



CSCI 740 - Programming Language Theory

Lecture 27

Introduction to Abstract Interpretation

Instructor: Hossein Hojjat

November 6, 2017

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

$$W = (b \Rightarrow \text{wp}(c, W) \wedge \neg b \Rightarrow Q)$$

- 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

A Tiny Language

- Consider the following language with only integers and multiplications

$$e ::= n \mid e_1 \times e_2$$

- Operational semantics of this language:

$$\frac{}{n \Downarrow n} \qquad \frac{e_1 \Downarrow n_1 \quad e_2 \Downarrow n_2 \quad n_3 = n_1 \times n_2}{e_1 \times e_2 \Downarrow n_3}$$

- Take the operational semantics as the “ground truth”
- For this language the precise semantics is computable
 - but in general it is not

An Abstraction

- 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

$$\alpha : \text{Exp} \rightarrow \{+, -, 0\}$$

- α is called the **abstraction** function
- $\{-, 0, +\}$ is the **abstract domain**

$$\alpha(n) = \text{sign}(n)$$

$$\alpha(e_1 \times e_2) = \alpha(e_1) \otimes \alpha(e_2)$$

\otimes	+	0	-
+	+	0	-
0	0	0	0
-	-	0	+

- 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

$$e \Downarrow n \wedge n > 0 \Leftrightarrow \alpha(e) = +$$

$$e \Downarrow n \wedge n = 0 \Leftrightarrow \alpha(e) = 0$$

$$e \Downarrow n \wedge n < 0 \Leftrightarrow \alpha(e) = -$$

Another View of Soundness

- Associate each abstract value with the set of concrete values it represents
- γ is called the **concretization** function

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

$$\gamma(+)=\{n \in \mathbb{Z} \mid n > 0\}$$

$$\gamma(0)=\{0\}$$

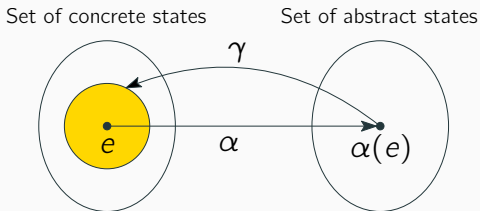
$$\gamma(-)=\{n \in \mathbb{Z} \mid n < 0\}$$

Another View of Soundness

- Soundness can be stated succinctly

$$\forall e \in \text{Exp}. \exists n \in \mathbb{Z}. e \Downarrow n \wedge n \in \gamma(\alpha(e))$$

- The real value of the expression is among the concrete values represented by the abstract value of the expression



- Extend our language with unary –

$$\frac{e \Downarrow n}{-e \Downarrow -n}$$

$$\alpha(-e) = \ominus\alpha(e)$$

	+	0	-
\ominus	-	0	+

Adding +

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

$$\frac{e_1 \Downarrow n_1 \quad e_2 \Downarrow n_2 \quad n_3 = n_1 + n_2}{e_1 + e_2 \Downarrow n_3}$$

$$\alpha(e_1 + e_2) = \alpha(e_1) \oplus \alpha(e_2)$$

\oplus	+	0	-
+	+	+	?
0	+	0	-
-	?	-	-

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

$$\gamma(\top) = \mathbb{Z}$$

\oplus	+	0	-	\top
+	+	+	\top	\top
0	+	0	-	\top
-	\top	-	-	\top
\top	\top	\top	\top	\top

Extending Other Operations

- We also need to extend the other abstract operations to work with \top

\otimes	+	0	-	\top
+	+	0	-	\top
0	0	0	0	0
-	-	0	+	\top
\top	\top	0	\top	\top

	+	0	-	\top
\ominus	-	0	+	\top

Exercise

- We know $(1 + 2) + -3 \Downarrow 0$
- Compute $\alpha((1 + 2) + -3)$

Exercise

- We know $(1 + 2) + -3 \Downarrow 0$
- Compute $\alpha((1 + 2) + -3)$

$$\alpha((1 + 2) + -3) =$$

$$(\alpha(1) \oplus \alpha(2)) \oplus \alpha(-3) =$$

$$(+ \oplus +) \oplus - = \top$$

Loss of Precision

- Abstract computation may lose information

$$(1 + 2) + -3 \Downarrow 0$$
$$\alpha((1 + 2) + -3) = \top$$

- 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

$$\gamma(\perp) = \{\}$$

\emptyset	+	0	-	\top	\perp
+	+	0	-	\top	\perp
0	\perp	\perp	\perp	\perp	\perp
-	-	0	+	\top	\perp
\top	\top	0	\top	\top	\perp
\perp	\perp	\perp	\perp	\perp	\perp

Adding Integer Division

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

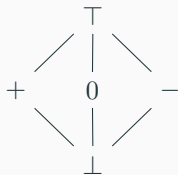
$$x \oplus \perp = \perp$$

$$x \otimes \perp = \perp$$

$$x \ominus \perp = \perp$$

The Abstract Domain

- Our abstract domain forms a **lattice**
- A partial order $x \leq y \Leftrightarrow \gamma(x) \subseteq \gamma(y)$
- Every finite subset has a least upper bound (lub) & greatest lower bound (glb)



Partial Orders

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

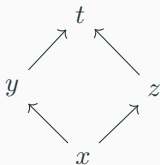
- x immediate predecessor of y : if $x \preccurlyeq y$ and there is no z such that

$$x \preccurlyeq z \preccurlyeq y$$

- 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.

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

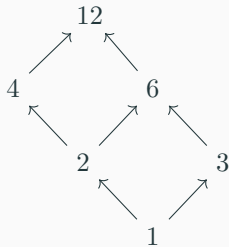


Exercise

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

Exercise

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



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

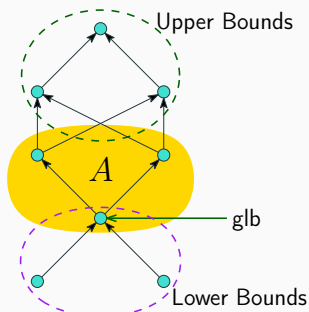
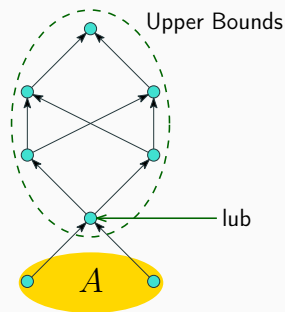
Total Order

- 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 $x \preceq y$ or $y \preceq x$ is true
- (\mathbb{N}, \leq) is total order
- Hasse diagram is one-track



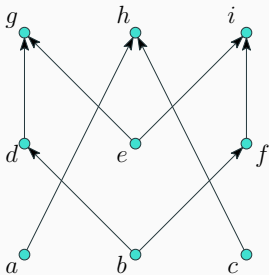
Subset Bounds

- 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: $\{b, d\}, \{a, c\}, \{d, e, f\}$

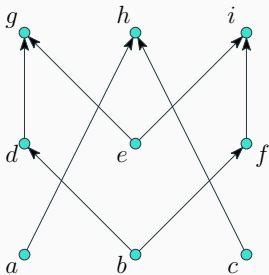


Exercise

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

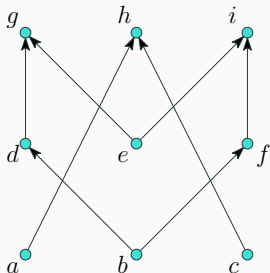
$\{b, d\}$:

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



Exercise

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



$\{b, d\}$:

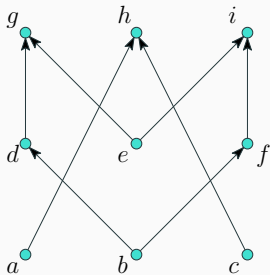
- Lower bounds: $\{b\}$ glb: b
- Upper bounds: $\{d, g\}$ lub: d because $d \preceq g$

$\{a, c\}$:

- 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

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)

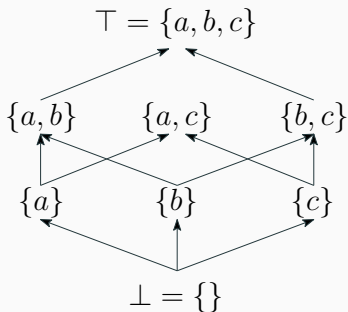
Example.

- For $(2^B, \subseteq)$: $x \sqcap y = x \cap y$, $x \sqcup y = x \cup y$
- For (\mathbb{Z}, \leq) : $x \sqcap y = \min(x, y)$, $x \sqcup y = \max(x, y)$

Complete Lattice

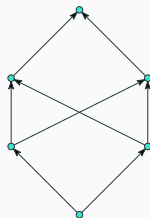
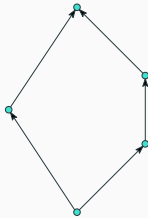
- **Complete lattice** is a poset in which any subset (finite or infinite) has a glb and a lub
 - Every finite lattice is complete
- A complete lattice must have:
 - a least element \perp
 - a greatest element \top

Example: Power Set Lattice



Exercise

- Which of the following posets are lattices?



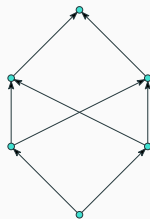
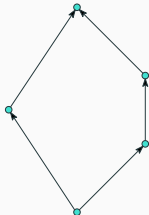
- To show a poset is not a lattice, it suffices to find a pair that does not have a lub or a glb
- Two elements that don't have a lub or glb cannot be comparable
- View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a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

Exercise

- Which of the following posets are lattices?



no



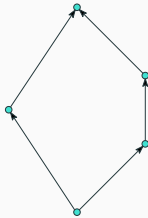
- To show a poset is not a lattice, it suffices to find a pair that does not have a lub or a glb
- Two elements that don't have a lub or glb cannot be comparable
- View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a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

Exercise

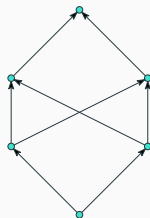
- Which of the following posets are lattices?



no



yes ✓



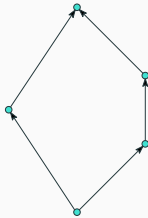
- To show a poset is not a lattice, it suffices to find a pair that does not have a lub or a glb
- Two elements that don't have a lub or glb cannot be comparable
- View the upper/lower bounds on a pair as a sub-Hasse diagram: If there is no greatest/least element in this sub-diagram, then it is not a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

Exercise

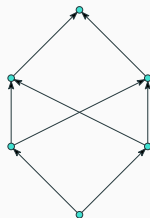
- Which of the following posets are lattices?



no



yes ✓



no

- To show a poset is not a lattice, it suffices to find a pair that does not have a lub or a glb
- Two elements that don't have a 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