Data-Flow Frameworks

Lattice-Theoretic Formulation

Meet-Over-Paths Solution

Monotonicity/Distributivity

Data-Flow Analysis Frameworks

- Generalizes and unifies each of the DFA examples from previous lecture.
- Important components:
  1. *Direction* D: forward or backward.
  2. *Domain* V (possible values for IN, OUT).
  3. *Meet operator* $\wedge$ (effect of path confluence).
  4. *Transfer functions* F (effect of passing through a basic block).

Gary Kildall

- This theory was the thesis at U. Wash. of Gary Kildall.
- Gary is better known for CP/M, the first real PC operating system.
- There is an interesting story.
  - Google query: kildall cpm
  - www.freeenterpriseland.com/BOOK/KILDALL.html

Semilattices

- V and $\wedge$ form a *semilattice* if for all x, y, and z in V:
  1. $x \wedge x = x$ (*idempotence*).
  2. $x \wedge y = y \wedge x$ (*commutativity*).
  3. $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ (*associativity*).
  4. *Top* element $\top$ such that for all x, $\top \wedge x = x$.
  5. *Bottom* element (optional) $\perp$ such that for all x, $\perp \wedge x = \perp$.

Example: Semilattice

- V = power set of some set.
- $\wedge$ = union.
- Union is idempotent, commutative, and associative.
- What are the top and bottom elements?

Partial Order for a Semilattice

- Say $x \leq y$ iff $x \wedge y = x$.
- Also, $x < y$ iff $x \leq y$ and $x \neq y$.
- $\leq$ is really a partial order:
  1. $x \leq y$ and $y \leq z$ imply $x \leq z$ (proof in text).
  2. $x \leq y$ and $y \leq x$ iff $x = y$. Proof: $x \wedge y = x$ and $y \wedge x = y$. Thus, $x = x \wedge y = y \wedge x = y$.

Axioms for Transfer Functions

1. F includes the identity function.
   - Why needed? Constructions often require introduction of an empty block.
2. F is closed under composition.
   - Why needed?
     - The concatenation of two blocks is a block.
     - Transfer function for a block can be constructed from individual statements.

Good News!

- The problems from the last lecture fit the model.
  - RD's: Forward, meet = union, transfer functions based on Gen and Kill.
  - AE's: Forward, meet = intersection, transfer functions based on Gen and Kill.
  - LV's: Backward, meet = union, transfer functions based on Use and Def.

Example: Reaching Definitions

- Direction D = forward.
- Domain V = set of all sets of definitions in the flow graph.
- $\wedge$ = union.
- Functions F = all "gen-kill" functions of the form $f(x) = (x - K) \cup G$, where K and G are sets of definitions (members of V).

Example: Satisfies Axioms

- Union on a power set forms a semilattice (idempotent, commutative, associative).
- Identity function: let K = G = $\emptyset$.
- Composition: A little algebra.

Example: Partial Order

- For RD's, $S \leq T$ means $S \cup T = S$.
- Equivalently $S \supseteq T$.
  - Seems "backward," but that's what the definitions give you.
- Intuition: $\leq$ measures "ignorance."
  - The more definitions we know about, the less ignorance we have.
  - $\top$ = "total ignorance."

DFA Frameworks

- (D, V, $\wedge$, F).
- A flow graph, with an associated function f_B in F for each block B.
- A boundary value v_{ENTRY} or v_{EXIT} if D = forward or backward, respectively.

Iterative Algorithm (Forward)

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

$OUT[entry] = v_{ENTRY};$

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

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

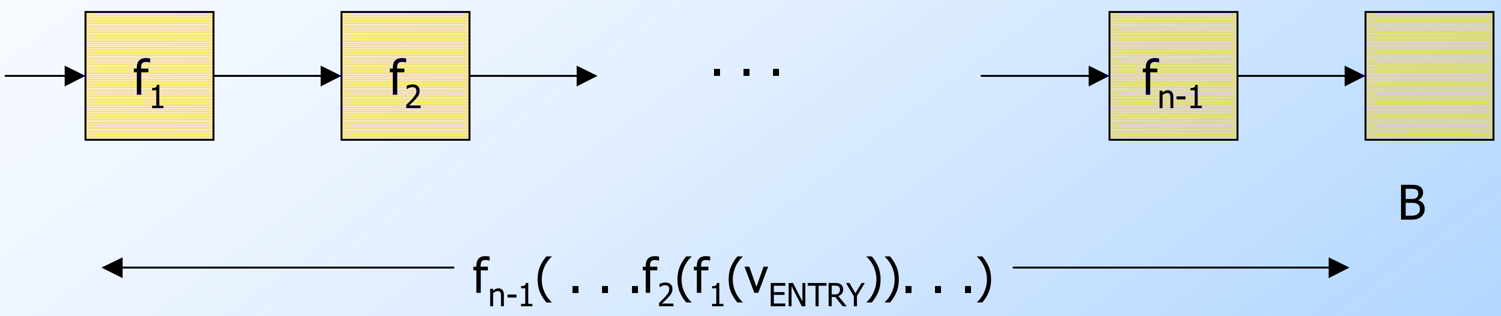
Iterative Algorithm (Backward)

- Same thing --- just:
  1. Swap IN and OUT everywhere.
  2. Replace entry by exit.

What Does the Iterative Algorithm Do?

- MFP (*maximal fixedpoint*) = result of iterative algorithm.
- MOP = meet over all paths from entry to a given point, of the transfer function along that path applied to v_{ENTRY}.
- IDEAL = ideal solution = meet over all executable paths from entry to a point.

Transfer Function of a Path



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 v_{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 --- (1)

- At each block B, MOP[B] $\leq$ IDEAL[B].
  - I.e., the meet over many paths is $\leq$ the meet over a subset.
  - Example: $x \wedge y \wedge z \leq x \wedge y$ because $x \wedge y \wedge z \wedge x \wedge y = x \wedge y \wedge z$.
- Intuition: Anything not $\leq$ IDEAL is not safe, because there is some executable path whose effect is not accounted for.

MOP Versus IDEAL --- (2)

- Conversely: 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 since MOP $\leq$ IDEAL, we have MFP $\leq$ IDEAL, and therefore MFP is safe.
- Yes, but … requires two assumptions about the framework:
  1. "Monotonicity."
  2. *Finite height* (no infinite chains $\ldots < x_2 < x_1 < x$).

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.

Monotonicity

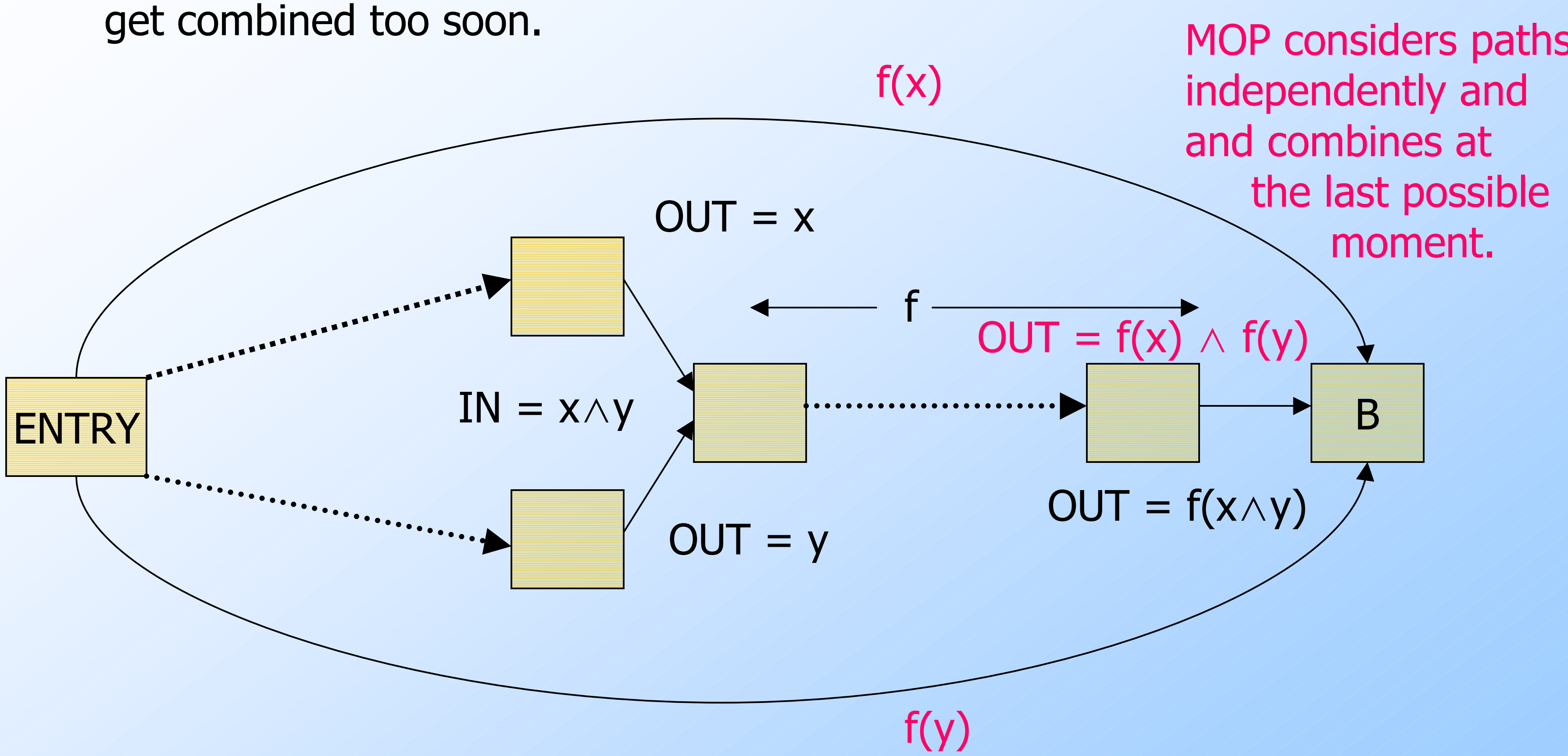
- A framework is *monotone* if the functions respect $\leq$. That is:
- If $x \leq y$, then $f(x) \leq f(y)$.
- Equivalently: $f(x \wedge y) \leq f(x) \wedge f(y)$.
- Intuition: it is conservative to take a meet before completing the composition of functions.

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.



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

Distributive Frameworks

- Strictly stronger than monotonicity is the *distributivity* condition:

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

Even More Good News!

- All the Gen-Kill 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.