CS477 Formal Software Development 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

May 7, 2014
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 $s, t, u \in S$
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 \(\text{lub}(X)\) or \(\bigvee X\) for the least upper bound of \(X\).
- Write \(x \lor y\) for \(\bigvee\{x, y\}\)
- **Note:** \(x \lor y = x \iff y \leq x\)
- **Note:** Given a set \(S\), \((\mathcal{P}(S), \subseteq)\) is a complete lattice.
- Write \(\bot = \bigvee\{\}\) and \(\top = \bigvee S\)
Example Complete Lattices
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 $f \leq_{\text{fun}} g$ to mean that for all $x \in X$ we have $f(x) \leq g(x)$
- Will leave off the subscript in general

Fact: $(\{ f \mid f : X \rightarrow B \}, \leq_{\text{fun}})$ is a partial order.

Fact: $(\{ f \mid f : X \rightarrow B \}, \leq_{\text{fun}})$ is a complete lattice if $B$ is.

Fact: $(\{ f \mid f : A \rightarrow B, \ f \text{ order-preserving} \} \leq_{\text{fun}})$ is a complete lattice if $B$ is.
A **Control-Flow Graph** (for a SIMPL-like language) is a tuple \((N, l, K, E)\) where

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

\[ n := 5; \ i := 1; \ p := 1; \text{ while } n > i \text{ do } i := i + 1; \ p := p \times i \text{ od} \]
Let \((N, I, K, E)\) be a control flow graph.

An abstract interpretation of control flow graphs is a pair \((A, I)\) where

- \(A\) is a complete lattice and
- \(I : ((E \rightarrow A) \times E) \rightarrow A\) (think next state information vector)
- for all \(f, g \in (E \rightarrow A)\), for all \(e \in E\), if \(f \leq g\) then \(I(f, e) \leq I(g, e)\)
Abstract Semantics

- Can define \( \overline{I} : (E \rightarrow A) \rightarrow (E \rightarrow A) \) by \( \overline{I}(f)(e) = I(f, e) \)
- **Fact:** \( \overline{I} \) 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 \( \overline{I}(c) = c \), and that \( c \) is the least such.
- Write \( \mu \overline{I} \) for the least fixed point of \( \overline{I} \)
- \( \mu \overline{I} \) is the abstract semantics of \( (N, l, K, E) \) with respect to \( (A, I) \).
Given \((N, l, K, E)\) a control flow graph with labels using variables from \(Var\).

Let \(Val = \text{values} \cup \{\top, \bot\}\), 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
  \(((e, \rho) \leq (e', \rho')) \equiv ((e \leq e') \land (\rho \leq \rho'))\).
Transitions in Control Flow Graphs

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

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

seq

1

i := 0

seq

2

if i ≤ 3

seq

Exit

0

i := i + 2

3

yes

4
Example: next_state

- \( \text{next}_\text{state}((0, \text{seq}, 1), \{i \mapsto \bot\}) = ((1, \text{seq}, 2), \{i \mapsto 0\}) \)
- \( \text{next}_\text{state}((1, \text{seq}, 2), \{i \mapsto 0\}) = ((2, \text{yes}, 3), \{i \mapsto 0\}) \)
- \( \text{next}_\text{state}((2, \text{yes}, 3), \{i \mapsto 0\}) = ((3, \text{seq}, 2), \{i \mapsto 0, i \mapsto 2 + 2\}) = ((3, \text{seq}, 2), \{i \mapsto 2\}) \)
- Since \( \{i \mapsto 2\}(i) = 2 \leq 3 \)
  \( \text{next}_\text{state}((3, \text{seq}, 2), \{i \mapsto 2\}) = ((2, \text{yes}, 3), \{i \mapsto 2\}) \)
- \( \text{next}_\text{state}((2, \text{yes}, 3), \{i \mapsto 2\}) = ((3, \text{seq}, 2), \{i \mapsto 2, i \mapsto 2 + 2\}) = ((3, \text{seq}, 2), \{i \mapsto 4\}) \)
- Since \( \{i \mapsto 4\}(i) = 4 \not\leq 3 \)
  \( \text{next}_\text{state}((3, \text{seq}, 2), \{i \mapsto 4\}) = ((2, \text{seq}, 4), \{i \mapsto 4\}) \)
Standard Interpretation and Semantics

- Let $\text{Interp}(\theta, (m, k, n))$ be the lifting of next_state to sets of environments (contexts)
  - Note: $l(m) \neq \text{Exit}$
  - $l(m) = \text{Enter} \implies \text{Interp}(\theta, (m, k, n)) = \{\{v \mapsto \bot | v \in \text{Var}\} = \{\lambda v. \bot\}$
  - $l(m) \neq \text{Enter} \implies$
    $\text{Interp}(\theta, (m, k, n)) = \{\rho | \exists m', k', \rho' | (m', k', m) \in E \land \rho' \in \theta((m', k', m)) \land \text{next_state}((m', k', m), \rho') = ((m, k, n), \rho)\}$
- If $\theta$ tells all the environments we might come into our edge with, $\text{Interp}(\theta, (m, k, n))$ tells us the set of environemnts we may leave with
Let $\text{Contexts} = \mathcal{P}(\text{Env})$

- $\text{Contexts}$ 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 $\text{Interp}(\theta, e') \subseteq \text{Interp}(\phi, e')$

**Result:** $(\text{Contexts}, \text{Interp})$ is an abstract interpretation

Recall: $\text{Interp} : ((E \rightarrow \text{Contexts}) \times E) \rightarrow \text{Contexts}$ so $\text{Interp} : (E \rightarrow \text{Contexts}) \rightarrow (E \rightarrow \text{Contexts})$

$\mu \text{Interp}$ tells us the best knowledge we can know statically about our program
Example: *Interp*

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

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

- $\text{Interp}(\theta)(e) = \text{Interp}(\theta, e)$
- $\text{Interp}^0(\theta)(e) = \{ \}$
- $\text{Interp}^{n+1}(\theta)(e) = \text{Interp}(\text{Interp}^n(\theta))(e)$
Example: $\mu\text{Interp}$

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

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

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

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

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

Elsa L Gunter  
CS477 Formal Software Development Method
Example: \( \mu \text{Interp} \)

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

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

- $\mu \text{Inter} : E \rightarrow \text{Contexts} = \mathcal{P}(\text{Env})$

- Start with minimal $\theta_0$ assigning no environments to any edge:
  $$\theta_0(e) = \{\}$$

- $\mu \text{Interp}(e) = \bigcup_{n \in \mathbb{N}} \text{Interp}^n(e)$

- $\mu \text{Interp}(0, \text{seq}, 1) = \{\{i \mapsto \perp\}\}$

- $\mu \text{Interp}(1, \text{seq}, 2) = \{\{i \mapsto 0\}\}$

- $\mu \text{Interp}(2, \text{yes}, 3) = \{\{i \mapsto 0\}, \{i \mapsto 2\}\}$

- $\mu \text{Interp}(3, \text{seq}, 2) = \{\{i \mapsto 2\}, \{i \mapsto 4\}\}$

- $\mu \text{Interp}((2, \text{no}, 4)) = \{\{i \mapsto 4\}\}$

![Flowchart](image-url)
Example: $\mu \text{Interp}$

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

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

- \(\alpha : \text{Contexts} \rightarrow A, \beta : A \rightarrow \text{Contexts}\)
- \(\alpha, \beta\) order preserving
- For all \(a \in A\) have \(\alpha(\beta(a)) = a\)
- For all \(S \in \text{Context}\), have \(S \subseteq \beta(\alpha(S))\)
- For all \(e \in E\), \(\alpha(\mu\text{Interp}(e)) = \mu\mathcal{I}(e)\)

- The abstract interpretation gives us more possibilities, is less precise