## 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

- Poset, Lattice, and Data Flow Frameworks: Review
- Connecting Tarski Lemma with Data Flow Analysis
- Soutions of Data Flow Analysis constraints

## 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, $\bigvee \text{ext}(f) = \bigwedge \text{fix}(f)$
  - $\text{fix}(f)$ is a complete lattice

## Application of Fixed Point Theorem

- $f : S \rightarrow S$ 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;
```

## Resemblance to Iterative Algorithm (Forward)

$\mathrm{OUT}[\mathit{Entry}] = \mathtt{Info}_{\mathit{Entry}}$;
for (other blocks $B$) $\mathrm{OUT}[B] = \top$;
while (changes to any OUT) {
    for (each block $B$) {
        $\mathrm{IN}(B) = \bigwedge_{P \in \mathrm{PRED}(B)} \mathrm{OUT}(P)$;
        $\mathrm{OUT}(B) = f_B(\mathrm{IN}(B))$;
    }
}

## Iterative Algorithm

- $f_B(X) = X - \mathrm{KILL}(B) \cup \mathrm{GEN}(B)$
- Backward:
  - Swap IN and OUT everywhere
  - Replace *Entry* by *Exit*
  - Replace predecessors by successors
- In other words: just “invert” the flow graph!!

## Acknowledgement

Rest of the slides based on the material at
`http://infolab.stanford.edu/~ullman/dragon/w06/w06.html`

## Solutions

- **IDEAL solution** = meet over all executable paths from entry to a point (ignore unrealizable paths)
- **MOP** = meet over all paths from entry to a given point, of the transfer function along that path applied to $\text{Info}_{Entry}$.
- **MFP** *(maximal fixedpoint)* = result of iterative algorithm.

## Maximum Fixedpoint

- **Fixedpoint** = solution to the equations used in the iteration:

$$\text{IN}(B) = \bigwedge_{P \in \text{PRED}(B)} \text{OUT}(P)$$

$$\text{OUT}(B) = f_B(\text{IN}(B))$$

- **Maximum Fixedpoint** = any other solution is $\leq$ the result if the iterative algorithm (MFP)
- $\leq$: carries less information.

## MOP and IDEAL

- All solutions are really meets of the result of starting with $\text{Info}_{Entry}$ and following some set of paths to the point in question.
- If we don't include **at least the IDEAL paths**, we have an error.
- But try not to include too many more.
- Less "ignorance," but we "know too much."

## MOP Versus IDEAL

- Any solution that is $\leq$ IDEAL accounts for all executable paths (and maybe more paths)
  - and is therefore conservative (safe)
  - even if not accurate.

## MFP vs MOP

- If MFP $\leq$ MOP?
  - If so, then MFP $\leq$ MOP $\leq$ IDEAL, therefore MFP is safe.
- Yes, but . . .
- Requires two assumptions about the framework:
  - "Monotonicity."
  - Finite height
    no infinite chains $\ldots < x_2 < x_1 < x < \ldots$

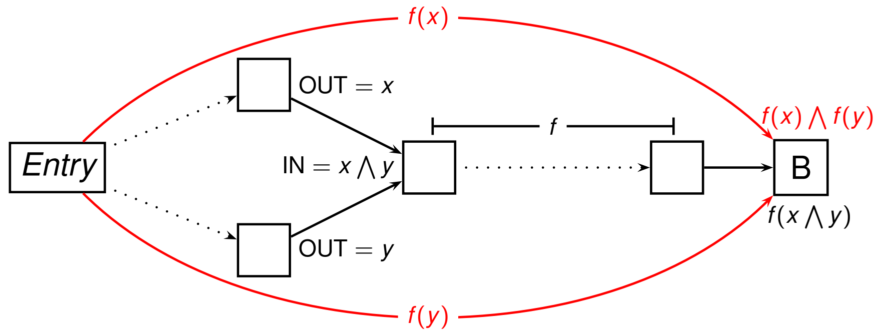
## MFP vs MOP

- **Intuition**: If we computed the MOP directly, we would compose functions along all paths, then take a big meet.
- But the MFP (iterative algorithm) alternates compositions and meets arbitrarily.

## Good News

- The frameworks we've studied so far are all monotone.
  - Easy proof for functions in Gen-Kill form.
- And they have finite height.
  - Only a finite number of defs, variables, etc. in any program.

## Two Paths to *B* that Meet Early




- MOP considers paths independently and combines at the last possible moment.
- In MFP, Values x and y get combined too soon.
- Since $f(x \bigwedge y) \leq f(x) \bigwedge f(y)$, it is as we added non-existent paths.

## Distributive Frameworks

- Distributivity:

$$f(x \bigwedge y) = f(x) \bigwedge f(y)$$

- Stronger than Monotonicity
  - Distributivity $\Rightarrow$ Monotonicity
  - But the reverse is not true

## Even More Good News!

- The 4 example frameworks are distributive.
- If a framework is distributive, then combining paths early doesn't hurt.
  - MOP = MFP.
  - That is, the iterative algorithm computes a solution that takes into account all and only the physical paths.