

CS477 Formal Software Dev Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu

<http://courses.engr.illinois.edu/cs477>

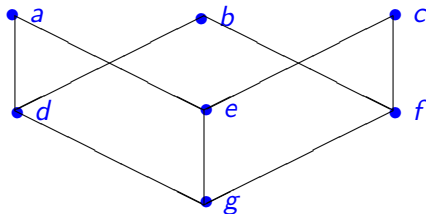
Slides based in part on previous lectures
by Mahesh Vishwanathan, and by Gul Agha

April 20, 2018

Partial Orders

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

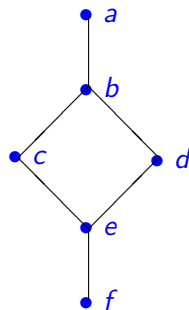
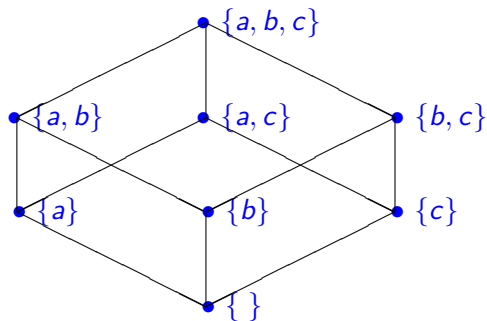
- **[Refl]** $s \leq s$ for all $s \in S$
- **[Antisym]** $s \leq t$ and $t \leq s$ implies $s = t$, for all $s, t \in S$
- **[Trans]** $s \leq t$ and $t \leq u$ implies $s \leq u$, for all $s, t, \in S$



Upper Bounds and Complete Lattices

- 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 $\text{lub}(X)$ 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 $\perp = \bigvee \{ \}$ and $\top = \bigvee S$

Example Complete Lattices

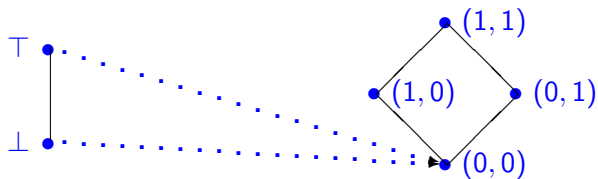


Partial Orders, Functions, and Complete Lattices

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

Example: Monotone Function Space

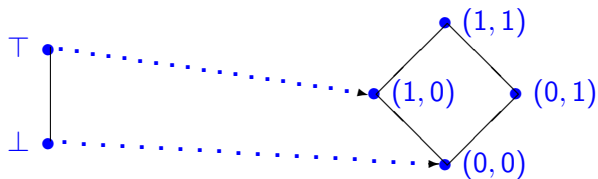
a



• a

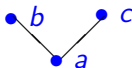
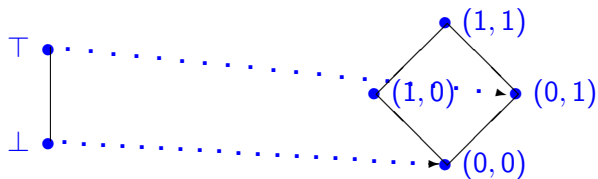
Example: Monotone Function Space

b



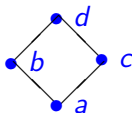
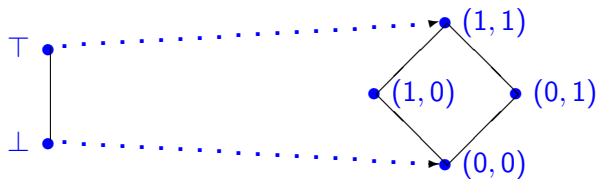
Example: Monotone Function Space

c

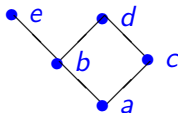
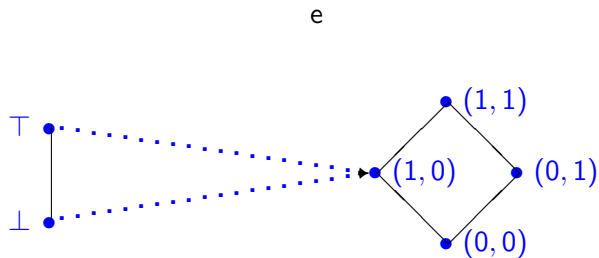


Example: Monotone Function Space

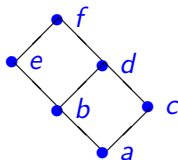
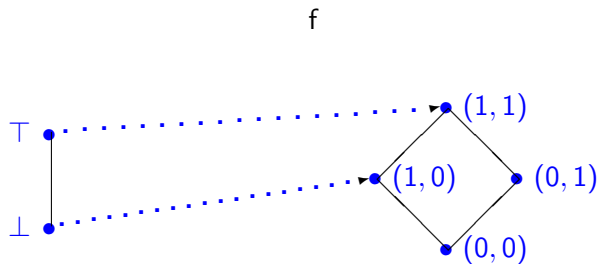
d



Example: Monotone Function Space

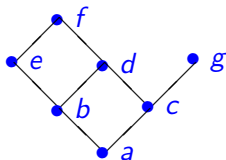
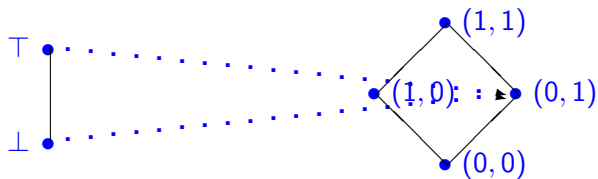


Example: Monotone Function Space

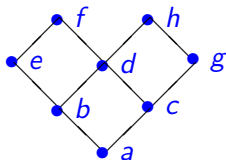
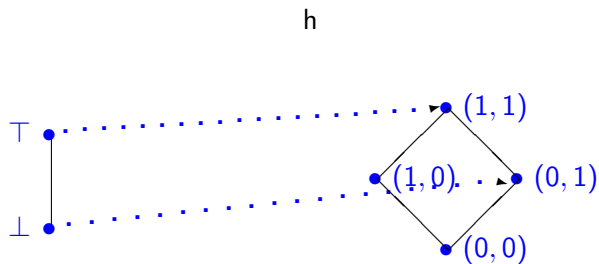


Example: Monotone Function Space

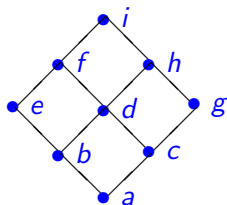
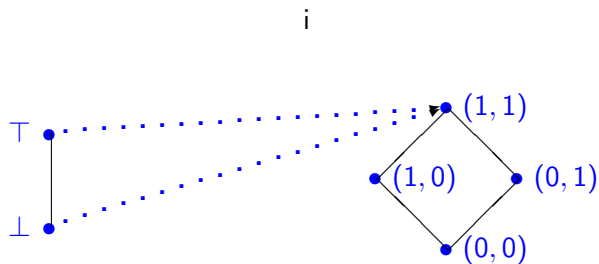
g



Example: Monotone Function Space



Example: Monotone Function Space



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
- $I : N \rightarrow \{\text{Entry, Exit, } i:=e, \text{ if } b, \}$
- $K = \{\text{yes, seq}\}$
- $E \subseteq N \times K \times N$ such that

Control-Flow Graphs

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

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

- N is a finite set of nodes
- $I : N \rightarrow \{\text{Entry}, \text{Exit}, i:=e, \text{if } b, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ 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 $|\{n \mid \exists k \in K. \exists m \in N. (m, k, n) \in E\}| = 0$

Control-Flow Graphs

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

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

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
- $I : N \rightarrow \{\text{Entry}, \text{Exit}, i := e, \text{if } b, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ 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 $|\{n \mid \exists k \in K. \exists m \in N. (m, k, n) \in E\}| = 0$
 - if $m \in N$ and $I(m) = \text{Entry}$ or $I(m) = i := e$ for some identifier i and expression e , then
 - $|\{n \mid \exists k \in K. \exists m \in N. (m, k, n) \in E\}| = 1$

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
- $I : N \rightarrow \{\text{Entry}, \text{Exit}, i := e, \text{if } b, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ 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 $|\{n \mid \exists k \in K. \exists m \in N. (m, k, n) \in E\}| = 0$
 - if $m \in N$ and $I(m) = \text{Entry}$ or $I(m) = i := e$ for some identifier i and expression e , then
 - $|\{n \mid \exists k \in K. \exists m \in N. (m, k, n) \in E\}| = 1$
 - 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, I, K, E) where

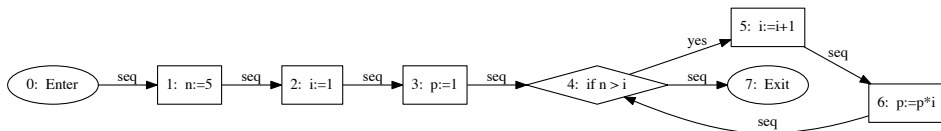
- N is a finite set of nodes
- $I : N \rightarrow \{\text{Entry}, \text{Exit}, i := e, \text{if } b, \}$
- $K = \{\text{yes}, \text{seq}\}$
- $E \subseteq N \times K \times N$ such that
 - for all $m, n, n' \in N$ and $k \in K$, if $(m, k, n) \in E$ and $(m, k, n') \in E$ then $n = n'$
 - if $m \in N$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

Example

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

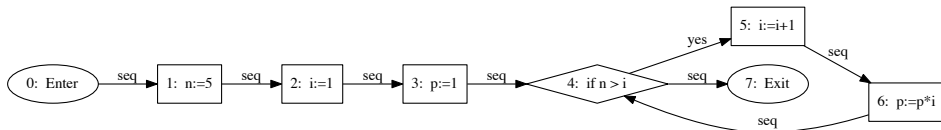
Example

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



Example

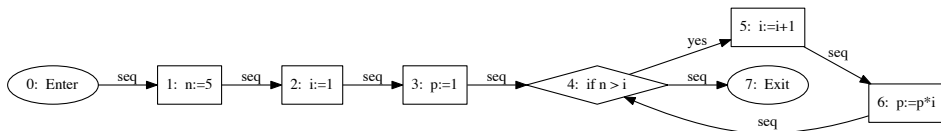
`n:=5; i:=1; p:=1; while n>i do i:=i+1; p:=p*i od`



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

Example

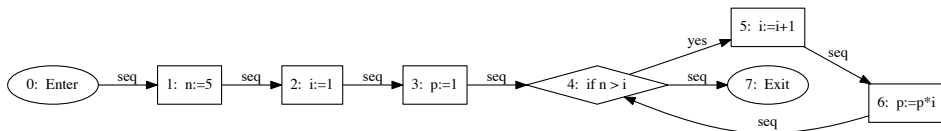
$n:=5; i:=1; p:=1; \text{while } n>i \text{ do } i:=i+1; p:=p*i \text{ od}$



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

Example

$n:=5; i:=1; p:=1; \text{while } n>i \text{ do } i:=i+1; p:=p*i \text{ od}$



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

Abstract Interpretation

- Let (N, l, 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 lattice and
 - $\mathcal{I} : ((E \rightarrow A) \times E) \rightarrow A$ (think *next state information vector*)
 - for all $f, g \in (E \rightarrow A)$, for all $e \in E$, if $f \leq g$ then $\mathcal{I}(f, e) \leq \mathcal{I}(g, e)$

- Can define $\bar{\mathcal{I}} : (E \rightarrow A) \rightarrow (E \rightarrow A)$ by $\bar{\mathcal{I}}(f)(e) = \mathcal{I}(f, e)$
- **Fact:** $\bar{\mathcal{I}}$ is order-preserving
- **Tarski's Fixed-Point Theorem:** If A is a complete lattice and $f : A \rightarrow A$ is order-preserving, then f has both a least and a greatest fixed-point (may or may not be the same).
- **Fact:** There exist $c : E \rightarrow A$ such that $\bar{\mathcal{I}}(c) = c$, and that c is the least such.
- Write $\mu\bar{\mathcal{I}}$ for the least fixed point of $\bar{\mathcal{I}}$
- $\mu\bar{\mathcal{I}}$ is the **abstract semantics** of (N, l, 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 $Val = values \cup \{\top, \perp\}$, the extended set of values, ordered as before
 - Val is a complete lattice.
- Let $Env = \{\rho \mid \rho : Var \rightarrow Val\}$
 - Env is a complete lattice
 - An env used to be a partial function; now map undefined to \perp
 - $val : (Exp \times Env) \rightarrow Val$
 - Will assume $\{\text{true}, \text{false}\} \subseteq values$
 - $bval : (BExp \times Env) \rightarrow \{\text{true}, \text{false}\} \cup \{\top, \perp\} \subseteq Val$
- Let $States = (E \cup \{\top, \perp\}) \times Env$
 - $States$ 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

- $\text{next_state} : \text{States} \rightarrow \text{States}$
- $\text{next_state}(\top, \rho) = (\top, \rho)$; $\text{next_state}(\perp, \rho) = (\perp, \rho)$
- $\text{next_state}((m, k, n), \rho)$ defined by cases on $l(n)$:

Transitions in Control Flow Graphs

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

Transitions in Control Flow Graphs

- $\text{next_state} : \text{States} \rightarrow \text{States}$
- $\text{next_state}(\top, \rho) = (\top, \rho)$; $\text{next_state}(\perp, \rho) = (\perp, \rho)$
- $\text{next_state}((m, k, n), \rho)$ 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

- $\text{next_state} : \text{States} \rightarrow \text{States}$
- $\text{next_state}(\top, \rho) = (\top, \rho)$; $\text{next_state}(\perp, \rho) = (\perp, \rho)$
- $\text{next_state}((m, k, n), \rho)$ 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 ,
 $(n, \text{succ}, p) \in E$.

Transitions in Control Flow Graphs

- $\text{next_state} : \text{States} \rightarrow \text{States}$
- $\text{next_state}(\top, \rho) = (\top, \rho)$; $\text{next_state}(\perp, \rho) = (\perp, \rho)$
- $\text{next_state}((m, k, n), \rho)$ 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 , $(n, \text{succ}, p) \in E$.
 - $\text{next_state}((m, k, n), \rho) = ((n, \text{succ}, p), \rho[i \mapsto \text{val}(e, \rho)])$

Transitions in Control Flow Graphs

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

Transitions in Control Flow Graphs

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

Transitions in Control Flow Graphs

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

Transitions in Control Flow Graphs

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

Transitions in Control Flow Graphs

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

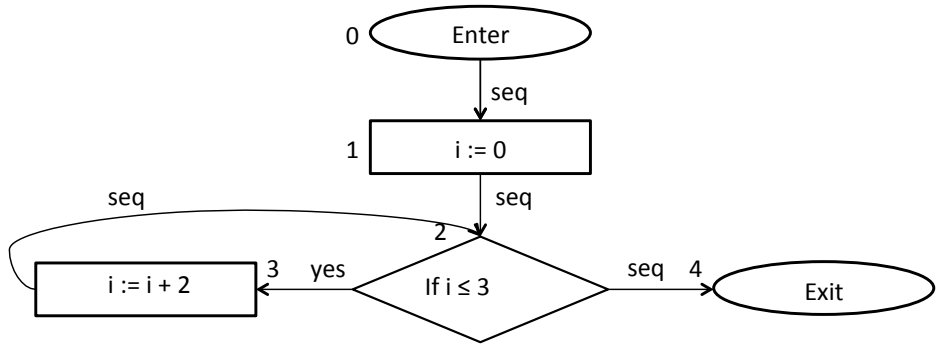
Transitions in Control Flow Graphs

- $\text{next_state} : States \rightarrow States$
- $\text{next_state}(\top, \rho) = (\top, \rho)$; $\text{next_state}(\perp, \rho) = (\perp, \rho)$
- $\text{next_state}((m, k, n), \rho)$ 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 ,
 $(n, \text{succ}, p) \in E$.
 - $\text{next_state}((m, k, n), \rho) = ((n, \text{succ}, p), \rho[i \mapsto \text{val}(e, \rho)])$
 - $l(n) = (\text{if } b)$, then n has two out arcs: (n, yes, p) and (n, seq, q)
 - if $\text{bval}(b, \rho) = \perp$ then $\text{next_state}((m, k, n), \rho) = (\perp, \rho)$
 - if $\text{bval}(b, \rho) = \top$ then $\text{next_state}((m, k, n), \rho) = (\top, \rho)$
 - $\text{bval}(b, \rho) = \text{true}$ then
 $\text{next_state}((m, k, n), \rho) = ((n, \text{yes}, p), \rho)$
 - $\text{bval}(b, \rho) = \text{false}$ then
 $\text{next_state}((m, k, n), \rho) = ((n, \text{seq}, q), \rho)$
- next_state is transition semantics for control flow graphs

Example

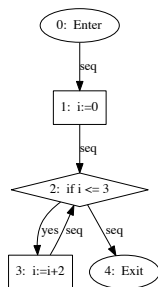
Consider the following control flow graph (N, I, K, E) where:

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



Example: next_state

- $\text{next_state}((0, \text{seq}, 1), \{i \mapsto \perp\}) = ((1, \text{seq}, 2), \{i \mapsto 0\})$
- $\text{next_state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\})$
- $\text{next_state}((2, \text{yes}, 3), \{i \mapsto 0\}) =$
 $((3, \text{seq}, 2), \{i \mapsto 0\}[i \mapsto 0 + 2]) = ((3, \text{seq}, 2), \{i \mapsto 2\})$
- 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: $I(m) \neq \text{Exit}$
 - $I(m) = \text{Enter} \Rightarrow Interp(\theta, (m, k, n)) = \{\{v \mapsto \perp \mid v \in \text{Var}\}\} = \{\lambda v. \perp\}$
 - $I(m) \neq \text{Enter} \Rightarrow$
 $Interp(\theta, (m, k, n)) =$
 $\{\rho \mid \exists m', k', \rho' \mid (m', k', m) \in E \wedge$
 $\rho' \in \theta((m', k', m)) \wedge$
 $\text{next_state}((m', k', m), \rho') = ((m, k, n), \rho)\}$
- 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 predicate 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 abstract interpretation
- Recall: $Interp : ((E \rightarrow Contexts) \times E) \rightarrow Contexts$ so $\overline{Interp} : (E \rightarrow Contexts) \rightarrow (E \rightarrow Contexts)$
- $\mu \overline{Interp}$ tells us the best knowledge we can know **statically** about our program

Example: *Interp*

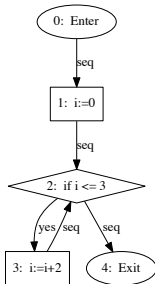
Let θ map edges to sets of environments. *Interp* will tell us the set of environments `next_state` will associate with each edge assuming θ gives a set of (possibly) possible environments for each predecessor edge:

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

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

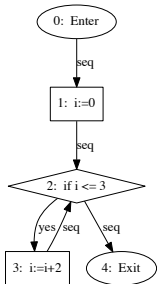
Example: $\overline{\mu\text{Interp}}$

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



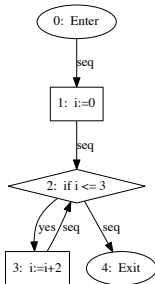
Example: $\overline{\mu\text{Interp}}$

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



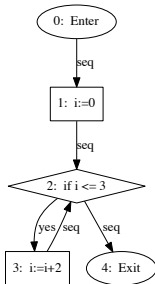
Example: $\overline{\mu\text{Interp}}$

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



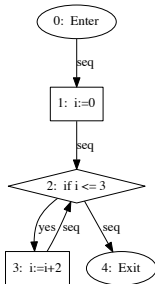
Example: $\overline{\mu\text{Interp}}$

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



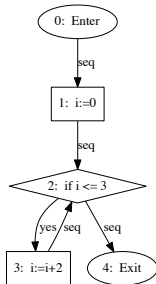
Example: $\overline{\mu\text{Interp}}$

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



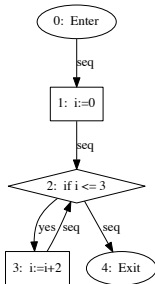
Example: $\overline{\mu\text{Interp}}$

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



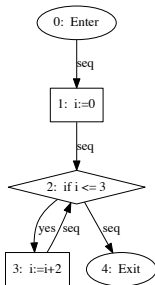
Example: $\overline{\mu\text{Interp}}$

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



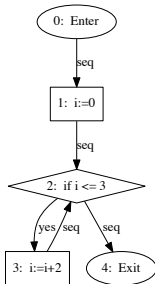
Example: $\overline{\mu\text{Interp}}$

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



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

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



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 α, β such that

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