Live variable analysis and available expressions analysis are similar

- Define some information that they need to compute
- Build constraints for the information
- Solve constraints iteratively:
  - Information always “increases” during iteration
  - Eventually, it reaches a fixed point

We would like a general framework

- Framework applicable to many other analyses
- Live variable/available expressions instances of the framework
Data-flow Analysis Framework

Data-flow analysis:

- Common framework for many compiler analyses
- Computes some information at each program point
- The computed information characterizes all possible executions of the program

Basic methodology:

- Describe information about the program using an algebraic structure called a lattice
- Build constraints that show how instructions and control flow influence the information in terms of values in the lattice
- Iteratively solve constraints

We start by defining lattices and see some of their properties
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 \( (P(S), \subseteq) \) is a poset
- \( (\mathbb{Z}, \leq) \) is a poset
Hasse Diagram

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

Hasse diagram:
- 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.
- $x \preceq y$, $y \preceq t$, $z \preceq t$, $x \preceq z$, $x \preceq t$
- $x \preceq x$, $y \preceq y$, $z \preceq z$, $t \preceq t$

```
\begin{tikzpicture}
  \node (x) at (0,0) {$x$};
  \node (y) at (1,1) {$y$};
  \node (z) at (1,-1) {$z$};
  \node (t) at (2,0) {$t$};

  \draw[->] (x) -- (y);
  \draw[->] (y) -- (z);
  \draw[->] (z) -- (x);
  \draw[->] (x) -- (t);
  \draw[->] (y) -- (t);
  \draw[->] (z) -- (t);
\end{tikzpicture}
```
Exercise

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

\[ \begin{array}{c}
12 \\
4 & 6 \\
2 & 3 \\
1 \\
\end{array} \]

$D_{12} = \{1, 2, 3, 4, 6, 12\}$
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
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 \((P(B), \subseteq)\): \(x \sqcap y = x \cap y\), \(x \sqcup y = x \cup y\)
- For \((\mathbb{Z}, \leq)\): \(x \sqcap y = \text{min}(x, y)\), \(x \sqcup y = \text{max}(x, y)\)
Complete Lattice

- **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**

$\top = \{a, b, c\}$

$\bot = \{\}$$

\{$a, b\} \quad \{$a, c\} \quad \{$b, c\}$

\{$a\} \quad \{$b\} \quad \{$c\}$
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?

- 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?

  ![Diagram 1](no)
  ![Diagram 2](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?

- no
- yes ✓
- 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
• Information computed by e.g. live variable and available expressions analyses can be expressed as elements of lattices

• If $x \leq y$ then $x$ is less or equally precise as $y$
  • i.e., $x$ is a conservative approximation of $y$

• Top $\top$: most precise, best case information

• Bottom $\bot$: least precise, worst case information

• Merge function = glb (meet) on lattice elements
  • Most precise element that is a conservative approximation of both elements
Example: Available Expressions

- Trivial answer with zero information, allows no optimization: \( \perp = \{\} \) (No expression available)
Example: Live Variables

- If $V$ is the set of all variables in a program and $P$ the power set of $V$, then $(P, \supseteq)$ is a lattice
- Sets of live variables are elements of this lattice
- Trivial answer with zero information, allows no optimization: $\bot = V$ (All variables are live, nothing is dead)
• Assume information we want to compute in a program is expressed using a lattice $L$

• To compute the information at each program point we need to:
  - Determine how each statement in the program changes the information
  - Determine how information changes at join/split points in the control flow
Transfer Functions

- Data-flow analysis defines a transfer function $F : L \rightarrow L$ for each statement in the program.
- Describes how the statement modifies the information.
- Consider $in(S)$ as information before $S$, and $out(S)$ as information after $S$.
- Forward analysis: $out(S) = F(in(S))$.
- Backward analysis: $in(S) = F(out(S))$. 
Sequential Composition

- Consider statements $S = S_1; ...; S_n$ with transfer functions $F_1, ..., F_n$
- $in(S)$ is information at the beginning
- $out(S)$ is information after at the end

- Forward analysis:
  $$out(S) = F_n(\cdots(F_1(in(S)))) = F_n \circ \cdots \circ F_1(in(S))$$

- Backward analysis:
  $$in(S) = F_1(\cdots(F_n(out(S)))) = F_1 \circ \cdots \circ F_n(out(S))$$
Split/Join Points

- Data-flow analysis uses meet/join operations at split/join points in the control flow
- Forward analysis:
  \[
  in(S) = \bigcap \{ out(S') | S' \in pred(S) \}
  \]
- Backward analysis:
  \[
  out(S) = \bigcap \{ in(S') | S' \in succ(S) \}
  \]