Lecture 3
Foundation of Data Flow Analysis

I Semi-lattice (set of values, meet operator)
II Transfer functions
III Correctness, precision and convergence
IV Meaning of Data Flow Solution

Reading: Chapter 9.3

I. Purpose of a Framework

- **Purpose 1**
  - Prove properties of entire family of problems once and for all
    - Will the program converge?
    - What does the solution to the set of equations mean?
- **Purpose 2:**
  - Aid in software engineering: re-use code

The Data-Flow Framework

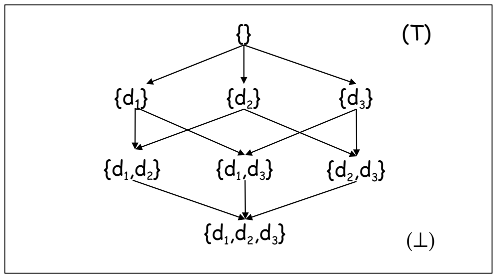
- **Data-flow problems (F, V, ∧) are defined by**
  - A semi-lattice
    - domain of values V
    - meet operator ∧: V x V → V
  - A family of transfer functions F: V → V

Semi-lattice: Structure of the Domain of Values

- **A semi-lattice S = <a set of values V, a meet operator ∧>**
- **Properties of the meet operator**
  - idempotent: x ∧ x = x
  - commutative: x ∧ y = y ∧ x
  - associative: x ∧ (y ∧ z) = (x ∧ y) ∧ z
- **Examples of meet operators ?**
- **Non-examples ?**

Example of a Semi-Lattice Diagram

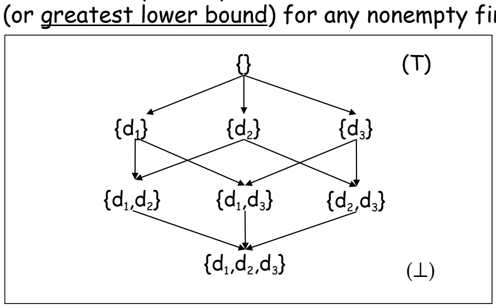
- (V, $\wedge$) : V = {x | such that $x \subseteq \{d_1,d_2,d_3\}$}, $\wedge$ = U



- x $\wedge$ y = first common descendant of x & y **important**
- A meet semi-lattice is bounded if there exists a top element T, such that x $\wedge$ T = x for all x.
- A bottom element $\perp$ exists, if x $\wedge$ $\perp$ = $\perp$ for all x.

Meet Semi-Lattices vs Partially Ordered Sets

- A **meet-semilattice** is a partially ordered set which has a meet (or greatest lower bound) for any nonempty finite subset.



- Greatest lower bound: x $\wedge$ y = First common descendant of x & y
- Largest: top element T, if x $\wedge$ T = x for all x.
- Smallest: bottom element $\perp$, if x $\wedge$ $\perp$ = $\perp$ for all x.

A Meet Operator Defines a Partial Order

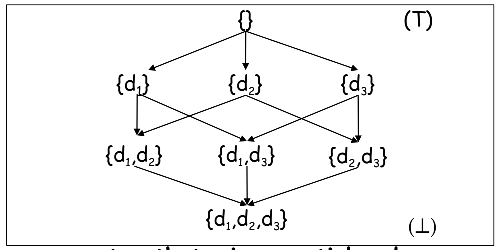
- **Partial order of a meet semi-lattice**

  $\le$: $x \le y$ if and only if $x \wedge y = x$

$y \xrightarrow{\text{path}} x \equiv (x \wedge y = x) \equiv (x \le y)$

- Meet operator: ∪

  Partial order $\le$:



- **Properties of meet operator guarantee that $\le$ is a partial order**
  - Reflexive: $x \le x$
  - Antisymmetric: if $x \le y$ and $y \le x$ then $x = y$
  - Transitive: if $x \le y$ and $y \le z$ then $x \le z$

Drawing a Semi-Lattice Diagram

- $(x < y) \equiv (x \le y) \wedge (x \neq y)$

- ***A* semi-lattice diagram:**
  - Set of nodes: set of values
  - Set of edges $\{(y, x)$: $x < y$ and $\neg \exists z$ s.t. $(x < z) \wedge (z < y)\}$

Summary

Three ways to define a semi-lattice:

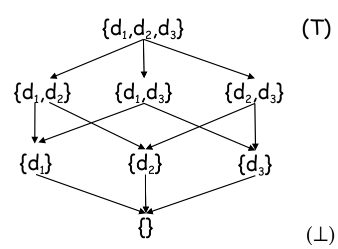
- Set of values + meet operator
  - idempotent: $x \wedge x = x$
  - commutative: $x \wedge y = y \wedge x$
  - associative: $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

- Set of values
  + partial order with a greatest lower bound for any nonempty subset
  - Reflexive: $x \leq x$
  - Antisymmetric: if $x \leq y$ and $y \leq x$ then $x = y$
  - Transitive: if $x \leq y$ and $y \leq z$ then $x \leq z$

- A semi-lattice diagram

Another Example

- Semi-lattice
  - $V = \{x \mid \text{such that } x \subseteq \{d_1, d_2, d_3\}\}$
  - $\wedge = \cap$



  - $\leq$ is

One Element at a Time

- **A semi-lattice for data flow problems can get quite large: 2^n elements for n var/definition**
- **A useful technique:**
  - define semi-lattice for 1 element
  - product of semi-lattices for all elements
- **Example**: Union of definitions
  - For each element

def1 def2 def1 x def2

$$\begin{tikzcd} \{\} \arrow[d] \\ \{d_1\} \end{tikzcd} \qquad \begin{tikzcd} \{\} \arrow[d] \\ \{d_2\} \end{tikzcd} \qquad \begin{tikzcd} & \{\},\{\} \arrow[dl] \arrow[dr] & \\ \{d_1\},\{\} \arrow[dr] & & \{\},\{d_2\} \arrow[dl] \\ & \{d_1\},\{d_2\} & \end{tikzcd}$$

  - $\langle x_1, x_2 \rangle \le \langle y_1, y_2 \rangle$ iff $x_1 \le y_1$ and $x_2 \le y_2$

Descending Chain

- **Definition**
  - The **height** of a lattice is the largest number of > relations that will fit in a descending chain.

$$x_0 > x_1 > \ldots$$

- **Height of values in reaching definitions?**
- **Important property: finite descending chains**

II. Transfer Functions

- ***A family of transfer functions*** F
- **Basic Properties** $f : V \rightarrow V$
  - Has an identity function
    - $\exists f$ such that $f(x) = x$, for all x.
  - Closed under composition
    - if $f_1, f_2 \in F$, $f_1 \bullet f_2 \in F$

Monotonicity: 2 Equivalent Definitions

- A framework (F, V, $\wedge$) is monotone iff
  - $x \leq y$ implies $f(x) \leq f(y)$
- Equivalently,
  a framework (F, V, $\wedge$) is monotone iff
  - $f(x \wedge y) \leq f(x) \wedge f(y)$,
  - meet inputs, then apply f
    $\leq$
    apply f individually to inputs, then meet results

Example

- **Reaching definitions: $f(x)$ = Gen ∪ (x - Kill), $\wedge$ = ∪**
  - Definition 1:
    - Let $x_1 \le x_2$,
      $f(x_1)$: Gen ∪ (x_1 - Kill)
      $f(x_2)$: Gen ∪ (x_2 - Kill)
  - Definition 2:
    - $f(x_1 \wedge x_2)$ = (Gen ∪ ((x_1 ∪ x_2) - Kill))
      $f(x_1) \wedge f(x_2)$ = (Gen ∪ (x_1 - Kill)) ∪ (Gen ∪ (x_2 - Kill))

Distributivity

- ***A framework (F, V, $\wedge$) is distributive if and only if***
  $f(x \wedge y) = f(x) \wedge f(y)$,

  meet input, then apply f is **equal to**
  apply the transfer function individually then merge result

Important Note

- Monotone framework **does not mean** that $f(x) \le x$
  - e.g. Reaching definition for two definitions in program
  - suppose: f: Gen = $\{d_1\}$; Kill = $\{d_2\}$

III. Properties of Iterative Algorithm

- **Given:**

  $\wedge$ and monotone data flow framework

  Finite descending chain

  $\Rightarrow$ Converges

- **Initialization of interior points to T**

  $\Rightarrow$ Maximum Fixed Point (MFP) solution of equations

Behavior of iterative algorithm (intuitive)

For each IN/OUT of an interior program point:

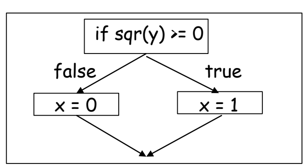
- Its value cannot go up (new value ≤ old value) during algorithm
- Start with T (largest value)
- Proof by induction
  - Apply 1st transfer function / meet operator ≤ old value (T)
  - Inputs to "meet" change (get smaller)
    - since inputs get smaller, new output ≤ old output
  - Inputs to transfer functions change (get smaller)
    - monotonicity of transfer function:
      since input gets smaller, new output ≤ old output
- Algorithm iterates until equations are satisfied
- Values do not come down unless some constraints drive them down.
- Therefore, finds the largest solution that satisfies the equations

IV. What Does the Solution Mean?

- **IDEAL data flow solution**
  - Let $f_1, \ldots, f_m : \in F$, f_i is the transfer function for node i

    $f_p = f_{n_k} \bullet \ldots \bullet f_{n_1}$, p is a path through nodes $n_1, \ldots, n_k$

    f_p = identify function, if p is an empty path
  - For each node n: $\wedge f_{p_i}$ (boundary value),
    for <u>all possibly executed paths</u> p_i reaching n
  - Example



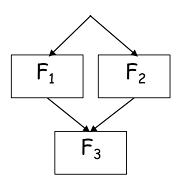
- **Determining all possibly executed paths is undecidable**

Meet-Over-Paths MOP

- **Err in the conservative direction**
- **Meet-Over-Paths MOP**
  - Assume every edge is traversed
  - For each node n:
  - MOP(n) = $\wedge f_{p_i}$ (boundary value), for all paths p_i reaching n
- **Compare MOP with IDEAL**
  - MOP includes more paths than IDEAL
  - MOP = IDEAL $\wedge$ Result(Unexecuted-Paths)
  - MOP $\leq$ IDEAL
  - MOP is a "smaller" solution, more conservative, **safe**
- **MOP $\leq$ IDEAL**
  - Goal: as close to MOP from below as possible

Solving Data Flow Equations

- **What is the difference between MOP and MFP of data flow equations?**



- **Therefore**
  - FP $\leq$ MFP $\leq$ MOP $\leq$ IDEAL
  - FP, MFP, MOP are safe
  - If framework is distributive, FP $\leq$ MFP = MOP $\leq$ IDEAL

Summary

- **A data flow framework**
  - Semi-lattice
    - set of values (top)
    - meet operator
    - finite descending chains?
  - Transfer functions
    - summarizes each basic block
    - boundary conditions
- **Properties of data flow framework:**
  - Monotone framework and finite descending chains

    $\Rightarrow$ iterative algorithm converges
    $\Rightarrow$ finds maximum fixed point (MFP)
    $\Rightarrow$ FP $\leq$ MFP $\leq$ MOP $\leq$ IDEAL

  - Distributive framework

    $\Rightarrow$ FP $\leq$ MFP $=$ MOP $\leq$ IDEAL