Lattice-Theoretic Framework for Data-Flow Analysis

Last time
- Generalizing data-flow analysis
- Reaching definitions vs. reaching constants

Today
- Introduce lattice-theoretic framework for data-flow analysis

Context

Goals
- Provide a single formal model that describes all data-flow analyses
- Formalize the notions of safe, conservative, and optimistic
- Place bounds on time complexity of data-flow analysis

Approach
- Define domain of program properties (flow values) computed by data-flow analysis, and organize the domain of elements as a lattice
- Define flow functions and a merge function over this domain using lattice operations
- Exploit lattice theory to achieving goals
Lattices

Define lattice \( L = (V, \wedge) \)
- \( V \) is a set of elements of the lattice
- \( \wedge \) is a binary relation over the elements of \( V \) (meet or greatest lower bound)

Properties of \( \wedge \)
- \( x, y \in V \Rightarrow x \wedge y \in V \) (closure)
- \( x, y \in V \Rightarrow x \wedge y = y \wedge x \) (commutativity)
- \( x, y, z \in V \Rightarrow (x \wedge y) \wedge z = x \wedge (y \wedge z) \) (associativity)

Lattices (cont)

Under (\( \leq \))
- Imposes a partial order on \( V \)
- \( x \leq y \iff x \wedge y = x \)

Top (\( T \))
- A unique “greatest” element of \( V \) (if it exists)
- \( \forall x \in V - \{T\}, x < T \)

Bottom (\( \bot \))
- A unique “least” element of \( V \) (if it exists)
- \( \forall x \in V - \{\bot\}, \bot < x \)

Height of lattice \( L \)
- The longest path through the partial order from greatest to least element (top to bottom)
**Data-Flow Analysis via Lattices**

**Relationship**
- Elements of the lattice (V) represent flow values (in[] and out[] sets)
  - *e.g.*, Sets of live variables for liveness
- T represents “best-case” information (initial flow value)
  - *e.g.*, Empty set
- ⊥ represents “worst-case” information
  - *e.g.*, Universal set
- ^ (meet) merges flow values
  - *e.g.*, Set union
- If x ≤ y, then x is a conservative approximation of y
  - *e.g.*, Superset

\[
\begin{align*}
\text{Initially} & \quad \text{for liveness} \\
\{T\} & \quad \text{print}(x) \quad \text{print}(y) \quad \{T\}
\end{align*}
\]

\[
\begin{align*}
\text{Finally} & \quad x = y \\
\{x\} & \quad \text{print}(x) \quad \text{print}(y) \quad \{y\}
\end{align*}
\]

---

**Data-Flow Analysis via Lattices (cont)**

Remember what these flow values represent
- At each program point a lattice element represents an in[] set or an out[] set

\[
\begin{align*}
\text{Initially} & \quad \text{for liveness} \\
\{T\} & \quad \{T\} \\
\{T\} & \quad \text{print}(x) \quad \text{print}(y) \quad \{T\}
\end{align*}
\]

\[
\begin{align*}
\text{Finally} & \quad x = y \\
\{x\} & \quad \text{print}(x) \quad \text{print}(y) \quad \{y\}
\end{align*}
\]
Data-Flow Analysis Frameworks

Data-flow analysis framework
- A set of flow values \( V \)
- A binary meet operator \( \wedge \)
- A set of flow functions \( F \) (also known as transfer functions)

Flow Functions
- \( F = \{ f: V \rightarrow V \} \)
  - \( f \) describes how each node in CFG affects the flow values
- Flow functions map program behavior onto lattices

Visualizing DFA Frameworks as Lattices

Example: Liveness analysis with 3 variables
\( S = \{ v1, v2, v3 \} \)

\( S^2 = \{ \{ v1, v2, v3 \}, \{ v1, v2 \}, \{ v1, v3 \}, \{ v2, v3 \}, \{ v1 \}, \{ v2 \}, \{ v3 \} \} \)

- Meet \( \wedge \):
  - \( \leq: \)
  - Top(\( T \)): \( \emptyset \)
  - Bottom (\( \bot \)): \( v \)
- \( F \):
  - \( f_n(X) = \text{Gen}_n \cup (X - \text{Kill}_n), \forall n \} \)

Inferior solutions are lower on the lattice
More conservative solutions are lower on the lattice
More Examples

Reaching definitions
- \( V: 2^S (S = \text{set of all defs}) \)
- \( \wedge: \cup \)
- \( \leq: \supseteq \)
- \( \text{Top}(T): \emptyset \)
- \( \text{Bottom} (\bot): S \)
- \( F: \ldots \)

Reaching Constants
- \( V: 2^{*c}, \text{variables v and constants c} \)
- \( \wedge: \cap \)
- \( \leq: \subseteq \)
- \( \text{Top}(T): v \lor c \)
- \( \text{Bottom} (\bot): \emptyset \)
- \( F: \ldots \)

Tuples of Lattices

Problem
- Simple analyses may require very complex lattices (e.g., Reaching constants)

Solution
- Use a tuple of lattices, one per variable

\[ L = (V, \wedge) = (L_T = (V_T, \wedge_T))^N \]
- \( V = (V_T)^N \)
- Meet (\( \wedge \)): point-wise application of \( \wedge_T \)
- \((..., v_i, ...) \leq (..., u_i, ...) \Rightarrow v_i \leq_T u_i, \forall i \)
- Top (T): tuple of tops (\( T_T \))
- Bottom (\( \bot \)): tuple of bottoms (\( \bot_T \))
- Height (\( L \)) = \( N \times \text{height}(L_T) \)
Tuples of Lattices Example

Reaching constants (previously)
- \( P = v \times c \), for variables \( v \) & constants \( c \)
- \( V: 2^P \)

Alternatively
- \( V = c \cup \{ T, \bot \} \)

The whole problem is a tuple of lattices, one for each variable

Examples of Lattice Domains

Two-point lattice (\( T \) and \( \bot \))
- Examples?
- Implementation?

Set of incomparable values (and \( T \) and \( \bot \))
- Examples?

Powerset lattice (\( 2^S \))
- \( T = \emptyset \) and \( \bot = S \), or vice versa
- Isomorphic to tuple of two-point lattices
Solving Data-Flow Analyses

Goal

– For a forward problem, consider all possible paths from the entry to a given program point, compute the flow values at the end of each path, and then meet these values together
– Meet-over-all-paths (MOP) solution at each program point

\[ \bigwedge_{\text{all paths } n_1, n_2, \ldots, n_i} (f_{n_i}(\ldots f_{n_2}(f_{n_1}(v_{\text{entry}})))) \]

Solving Data-Flow Analyses (cont)

Problems

– Loops result in an infinite number of paths
– Statements following merge must be analyzed for all preceding paths
  – Exponential blow-up

Solution

– Compute meets early (at merge points) rather than at the end
– Maximum fixed-point (MFP)

Questions

– Is this legal?
– Is this efficient?
– Is this accurate?
Legality

“Is $v_{MFP}$ legal?” $\equiv$ “Is $v_{MFP} \leq v_{MOP}$?”

Look at Merges

$v_{MOP} = F_x(v_{p1}) \land F_y(v_{p2})$
$v_{MFP} = F_x(v_{p1} \land v_{p2})$
$v_{MFP} \leq v_{MOP} = F_x(v_{p1} \land v_{p2}) \leq F_x(v_{p1}) \land F_x(v_{p2})$

Observation

$\forall x, y \in V$

$f(x \land y) \leq f(x) \land f(y)$ $\iff$ $x \leq y \Rightarrow f(x) \leq f(y)$

$\therefore$ $v_{MFP}$ legal when $F_x$ (really, the flow functions) are monotonic
Efficiency

Parameters
- \( n \): Number of nodes in the CFG
- \( k \): Height of lattice
- \( t \): Time to execute one flow function

Complexity
- \( O(nkt) \)

Example
- Reaching definitions?

Accuracy

Distributivity
- \( f(u^\wedge v) = f(u)^\wedge f(v) \)
- \( \nu_{MFP} \leq \nu_{MOP} = F_{r}(v_{p1}^\wedge v_{p2}) \leq F_{r}(v_{p1})^\wedge F_{r}(v_{p2}) \)
- If the flow functions are distributive, \( MFP = MOP \)

Examples
- Reaching definitions?
- Reaching constants?
  \[
  f(u^\wedge v) = f(\{x=2,y=3\}^\wedge \{x=3,y=2\}) \\
  = f(\emptyset) = \emptyset \\
  f(u)^\wedge f(v) = f(\{x=2,y=3\})^\wedge f(\{x=3,y=2\}) \\
  = \{x=2,y=3\}^\wedge \{x=3,y=2\} = \{w=5\} \\
  \Rightarrow MFP \neq MOP
  \]
Another Example

Integer range analysis
- Calculate an approximation to the set of integer values that integer variables can take on
- Uses
  - Array-bounds-check elimination
  - Array access dependence testing
  - Overflow check elimination

What is the domain?
- What is its height?

What are the flow functions?
- Are they monotonic?

Concepts

Lattices
- Conservative approximation
- Optimistic (initial guess)
- Data-flow analysis frameworks
- Tuples of lattices

Data-flow analysis
- Fixed point
- Meet-over-all-paths (MOP)
- Maximum fixed point (MFP)
- Legal/safe (monotonic)
- Efficient
- Accurate (distributive)
Next Time

Lecture
– Program representations (static single assignment)