Lattices

- **Lattice:**
  - Set augmented with a partial order relation \(\sqsubseteq\)
  - Each subset has a LUB and a GLB
  - Can define: meet \(\sqcap\), join \(\sqcup\), top \(\top\), bottom \(\bot\)

- **Use lattice** to express information about a point in a program, where \(S_1 \sqsubseteq S_2\) means “\(S_1\) is less or equally precise as \(S_2\)”

- **To compute information:** build constraints that describe how the lattice information changes
  - Effect of instructions: transfer functions
  - Effect of control flow: meet operation
Transfer Functions

- Let $L$ = dataflow information lattice

- Transfer function $F_I : L \rightarrow L$ for each instruction $I$
  - Describes how $I$ modifies the information in the lattice
  - If $in[I]$ is info before $I$ and $out[I]$ is info after $I$, then
    - Forward analysis: $out[I] = F_I(in[I])$
    - Backward analysis: $in[I] = F_I(out[I])$

- Transfer function $F_B : L \rightarrow L$ for each basic block $B$
  - Is composition of transfer functions of instructions in $B$
  - If $in[B]$ is info before $B$ and $out[B]$ is info after $B$, then
    - Forward analysis: $out[B] = F_B(in[B])$
    - Backward analysis: $in[B] = F_B(out[B])$
Control Flow

• Meet operation models how to combine information at split/join points in the control flow
  – If in[B] is info before B and out[B] is info after B, then:
    Forward analysis: \( \text{in}[B] = \bigwedge \{ \text{out}[B'] \mid B' \in \text{pred}(B) \} \)
    Backward analysis: \( \text{out}[B] = \bigwedge \{ \text{in}[B'] \mid B' \in \text{succ}(B) \} \)

• Can alternatively use join operation \( \bigvee \) (equivalent to using the meet operation \( \bigwedge \) in the reversed lattice)
Treatment as $F: L^n \rightarrow L^n$

- For a data flow analysis problem
  - With lattice $L$
  - Basic blocks $B_1$, $B_2$, ..., $B_n$
  - Transfer functions $F_1$, $F_2$, ..., $F_n$

- Treat as
  - Iteration of function $F$: $L^n \rightarrow L^n$
    $\top, F(\top), F(F(\top)), ...$
  - Where $F$ summarizes effect of one sweep for all blocks $B$ in a given order of either
    $\text{out}[B] = ...$ and $\text{in}[B] = F_B(\text{out}[B])$ (for backward)
    $\text{in}[B] = ...$ and $\text{out}[B] = F_B(\text{in}[B])$ (for forward)
Monotonicity

- Function $F : L \rightarrow L$ is monotonic if $x \preceq y$ implies $F(x) \preceq F(y)$
- A monotonic function is “order preserving”
- Contrast with
  
  For all $x$, $F(x) \preceq x$
- $F$ is monotonic but $C = F(B) \npreceq B$
Monotonicity of Meet

- Meet operation is monotonic over \( L \times L \), i.e.,
  \[
  x_1 \sqsubseteq y_1 \text{ and } x_2 \sqsubseteq y_2 \implies (x_1 \sqcap x_2) \sqsubseteq (y_1 \sqcap y_2)
  \]

- **Proof:**
  - any lower bound of \( \{x_1, x_2\} \) is also a lower bound of \( \{y_1, y_2\} \), because \( x_1 \sqsubseteq y_1 \) and \( x_2 \sqsubseteq y_2 \)
  - \( x_1 \sqcap x_2 \) is a lower bound of \( \{x_1, x_2\} \)
  - So \( x_1 \sqcap x_2 \) is a lower bound of \( \{y_1, y_2\} \)
  - But \( y_1 \sqcap y_2 \) is the greatest lower bound of \( \{y_1, y_2\} \)
  - Hence \( (x_1 \sqcap x_2) \sqsubseteq (y_1 \sqcap y_2) \)
Fixed Points

• x in lattice L is a **fixed point of function** F iff x=F(x)
• Tarski-Knaster Fixed Point Theorem. The fixed points of a monotonic function on a complete lattice form a complete lattice. In particular, there is a maximal fixed point (MFP).
Chains in Lattices

- A chain in a lattice $L$ is a totally ordered subset $S$ of $L$: $x \sqsubseteq y$ or $y \sqsubseteq x$ for any $x, y \in S$

- In other words:
  Elements in a totally ordered subset $S$ can be indexed to form an ascending sequence:
  
  $x_1 \sqsubseteq x_2 \sqsubseteq x_3 \sqsubseteq \ldots$

  or they can be indexed to form a descending sequence:

  $x_1 \sqsupseteq x_2 \sqsupseteq x_3 \sqsupseteq \ldots$

- Height of a lattice = size of its largest chain
- Lattice with finite height: only has finite chains
Iterative Computation of Solution

• Let $F$ be a monotonic function over lattice $L$
• $\top \subseteq F(\top) \subseteq F(F(\top)) \subseteq \ldots$ is a descending chain
• If $L$ has finite height, the chain ends at the maximal fixed point of $F$ (MFP)
Multiple Solutions

• Dataflow equations may have multiple solutions

• **Example:** live variables

  Equations:  
  \[ I_1 = I_2 \setminus \{y\} \]
  \[ I_3 = (I_4 \setminus \{x\}) \cup \{y\} \]
  \[ I_2 = I_1 \cup I_3 \]
  \[ I_4 = \{x\} \]

  **Solution 1:**  
  \[ I_1 = \emptyset, \quad I_2 = \{y\}, \quad I_3 = \{y\}, \quad I_4 = \{x\} \]

  **Solution 2:**  
  \[ I_1 = \{x\}, \quad I_2 = \{x, y\}, \quad I_3 = \{y\}, \quad I_4 = \{x\} \]

For any solution \( FP \) of the dataflow equations \( FP \subseteq MFP \)
FP is said to be a conservative or safe solution
Meet Over Paths Solution (forward)

- Is MFP the best solution to an analysis problem?

- Alternative to MFP: a different way to compute solution
  - Let G be the control flow graph with start block $B_0$
  - For each path $p_n = [B_0, B_1, \ldots, B_n]$ from $B_0$ to block $B_n$
    define $F[p_n] = F_{B_{n-1}} \circ F_{B_{n-1}} \circ \ldots \circ F_{B_0}$
  - Compute solution as
    $$\text{in}[B_n] = \prod \{ F[p_n](\text{start value}) | \text{all paths } p_n \text{ from } B_0 \text{ to } B_n \}$$
- This solution is the Meet Over Paths (MOP) solution for block $B_n$
MFP versus MOP

• **Precision**: MOP solution is at least as precise as MFP
  \[ \text{MFP} \sqsubseteq \text{MOP} \]

• **Why not use MOP?**
  1. **Exponential number of paths**: for a program consisting of a sequence of \( N \) if statement, there will \( 2^N \) paths in the control flow graph
  2. **Infinite number of paths**: for loops in the CFG
Distributivity

• Function $F : L \rightarrow L$ is **distributive** if
  \[ F(x \cap y) = F(x) \cap F(y) \]

• **Property:** $F$ is monotonic iff $F(x \cap y) \subseteq F(x) \cap F(y)$
  - any distributive function is monotonic!
Importance of Distributivity

• **Property:** if transfer functions are *distributive*, then the solution to the dataflow equations is identical to the meet-over-paths solution

\[ \text{MFP} = \text{MOP} \]

• For distributive transfer functions, can compute the intractable MOP solution using the iterative fixed-point algorithm
Better Than MOP?

• Is MOP the best solution to the analysis problem?

• MOP computes solution for all paths in the CFG

• There may be paths that will never occur in any execution

• So MOP is conservative

• IDEAL = solution that takes into account only paths that occur in some execution

• This is the best solution

• … but it is undecidable
Dataflow Equations

• Solve equations: use an iterative algorithm
  - Initialize in[B_s] = start value
  - Initialize everything else to ⊤
  - Repeatedly apply rules
  - Stop when reach a fixed point
Kildall Algorithm (forward)

\[ \text{in}[B_S] = \text{start value} \]
\[ \text{out}[B] = \top, \text{for all } B \]

\textbf{repeat} \\
\hspace{1em} \textbf{for} each basic block \( B \neq B_S \) \\
\hspace{2em} \text{in}[B] = \cap \{ \text{out}[B'] \mid B' \in \text{pred}(B) \} \\
\hspace{1em} \textbf{for} each basic block \( B \) \\
\hspace{2em} \text{out}[B] = F_B(\text{in}[B]) \\
\textbf{until} no change
Efficiency

• Algorithm is inefficient
  - Effects of basic blocks re-evaluated even if the input information has not changed

• Better: re-evaluate blocks only when necessary

• Use a worklist algorithm
  - Keep of list of blocks to evaluate
  - Initialize list to the set of all basic blocks
  - If out[B] changes after evaluating out[B] = F_B(in[B]), then add all successors of B to the list
Worklist Algorithm (forward)

\[ \text{in}[B_S] = \text{start value} \]
\[ \text{out}[B] = \top, \text{for all } B \]
\[ \text{worklist} = \text{set of all basic blocks } B \]

\textbf{repeat}

\textbf{remove} a node \( B \) from the worklist
\[ \text{in}[B] = \cap \{ \text{out}[B'] | B' \in \text{pred}(B) \} \]
\[ \text{out}[B] = F_B(\text{in}[B]) \]
\textbf{if} out\([B]\) has changed \textbf{then}
\[ \text{worklist} = \text{worklist} \cup \text{succ}(B) \]

\textbf{until} worklist = \( \emptyset \)
Correctness

• Initial algorithm is correct
  – If dataflow information does not change in the last iteration, then it satisfies the equations

• Worklist algorithm is correct
  – Maintains the invariant that
    \[ \text{in}[B] = \prod \{\text{out}[B'] | B' \in \text{pred}(B)\} \]
    \[ \text{out}[B] = F_B(\text{in}[B]) \]
    for all the blocks \( B \) not in the worklist
  – At the end, worklist is empty
Summary

• Dataflow analysis
  – sets up system of equations
  – iteratively computes MFP
  – Terminates because transfer functions are monotonic and lattice has finite height

• Other possible solutions: FP, MOP, IDEAL
• All are safe solutions, but some are more precise:
  \[ \text{FP} \sqsubseteq \text{MFP} \sqsubseteq \text{MOP} \sqsubseteq \text{IDEAL} \]
• MFP = MOP if distributive transfer functions

• MOP and IDEAL are intractable
• Compilers use dataflow analysis and MFP