Course Recap

• What we have discussed so far

Operational Semantics

• How will a given program behave on a given input?
• The ground truth for any analysis

Types

• Annotations describe properties of the data that can be referred by a variable
• Easy to describe properties that are global to the execution, but only one variable at a time
  • (at least with the machinery we have seen here)
• Properties are fixed a priori by the type system designer
• Actual analysis is cheap
• Annotations can often be inferred

Program Logics

• Annotations describe properties of the state at a given point in the program
• Easy to describe complex properties of the overall program state, but messy to describe properties that hold over time
• Logic provides a rich language for properties
• Actual analysis can be expensive
• Annotations are hard to infer
Abstract Interpretation

\[ W = \text{wp}(\text{while } b \text{ do } c, Q) \]

- Recall: computing the wp of loop requires solving recursive equation
  \[ W = (b \Rightarrow \text{wp}(c, W) \land \neg b \Rightarrow Q) \]
- \( W \) is the (greatest) fixpoint of the recursive equation
- Abstract Interpretation: interpret a program over an abstract domain
- Find the best possible fixpoint

Patrick Cousot, Radhia Cousot:
“Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”,
(POPL 1977), pp. 238-252
• Consider the following language with only integers and multiplications

\[ e ::= n \mid e_1 \times e_2 \]

• Operational semantics of this language:

\[
\begin{align*}
& n \Downarrow n \\
& e_1 \Downarrow n_1 \\
& e_2 \Downarrow n_2 \\
& n_3 = n_1 \times n_2 \\
& e_1 \times e_2 \Downarrow n_3
\end{align*}
\]

• Take the operational semantics as the “ground truth”

• Using operational semantics rules we can define \( eval : e \rightarrow \mathbb{Z} \)

  • \( eval(n) = n \)
  
  • \( eval(e_1 \times e_2) = eval(e_1) \times eval(e_2) \)

• For this language the precise semantics is computable

  • but in general it is not
• Assume that we are interested not in the value of the expression, but only in its sign
  • positive (+), negative (-), or zero (0)
• We can define an abstract semantics that computes only the sign of the result

$$eval^A : e \rightarrow \{+, -, 0\}$$

• $\{-, 0, +\}$ is the abstract domain

\[
\begin{array}{c|ccc}
\otimes & + & 0 & - \\
\hline
+ & + & 0 & - \\
0 & 0 & 0 & 0 \\
- & - & 0 & + \\
\end{array}
\]

$$eval^A(n) = \text{sign}(n)$$

$$eval^A(e_1 \times e_2) = eval^A(e_1) \otimes eval^A(e_2)$$
Soundness

- We can show that this abstraction is correct in the sense that it correctly predicts the sign of an expression.
- Proof is by structural induction on $e$

\[
\begin{align*}
\text{eval}(e) > 0 & \iff \text{eval}^A(e) = + \\
\text{eval}(e) = 0 & \iff \text{eval}^A(e) = 0 \\
\text{eval}(e) < 0 & \iff \text{eval}^A(e) = -
\end{align*}
\]
• Associate each abstract value with the set of concrete values it represents
• $\gamma$ is called the **concretization** function

\[
\gamma : \{-, 0, +\} \rightarrow 2^\mathbb{Z}
\]

\[
\gamma(+) = \{ n \in \mathbb{Z} \mid n > 0 \}
\]
\[
\gamma(0) = \{0\}
\]
\[
\gamma(-) = \{ n \in \mathbb{Z} \mid n < 0 \}
\]
Another View of Soundness

- Soundness can be stated succinctly

\[ \forall e. \, eval(e) \in \gamma(eval^A(e)) \]

- The real value of the expression is among the concrete values represented by the abstract value of the expression
• Extend our language with unary —

\[
\begin{align*}
  e \downarrow n \\
  -e \downarrow -n
\end{align*}
\]

\[
eval^A(-e) = \ominus \eval^A(e)
\]
Adding +

- Adding addition is not so easy
- The abstract values are not closed under addition

\[ e_1 \downarrow n_1 \quad e_2 \downarrow n_2 \quad n_3 = n_1 + n_2 \]
\[ e_1 + e_2 \downarrow n_3 \]

\[ \text{eval}^A(e_1 + e_2) = \text{eval}^A(e_1) \oplus \text{eval}^A(e_2) \]

<table>
<thead>
<tr>
<th>( \oplus )</th>
<th>+</th>
<th>0</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>+</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>?</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Solution

- We need another abstract value to represent a result that can be any integer.
- Finding a domain closed under all the abstract operations is often a key design problem.

\[ \gamma(T) = \mathbb{Z} \]
We also need to extend the other abstract operations to work with $\top$.

<table>
<thead>
<tr>
<th>$\otimes$</th>
<th>$+ \top 0$</th>
<th>$- \top$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>$+$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$-$</td>
<td>$-$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$\top$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\ominus$</th>
<th>$+$</th>
<th>$0$</th>
<th>$-$</th>
<th>$\top$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-$</td>
<td>$-$</td>
<td>$0$</td>
<td>$+$</td>
<td>$\top$</td>
</tr>
</tbody>
</table>
Exercise

- We know \((1 + 2) + -3 \Downarrow 0\)
- Compute \(eval^A((1 + 2) + -3)\)
Exercise

• We know $(1 + 2) + -3 \iff 0$
• Compute $eval^A((1 + 2) + -3)$

$$eval^A((1 + 2) + -3) = (eval^A(1) \oplus eval^A(2)) \oplus eval^A(-3) = (+ \oplus +) \oplus - = T$$
Loss of Precision

- Abstract computation may lose information

\[(1 + 2) + -3 \downarrow 0\]

\[\text{eval}^A((1 + 2) + -3) = \top\]

- We lost some precision
- But this will simplify the computation of the abstract answer in cases when the precise answer is not computable
Adding Integer Division

- Adding \( / \) is straightforward except for the case of division by 0
- If we divide each integer in a set by 0, what set of integers results?
  - The empty set

\[
\gamma(\bot) = \{\}
\]

<table>
<thead>
<tr>
<th>( \emptyset )</th>
<th>+</th>
<th>0</th>
<th>-</th>
<th>\top</th>
<th>\bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>\top</td>
<td>\bot</td>
</tr>
<tr>
<td>0</td>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>\top</td>
<td>\bot</td>
</tr>
<tr>
<td>\top</td>
<td>\top</td>
<td>0</td>
<td>\top</td>
<td>\top</td>
<td>\bot</td>
</tr>
<tr>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
<td>\bot</td>
</tr>
</tbody>
</table>
Adding Integer Division

- As before we need to extend the other abstract operations
- In this case, every entry involving bottom is bottom

$$x \uplus \bot = \bot$$

$$x \otimes \bot = \bot$$

$$x \ominus \bot = \bot$$
The Abstract Domain

- Our abstract domain forms a **lattice**
- A partial order \( x \leq y \iff \gamma(x) \subseteq \gamma(y) \)
- \( x \leq y \) means that \( x \) is more precise than \( y \)
- \( \top \) corresponds to all values in the concrete domain
  - (the least information)
- Every finite subset has a least upper bound (lub) & a greatest lower bound (glb)
A relation $\preceq \subseteq D \times D$ on a set $D$ is a **partial order** iff $\preceq$ is

1. Reflexive: $x \preceq x$
2. Anti-symmetric: $x \preceq y$ and $y \preceq x \Rightarrow x = y$
3. Transitive: $x \preceq y$ and $y \preceq z \Rightarrow x \preceq z$

• A set with a partial order is called a **poset**

**Examples:**

• If $S$ is a set then $(2^S, \subseteq)$ is a poset
• $(\mathbb{Z}, \leq)$ is a poset
Hasse Diagram

- $x$ immediate predecessor of $y$: if $x \preceq y$ and there is no $z$ such that $x \preceq z \preceq y$

- Hasse diagram: a directed acyclic graph where the vertices are elements of the set $D$

- There exists an edge $x \rightarrow y$ if $x$ is an immediate predecessor of $y$

Example.

- $x \preceq y$, $y \preceq t$, $z \preceq t$, $x \preceq z$, $x \preceq t$
  
  $x \preceq x$, $y \preceq y$, $z \preceq z$, $t \preceq t$

![Hasse Diagram]

$t$

$y$

$x$

$z$
Exercise

- $D_n = \{\text{all divisors of } n\}$, with $d \preceq d' \iff d \mid d'$
- Draw the Hasse diagram for $D_{12} = \{1, 2, 3, 4, 6, 12\}$
Exercise

- $D_n = \{\text{all divisors of } n\}$, with $d \preceq d' \iff d \mid d'$
- Draw the Hasse diagram for $D_{12} = \{1, 2, 3, 4, 6, 12\}$

$D_{12} = \{1, 2, 3, 4, 6, 12\}$
Total Order

- Partial order: no guarantee that all elements can be compared to each other
- Total order (linear order): If for any two elements $x$ and $y$ at least one of $x \leq y$ or $y \leq x$ is true
- $(\mathbb{N}, \leq)$ is total order
- Hasse diagram is one-track
Subset Bounds

- Let \((X, \preceq)\) be a poset and let \(A \subseteq X\) be any subset of \(X\).
- An element, \(b \in X\), is a **lower bound** of \(A\) iff \(b \preceq a\) for all \(a \in A\).
- An element, \(m \in X\), is an **upper bound** of \(A\) iff \(a \preceq m\) for all \(a \in A\).
- An element, \(b \in X\), is the **greatest lower bound** (glb) of \(A\) iff the set of lower bounds of \(A\) is nonempty and if \(b\) is the greatest element of this set.
- An element, \(m \in X\), is the **least upper bound** (lub) of \(A\) iff the set of upper bounds of \(A\) is nonempty and if \(m\) is the least element of this set.
Exercise

Find lower/upper bounds and glb/lub for these sets: \( \{b, d\}, \{a, c\}, \{d, e, f\} \)
Exercise

Find lower/upper bounds and glb/lub for these sets: \(\{b, d\}, \{a, c\}, \{d, e, f\}\)

\(\{b, d\}\):
- Lower bounds: \(\{b\}\)    glb: \(b\)
- Upper bounds: \(\{d, g\}\)    lub: \(d\) because \(d \preceq g\)

\(\{a, c\}\):
- Lower bounds: \(\emptyset\)
- Upper bounds: \(\{h\}\) lub: \(h\)
Exercise

Find lower/upper bounds and glb/lub for these sets: \( \{b, d\}, \{a, c\}, \{d, e, f\} \)

\( \{b, d\} \):
- Lower bounds: \( \{b\} \)  
glb: \( b \)
- Upper bounds: \( \{d, g\} \)  
lub: \( d \) because \( d \preceq g \)

\( \{a, c\} \):
- Lower bounds: \( \{\} \)  
no glb
- Upper bounds: \( \{h\} \)  
lub: \( h \)
Exercise

Find lower/upper bounds and glb/lub for these sets: \{b, d\}, \{a, c\}, \{d, e, f\}

\{b, d\}:
- Lower bounds: \{b\}  glb: b
- Upper bounds: \{d, g\}  lub: d because \(d \preceq g\)

\{a, c\}:
- Lower bounds: \{\}  no glb
- Upper bounds: \{h\}  lub: h

\{d, e, f\}:
- Lower bounds: \{\}  no glb
- Upper bounds: \{\}  no lub
Lattice

Poset \((D, \preceq)\) is called a lattice if

- For any \(x, y \in D\), \(\{x, y\}\) has a lub, which is denoted as \(x \sqcup y\) (join)
- For any \(x, y \in D\), \(\{x, y\}\) has a glb, which is denoted as \(x \sqcap y\) (meet)

Example.

- For \((2^B, \subseteq)\): \(x \sqcap y = x \cap y\), \(x \sqcup y = x \cup y\)
- For \((\mathbb{Z}, \leq)\): \(x \sqcap y = \text{min}(x, y)\), \(x \sqcup y = \text{max}(x, y)\)
• **Complete lattice** is a poset in which any subset (finite or infinite) has a glb and a lub
  • Every finite lattice is complete
• A complete lattice must have:
  • a least element \( \bot \)
  • a greatest element \( \top \)

**Example: Power Set Lattice**

\[
\begin{align*}
\top &= \{a, b, c\} \\
\{a, b\} &\quad \{a, c\} & \quad \{b, c\} \\
\{a\} &\quad \{b\} & \quad \{c\} \\
\bot &= \{\} 
\end{align*}
\]
Exercise

- Which are the following posets are lattices?

- To show a poset is not a lattice, it suffices to find a pair that does not have an lub or a glb.
- Two elements that don’t have an lub or glb cannot be comparable.
- View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a lattice.
Exercise

• Which are the following posets are lattices?

- No

• To show a poset is not a lattice, it suffices to find a pair that does not have an lub or a glb

• Two elements that don’t have an lub or glb cannot be comparable

• View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a lattice
Exercise

- Which are the following posets are lattices?

  - No

  - Yes

- To show a poset is not a lattice, it suffices to find a pair that does not have an lub or a glb.

- Two elements that don’t have an lub or glb cannot be comparable.

- View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a lattice.
Exercise

- Which are the following posets are lattices?

- To show a poset is not a lattice, it suffices to find a pair that does not have an lub or a glb.
- Two elements that don’t have an lub or glb cannot be comparable.
- View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a lattice.

- no
- yes ✓
- no