do i:=i+1; p:=p*i od

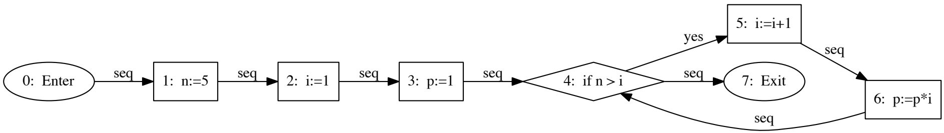
Example

n:=5; i:=1; p:=1; while n>i do i:=i+1; p:=p*i od



Example

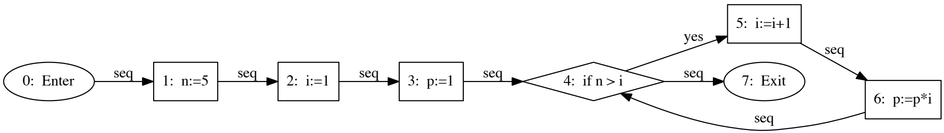
n:=5; i:=1; p:=1; while n>i do i:=i+1; p:=p*i od



- $N = \{0, 1, 2, 3, 4, 5, 6, 7\}$

Example

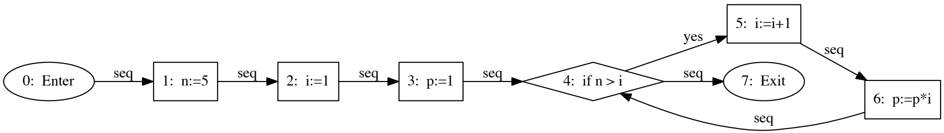
n:=5; i:=1; p:=1; while n>i do i:=i+1; p:=p*i od



- $N = \{0, 1, 2, 3, 4, 5, 6, 7\}$
- $I =$
$$\left\{\begin{array}{llll} 0 \mapsto \texttt{Enter}, & 1 \mapsto \texttt{n:=5}, & 2 \mapsto \texttt{i:=1}, & 3 \mapsto \texttt{p:=1}, \\ 4 \mapsto \texttt{if n>1}, & 5 \mapsto \texttt{i:=i+1}, & 6 \mapsto \texttt{p:=p:=p*i}, & 7 \mapsto \texttt{Exit} \end{array}\right\}$$

Example

n:=5; i:=1; p:=1; while n>i do i:=i+1; p:=p*i od



- $N = \{0, 1, 2, 3, 4, 5, 6, 7\}$
- $I =$
$$\left\{ \begin{array}{llll} 0 \mapsto \texttt{Enter}, & 1 \mapsto \texttt{n:=5}, & 2 \mapsto \texttt{i:=1}, & 3 \mapsto \texttt{p:=1}, \\ 4 \mapsto \texttt{if n>1}, & 5 \mapsto \texttt{i:=i+1}, & 6 \mapsto \texttt{p:=p:=p*i}, & 7 \mapsto \texttt{Exit} \end{array} \right\}$$
- $E = \left\{ \begin{array}{llll} (0, \texttt{seq}, 1), & (1, \texttt{seq}, 2), & (3, \texttt{seq}, 4) & (2, \texttt{seq}, 3) \\ (4, \texttt{yes}, 5), & (4, \texttt{seq}, 7), & (5, \texttt{seq}, 6), & (6, \texttt{seq}, 7) \end{array} \right\}$

Abstract Interpretation

- Let (N, I, K, E) be a control flow graph.
- An abstract interpretation of control flow graphs is a pair $(A, \mathcal{I})$ where
  - A is a complete latice and
  - $\mathcal{I} : ((E \to A) \times E) \to A$ (think *next state information vector*)
  - for all $f, g \in (E \to A)$, for all $e \in E$, if $f \le g$ then $\mathcal{I}(f, e) \le \mathcal{I}(g, e)$

Abstract Semantics

- Can define $\overline{\mathcal{I}} : (E \to A) \to (E \to A)$ by $\overline{\mathcal{I}}(f)(e) = \mathcal{I}(f, e)$
- **Fact:** $\overline{\mathcal{I}}$ is order-preserving
- **Tarski's Fixed-Point Theorem:** If A is a complete lattice and $f : A \to A$ is order-preserving, then f has both a least and a greatest fixed-point (may or may not be the same).
- **Fact:** There exist $c : E \to A$ such that $\overline{\mathcal{I}}(c) = c$, and that c is the least such.
- Write $\mu\overline{\mathcal{I}}$ for the least fixed point of $\overline{\mathcal{I}}$
- $\mu\overline{\mathcal{I}}$ is the abstract semantics of (N, I, K, E) with respect to $(A, \mathcal{I})$.

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 $l(n)$:

Transitions in Control Flow Graphs

- $\text{next\_state} : \mathit{States} \rightarrow \mathit{States}$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq \text{Enter}$

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 $l(n)$:
  - $l(n) \neq \text{Enter}$
  - $l(n) = \text{Exit} \Rightarrow \text{next\_state}((m, k, n), \rho) = ((m, k, n), \rho)$

Transitions in Control Flow Graphs

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

Transitions in Control Flow Graphs

- $\text{next\_state} : \mathit{States} \to \mathit{States}$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ 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, $(n, \text{suc}, p) \in E$.
    - $\text{next\_state}((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto \mathit{val}(e, \rho)])$

Transitions in Control Flow Graphs

- next_state : $States \to States$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ 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

- $\text{next\_state} : \textit{States} \rightarrow \textit{States}$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ 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, $(n, \text{suc}, p) \in E$.
    - $\text{next\_state}((m, k, n), \rho) = ((n, \text{suc}, p), \rho[i \mapsto \textit{val}(e, \rho)])$
  - $l(n) = (\textit{if}\ \ b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
    - if $\textit{bval}(b, \rho) = \bot$ then $\text{next\_state}((m, k, n), \rho) = (\bot, \rho)$

Transitions in Control Flow Graphs

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

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)
    - if $bval(b, \rho) = \bot$ then $\text{next\_state}((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then $\text{next\_state}((m, k, n), \rho) = (\top, \rho)$
    - $bval(b, \rho) = \text{true}$ then $\text{next\_state}((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$

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)
    - if $bval(b, \rho) = \bot$ then $\text{next\_state}((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then $\text{next\_state}((m, k, n), \rho) = (\top, \rho)$
    - $bval(b, \rho) = \text{true}$ then
      $\text{next\_state}((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$
    - $bval(b, \rho) = \text{false}$ then
      $\text{next\_state}((m, k, n), \rho) = ((n, \text{suc}, q), \rho)$

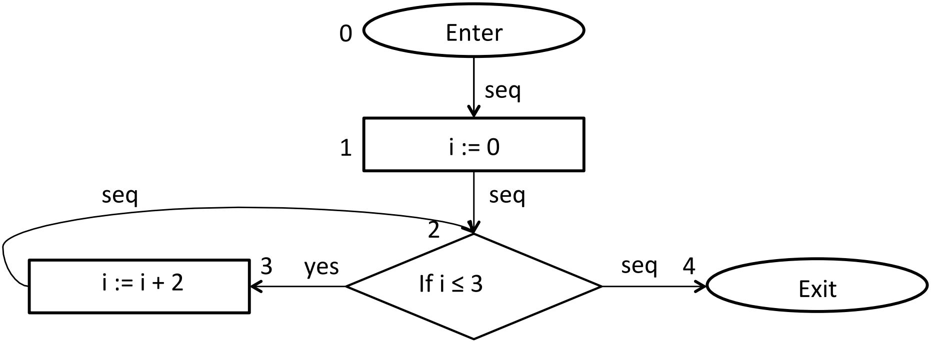
Transitions in Control Flow Graphs

- next_state : $States \to States$
- $\text{next\_state}(\top, \rho) = (\top, \rho)$; $\text{next\_state}(\bot, \rho) = (\bot, \rho)$
- $\text{next\_state}((m, k, n), \rho)$ defined by cases on $l(n)$:
  - $l(n) \neq$ 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)
    - if $bval(b, \rho) = \bot$ then $\text{next\_state}((m, k, n), \rho) = (\bot, \rho)$
    - if $bval(b, \rho) = \top$ then $\text{next\_state}((m, k, n), \rho) = (\top, \rho)$
    - $bval(b, \rho) = \text{true}$ then $\text{next\_state}((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$
    - $bval(b, \rho) = \text{false}$ then $\text{next\_state}((m, k, n), \rho) = ((n, \text{suc}, q), \rho)$
- next_state is transition semantics for control flow graphs

Example

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

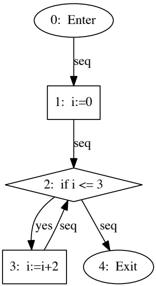
- $Var = \{i\}, \quad values = \mathbb{Z}$
- $N = \{0, 1, 2, 3, 4, 5, 6\}$
- $I(0) = \text{Enter}, \; I(1) = \texttt{i:=0}, \; I(2) = \texttt{if i} \leq \texttt{3},$ $I(3) = \texttt{i:=i+2}, \; I(4) = \text{Exit}$
- $K = \{\text{yes}, \text{seq}\}$
- $E = \left\{ \begin{array}{l} (0, \text{seq}, 1),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



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

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 $\{i \mapsto 4\}(i) = 4 \not\leq 3$
  $\text{next\_state}((3, \text{seq}, 2), \{i \mapsto 4\}) = ((2, \text{seq}, 4), \{i \mapsto 4\})$



Standard Interpretation and Semantics

- Let $Interp(\theta, (m, k, n))$ be the lifting of next_state to sets of environments (contexts)
  - Note: $I(m) \neq \text{Exit}$
  - $I(m) = \text{Enter} \Rightarrow Interp(\theta, (m, k, n)) = \{\{v \mapsto \bot | v \in Var\}\} = \{\lambda v.\bot\}$
  - $I(m) \neq \text{Enter} \Rightarrow$
    $$Interp(\theta, (m, k, n)) = \{\rho \mid \exists m',\ k',\ \rho' \mid (m', k', m) \in E \wedge \rho' \in \theta((m', k', m)) \wedge \text{next\_state}((m', k', m), \rho') = ((m, k, n), \rho)\}$$
- 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

Standard Interpretation and Semantics

- Let $Contexts = \mathcal{P}(Env)$
  - $Contexts$ is a complete lattice
  - A context corresponds to a formula in prediacte logic over the program variables
- 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 $\overline{Interp} : (E \to Contexts) \to (E \to Contexts)$
- $\mu\,\overline{Interp}$ tells us the best knowledge we can know statically about our program

Example: *Interp*

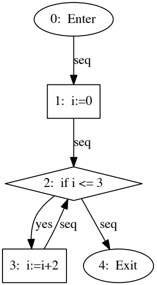
Let θ map edges to sets of environments. *Interp* will tell us the set of environments next_state will associate with each edge assuming θ gives a set of (possibly) possible environments for each predecessor edge:

- Since $Var = \{i\}$, $Interp(\theta, (0, \mathrm{seq}, 1)) = \{\{i \mapsto \bot\}\}$
- If $\theta(e) = \{\,\}$ then $Interp(\theta, e) = \{\,\}$, so assume $\theta(e) \neq \{\,\}$
- $Interp(\theta, (1, \mathrm{seq}, 2))$
  $= \{\rho \mid \exists \rho' \in \theta((0, \mathrm{seq}, 1)) \,|\, \rho = \rho'[i \mapsto 0]\} = \{\{i \mapsto 0\}\}$
- $Interp(\theta, (2, \mathrm{yes}, 3)) = \{\rho \in \theta(1, \mathrm{seq}, 2) \cup \theta(3, \mathrm{seq}, 2) \mid \rho(i) \leq 3\}$
- $Interp(\theta, (3, \mathrm{seq}, 2)) = \{\rho \mid \exists \rho' \in \theta(2, \mathrm{yes}, 3) \,|\, \rho = \rho`[i \mapsto \rho'(i) + 2]\}$
- $Interp(\theta, (2, \mathrm{no}, 4)) = \{\rho \in \theta(3, \mathrm{seq}, 2) \mid \rho(i) > 3\}$

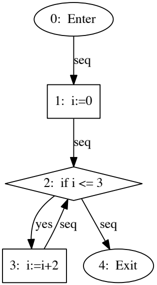
- $\overline{Interp}(\theta)(e) = Interp(\theta, e)$
- $\overline{Interp}^0(\theta)(e) = \{\,\}$ $\quad \overline{Interp}^{n+1}(\theta)(e) = \overline{Interp}(\overline{Interp}^n(\theta))(e)$

Example: $\mu\overline{Interp}$

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

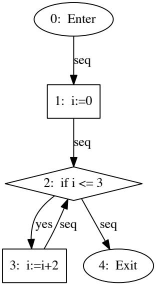


- $\mu\overline{Inter} : E \rightarrow Contexts = \mathcal{P}(Env)$
- Start with minimal θ_0 assigning no environments to any edge: $\theta_0(e) = \{\,\}$
- $\mu\overline{Inter}p(e) = \bigcup_{n\in\mathbb{N}} \overline{Interp}^n(e)$
- $\mu\overline{Inter}p(0, \text{seq}, 1) = \{\{i \mapsto \bot\}\}$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



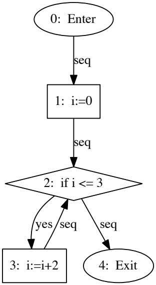
Example: $\mu\overline{Interp}$

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



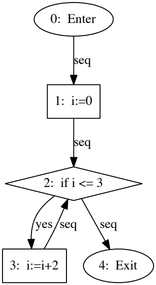
Example: $\mu\overline{Inter}p$

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



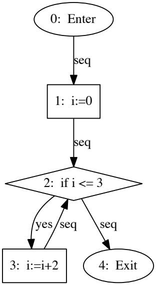
Example: $\mu\overline{Interp}$

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



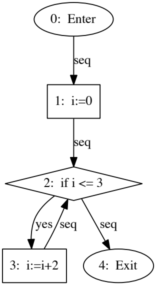
Example: $\mu\overline{Inter}p$

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



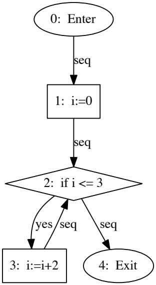
Example: $\mu\overline{Interp}$

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



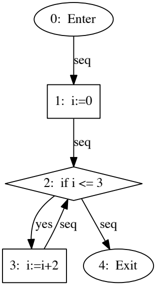
Example: $\mu\overline{Interp}$

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



Example: $\mu\overline{Interp}$

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



Soundness of Abstract Semantics

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

- $\alpha : \mathit{Contexts} \rightarrow A$, $\beta : A \rightarrow \mathit{Contexts}$
- $\alpha,\ \beta$ order preserving
- For all $a \in A$ have $\alpha(\beta(a)) = a$
- For all $S \in \mathit{Contex}$, have $S \subseteq \beta(\alpha(S))$
- For all $e \in E$, $\alpha(\mu\overline{\mathit{Interp}}(e)) = \mu\overline{\mathcal{I}}(e)$
    - The abtract interpretation gives us more possibilities, is less precise