Program Analysis

https://www.cse.iitb.ac.in/~karkare/cs618/

Foundations of Data Flow Analysis (contd …)

Amey Karkare
Dept of Computer Science and Engg
IIT Kanpur
Visiting IIT Bombay
karkare@cse.iitk.ac.in
karkare@cse.iitb.ac.in

Acknowledgement

- Slides based on the material at http://infolab.stanford.edu/~ullman/dragon/w06/w06.html

Knaster-Tarski Fixed Point Theorem

Let $f: S \to S$ be a monotonic function on a complete lattice $(S, \vee, \wedge)$. Define

- $red(f) = \{v | v \in S, f(v) \leq v\}$, pre fix-points
- $ext(f) = \{v | v \in S, f(v) \geq v\}$, post fix-points
- $fix(f) = \{v | v \in S, f(v) = v\}$, fix-points

Then,

- $\wedge\, red(f) \in fix(f)$, $\wedge\, red(f) = \wedge\, fix(f)$
- $\vee\, ext(f) \in fix(f)$, $\vee\, ext(f) = \vee\, fix(f)$
- $fix(f)$ is a complete lattice

Application of Fixed Point Theorem

- $f: S \to S$ a monotonic function
- $(S, \wedge)$ is a finite height semilattice,
- T is top element
- $f^0(x) = x, f^{i+1}(x) = f\left(f^i(x)\right), i \geq 0$
- The greatest fixed point of f is $f^k(\mathrm{T})$ where $f^{k+1}(\mathrm{T}) = f^k(\mathrm{T})$

Fixed Point Algorithm

```
// monotonic f on a meet semilattice
x := ⊤ ;

while (x != f(x))     x := f(x);

return x;
```

Resemblance to Iterative Algorithm (Forward)

```
OUT[entry] = Info_ENTRY;
for (other blocks B) OUT[B] = ⊤;
while (changes to any OUT) {
  for (each block B) {
    IN(B) = ∧ predecessors P of B OUT(P);
    OUT(B) = f_B(IN(B));
  }
}
```

$$IN(B) = \wedge_{\text{predecessors P of B}} OUT(P);$$

$$OUT(B) = f_B(IN(B));$$

Iterative Algorithm

- $f_B(x) = X - kill(B) \cup gen(B)$
- BACKWARD:
  - Swap IN and OUT everywhere
  - Replace ENTRY by EXIT
  - Replace predecessors by successors
- In other words
  - just “invert” the flow graph!!

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 $\text{Info}_{\text{ENTRY}}$.
- MFP (*maximal fixedpoint*) = result of iterative algorithm.

Maximum Fixedpoint

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

  $IN(B) = \wedge_{\text{predecessors P of B}} OUT(P);$

  $OUT(B) = f_B(IN(B));$

- *Maximum* = any other solution is $\leq$ the result of the iterative algorithm (MFP).

MOP and IDEAL

- All solutions are really meets of the result of starting with $\text{Info}_{\text{ENTRY}}$ 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 Versus MOP --- (1)

- Is MFP $\leq$ MOP?
  - If so, then MFP $\leq$ MOP $\leq$ IDEAL, therefore MFP is safe.
- Yes, but … requires two assumptions about the framework:
  1. "Monotonicity."
  2. *Finite height*

     no infinite chains … $< x_2 < x_1 < x < \ldots$

MFP Versus MOP --- (2)

- 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.

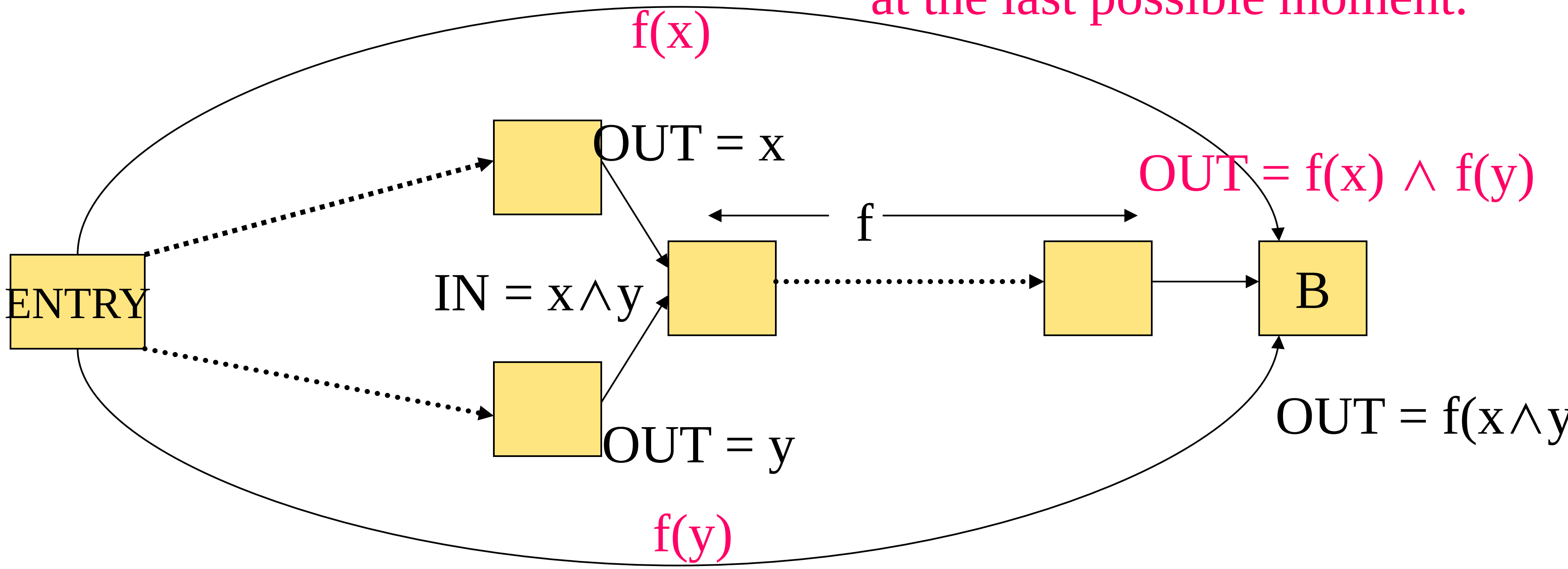
Good News!

- 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

In MFP, Values x and y get combined too soon.

MOP considers paths independently and combines at the last possible moment.



Since $f(x \wedge y) \leq f(x) \wedge f(y)$, it is as if we added nonexistent paths.

Distributive Frameworks

- Distributivity:

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

- Stronger than monotonicity
  - Distributivity $\Rightarrow$ monotonicity
  - But 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.