CS738: Advanced Compiler Optimizations

Foundations of Data Flow Analysis

Amey Karkare
karkare@cse.iitk.ac.in

http://www.cse.iitk.ac.in/~karkare/cs738
Department of CSE, IIT Kanpur

Agenda

- *Intraprocedural* Data Flow Analysis
  - We looked at 4 classic examples
  - Today: Mathematical foundations

Taxonomy of Dataflow Problems

- Categorized along several dimensions
  - the information they are designed to provide
  - the direction of flow
  - confluence operator
- Four kinds of dataflow problems, distinguished by
  - the operator used for confluence or divergence
  - data flows backward or forward

Taxonomy of Dataflow Problems

Confluence $\rightarrow$ **Direction** $\downarrow$	$\bigcup$	$\bigcap$
Forward	R D	Av E
Backward	L V	V B E

Why Data Flow Analysis Works?

- Suitable initial values and boundary conditions
- Suitable domain of values
  - Bounded, Finite
- Suitable meet operator
- Suitable flow functions
  - monotonic, closed under composition
- But what is **SUITABLE** ?

Lattice Theory

Partially Ordered Sets

- Posets
  S: a set
  $\leq$: a relation
  $(S, \leq)$ is a **poset** if for $x, y, z \in S$
  - $x \leq x$ (reflexive)
  - $x \leq y$ and $y \leq x \Rightarrow x = y$ (antisymmetric)
  - $x \leq y$ and $y \leq z \Rightarrow x \leq z$ (transitive)

Chain

- Linear Ordering
- Poset where every pair of elements is comparable
- $x_1 \leq x_2 \leq \ldots \leq x_k$ is a chain of length k
- We are interested in chains of finite length

Observation

- Any **finite nonempty subset** of a poset has **minimal** and **maximal** elements
- Any **finite nonempty chain** has **unique** minimum and maximum elements

Semilattice

- Set S and meet $\bigwedge$
- $x, y, z \in S$
  - $x \bigwedge x = x$ (idempotent)
  - $x \bigwedge y = y \bigwedge x$ (commutative)
  - $x \bigwedge (y \bigwedge z) = (x \bigwedge y) \bigwedge z$ (associative)
- Partial order for semilattice
  - $x \leq y$ if and only if $x \bigwedge y = x$
  - Reflexive, antisymmetric, transitive

Border Elements

- Top Element ($\top$)
  - $\forall x \in S, x \bigwedge \top = \top \bigwedge x = x$
- (Optional) Bottom Element ($\bot$)
  - $\forall x \in S, x \bigwedge \bot = \bot \bigwedge x = \bot$

Familiar (Semi)Lattices

- Powerset for a set S, 2^S
- Meet $\bigwedge$ is $\cap$
- Partial Order is $\subseteq$
- Top element is S
- Bottom element is $\emptyset$

Familiar (Semi)Lattices

- Powerset for a set S, 2^S
- Meet $\bigwedge$ is $\cup$
- Partial Order is $\supseteq$
- Top element is $\emptyset$
- Bottom element is S

Greatest Lower Bound (glb)

- $x, y, z \in S$
- glb of x and y is an element g such that
  - $g \leq x$
  - $g \leq y$
  - if $z \leq x$ and $z \leq y$ then $z \leq g$

QQ

- $x, y \in S$
- $(S, \bigwedge)$ is a semilattice
- Prove that $x \bigwedge y$ is glb of x and y.

Semi(?)-Lattice

- We can define symmetric concepts
  - $\geq$ order
  - Join operation ($\bigvee$)
  - Least upper bound (lub)

Lattice

- $(S, \bigwedge, \bigvee)$ is a lattice iff for each **non-empty finite** subset Y of S both $\bigwedge Y$ and $\bigvee Y$ are in S.
- $(S, \bigwedge, \bigvee)$ is a complete lattice iff for each subset Y of S both $\bigwedge Y$ and $\bigvee Y$ are in S.

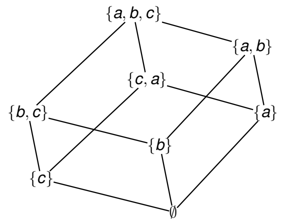
Lattice

- Complete lattice $(S, \bigwedge, \bigvee)$
  - For every pair of elements x and y, both $x \bigwedge y$ and $x \bigvee y$ should be in S
  - Example : Powerset lattice
- We will talk about **meet** semi-lattices only
  - except for some proofs

Lattice Diagram

- Graphical view of posets
- Elements = the nodes in the graph
- If $x < y$ then x is depicted lower than y in the diagram
- An edge between x and y (x lower than y) implies $x < y$ and no other element z exists s.t. $x < z < y$ (i.e. transitivity is excluded)

Lattice Diagram



Lattice Diagram for $(\{a, b, c\}, \cap)$

$x \bigwedge y$ = the highest z for which there are paths downward from both x and y.

Data Flow Analysis Framework

- $(D, S, \bigwedge, F)$
- D: direction – Forward or Backward
- $(S, \bigwedge)$: Semilattice – Domain and meet
- F: family of transfer functions of type $S \to S$ (see next slide)

Transfer Functions

- F: family of functions $S \to$ S. Must Include
  - functions suitable for the boundary conditions (constant transfer functions for *Entry* and *Exit* nodes)
  - Identity function I:

$$I(x) = x \quad \forall x \in S$$

  - Closed under composition:

$$f, g \in F, \quad f \circ g \quad \Rightarrow \quad h \in F$$

Monotonic Functions

- $(S, \leq)$: a poset
- $f : S \to S$ is monotonic iff

$$\forall x, y \in S \quad x \leq y \Rightarrow f(x) \leq f(y)$$

- Composition preserves monotonicity
  - If f and g are monotonic, $h = f \circ g$, then h is also monotonic

Monotone Frameworks

- $(D, S, \bigwedge, F)$ is monotone if the family F consists of monotonic functions only

$$f \in F, \quad \forall x, y \in S \qquad x \leq y \Rightarrow f(x) \leq f(y)$$

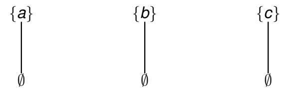
- Equivalently

$$f \in F, \quad \forall x, y \in S \qquad f(x \bigwedge y) \leq f(x) \bigwedge f(y)$$

- Proof? : QQ in class

What if there is a large number of elements?

- Combine simple lattices to build a complex one
- Superset lattices for singletons

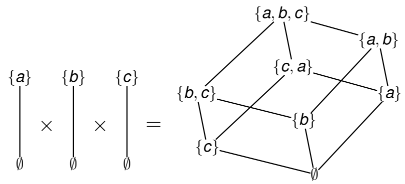


- Combine to form superset lattice for multi-element sets

Product Lattice

- $(S, \bigwedge)$ is product lattice of $(S_1, \bigwedge_1)$ and $(S_2, \bigwedge_2)$ when
  $S = S_1 \times S_2$ (domain)
  For (a_1, a_2) and $(b_1, b2) \in S$
  $(a_1, a_2) \bigwedge (b_1, b_2) = (a_1 \bigwedge_1 b_1, a_2 \bigwedge_2 b_2)$
  $(a_1, a_2) \leq (b_1, b_2)$ iff $a_1 \leq_1 b_1$ and $a_2 \leq_2 b_2$
  $\leq$ relation follows from $\bigwedge$
- Product of lattices is associative
- Can be generalized to product of $N > 2$ lattices
- $(S_1, \bigwedge_1), (S_2, \bigwedge_2), \ldots$ are called component lattices

Product Lattice: Example



Height of a Semilattice

- Length of a chain $x_1 \leq x_2 \leq \ldots \leq x_k$ is k
- Let K = max over lengths of all the chains in a semilattice
- Height of the semilattice = $K - 1$

Knaster-Tarski Fixed Point Theorem

- Let f be a monotonic function on a complete lattice $(S, \bigwedge, \bigvee)$. Define
  - $\text{red}(f) = \{v \mid v \in S, f(v) \leq v\}$, pre fix-points
  - $\text{ext}(f) = \{v \mid v \in S, f(v) \geq v\}$, post fix-points
  - $\text{fix}(f) = \{v \mid v \in S, f(v) = v\}$, fix-points

  Then,
  - $\bigwedge \text{red}(f) \in \text{fix}(f)$. Further, $\bigwedge \text{red}(f) = \bigwedge \text{fix}(f)$
  - $\bigvee \text{ext}(f) \in \text{fix}(f)$. Further, $\bigvee \text{ext}(f) = \bigvee \text{fix}(f)$
  - $\text{fix}(f)$ is a complete lattice

Application of Fixed Point Theorem

- $f : S \to S$ is a **monotonic** function
- $(S, \bigwedge)$ is a **finite height** semilattice
- $\top$ is the top element of $(S, \bigwedge)$
- Notation: $f^0(x) = x, f^{i+1}(x) = f(f^i(x)), \forall i \geq 0$
- The greatest fixed point of f is

$$f^k(\top), \text{ where } f^{k+1}(\top) = f^k(\top)$$

Fixed Point Algorithm

```
// monotonic function f on a meet semilattice
x := ⊤;
while (x ≠ f(x)) x := f(x);
return x;
```