Iterative Data-Flow Frameworks (Objectives)

- Given a data-flow problem, the student will be able to formulate it using an iterative data-flow framework.
- Given a data-flow problem, the student will be able to determine if the iterative solution gives the maximal fixed point.
- The student will be able to describe what makes an iterative data-flow problem halt, give a maximal fixed point and be "rapid".

Iterative Data-flow Frameworks

- An iterative data-flow framework has four elements (G,L,F,M)
  1. G is a control-flow graph
  2. L is a semi-lattice that represent the facts being derived (e.g., availability, liveness). A semi-lattice is a triple $(S,\wedge,\perp)$, where S is a set, $\wedge$ is an operator defined over S (called the **meet** operator) and $\perp$ is a designated element in S (called the **bottom** element of L). There is also a **top** element, T.
  3. F is a function space, $F:L \rightarrow L$. The analyzer uses functions in F to model the transmission of values in L along the edges of G and through the nodes of G
  4. M is a map that takes the nodes and edges in G and maps them to specific functions in F

Semi-lattice Properties

- If L is a semi-lattice, the $\wedge$ must be idempotent, commutative and associative when applied to elements of S. $\forall$ a, b, c $\in$ S

  $a \wedge a = a$ (idempotent)

  $a \wedge b = b \wedge a$ (commutative)

  $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (associative)

- The meet operator imposes a partial order on L. For any a, b $\in$ S.

  $a \geq b \Leftrightarrow a \wedge b = b$

  $a > b \Leftrightarrow a \geq b$ and $a \neq b$

  $a \wedge \perp = \perp$

  $a \wedge T = a$

Examples

- Available Expressions
  - Let U be the set of all expressions in a procedure.

    $L = (P(U), \cap, \emptyset)$, $T = U$
    $\forall\ b \in G$, $f_b \in F$, $f_b = GEN(b) \cup (IN(b) \cap PRSV(b))$
- Live-variable Analysis
  - Let U be the set of all variables in a procedure

    $L = (P(U), \cup, U)$, $T = \emptyset$
    $\forall\ b \in G$, $f_b \in F$, $f_b = GEN(b) \cup (IN(b) \cap PRSV(b))$

Iterative Solution (forward problems)

OUT(ENTRY) = $f_b(\perp)$
for each $b \in G$, $b \neq$ ENTRY
    OUT(b) = $f_b(T)$
repeat {
    for each $b \in G$ {
        IN(b) = $\wedge_{p \in pred(b)}$ OUT(p)
        OUT(b) = f_b(IN(b))
    }
} until no change in any OUT(b)

- swap IN(b) and OUT(b), change pred to succ to get backward problem solution

Termination

- Def^n: A chain is a sequence that can result from a series of meet operations: a sequence $x_1, x_2, x_3, \ldots, x_n$, where $x_i \in \mathrm{L}$, $1 \le i \le n$ and $x_i > x_{i+1}$, $1 \le i < n$.
- All chains in L must be bounded by some integer d. This is called the finite descending chain property.
- Def^n: A function f is monotone if $\forall x, y \in \mathrm{L}$

  $$f(x \wedge y) \le f(x) \wedge f(y)$$
  $$x \le y \rightarrow f(x) \le f(y)$$
- If a data flow problem has the finite descending chain property and all functions are monotone, the iterative algorithm must halt.

Termination

1. There are a finite number of nodes in *G*.
2. A bounded semi-lattice means a value can be lowered on the lattice only a finite number of times
3. Monotonic functions mean a lower value produces a lower result. Since the meet operations only lowers values, result can only move down the lattice.

Example

- Show that the iterative algorithm will halt for available expressions.

Good Solutions

- How good are the solutions obtained by the iterative algorithm?
- Consider a path $p = b_0 \rightarrow b_1 \rightarrow \ldots \rightarrow b_k$ with the transfer function

$$f_p(a) = f_{k-1}(f_{k-2}(\ldots f_1(f_0(a))\ldots))$$

  at the beginning of b_k. We define the meet over all paths (MOP) as

$$MOP(b_k) = \wedge_{p \in path(b)} f_p(T)$$

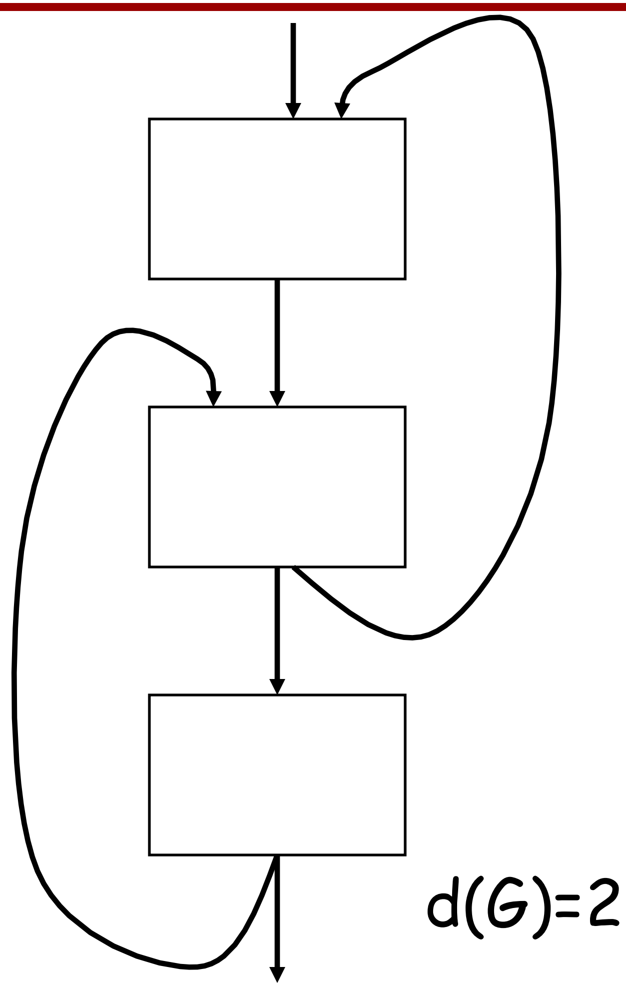
  where $path(b_k)$ is the set of all paths from the entry to b_k (possibly infinite)
- MOP is the best that we can do

Good Solutions

- Def^n: A data-flow framework is distributive if
  $f(a \wedge b) = f(a) \wedge f(b)$
- If a framework is distributive, then the MOP can be computed by the iterative algorithm.
  - at each point where the meet is applied no extra information is lost
  - So, the iterative algorithm does not lose information by applying the meet at each join point instead of considering all paths
- A framework that is only monotone does not have this property

Complexity

- Def^n: Let d(G) be the loop interconnectedness of G. This is the number of consecutive back arcs that can be followed without repeats.
- The iterative algorithm can solve distributive problems in d(G) + 2 iterations.
- These are called rapid problems



Constant Propagation – A Non-distributive Framework

- Let U be the set of the constant status of all variables.

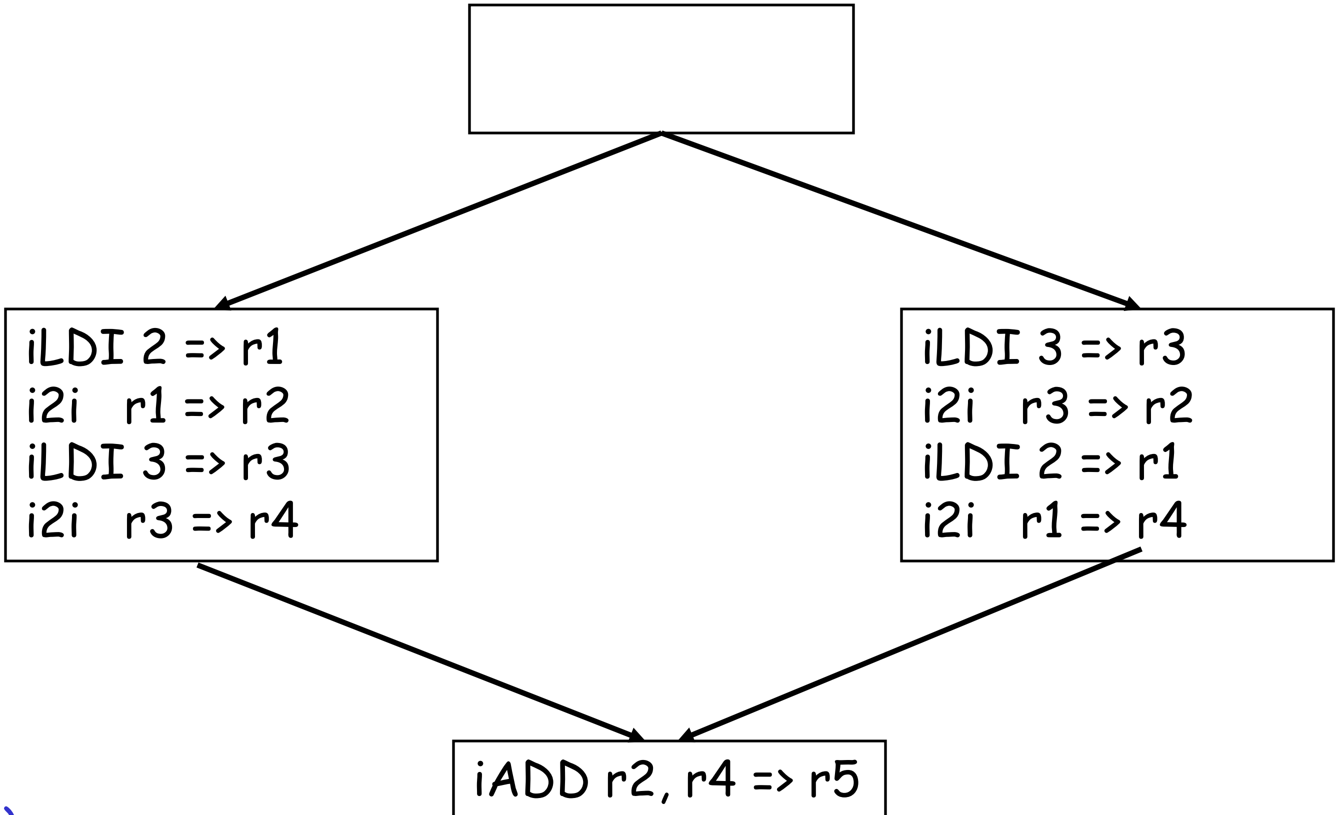
  L = (P(U),$\wedge$,notconst)

  T = undef

  a $\wedge$ b = a, if a = b

  a $\wedge$ b = $\perp$, otherwise

Example



$f(a \wedge b) =$

$f(a) \wedge f(b) =$

Why More Than d(G) + 2?

- Can the constantness of one variable depend upon the constantness of another?
- Can the liveness of one variable depend upon the liveness of another in live-variable analysis?
- When one variable depends upon another we can iterate d(G) + 2 times for the first variable and then the second variable's status may change. This may require more iterations to propagate this information in G.