Dataflow Analysis - 2

- **Monotone dataflow frameworks**
  - Definition
  - Convergence
  - Safety
- **Relation of MOP to MFP**
  - Constant propagation
- **Categorization of dataflow problems**

Monotone Dataflow Frameworks

- **Formalism for expressing and categorizing data flow problems** (Kildall, POPL'73) **<G, L, F, M>**
  - G, flowgraph <N, E, ρ>
  - L, (semi-)lattice with meet $\wedge$
    - usually assume L has a 0 and 1 element
    - finite chains
  - F, function space, $\forall f \in F, f: L \dashrightarrow L$
    - Contains identity function
    - Closed under composition $\forall f, g \in F, f \circ g \in F$
    - Closed under pointwise meet, if $h(x) = f(x) \wedge g(x)$ then $h \in F$
  - M : E --> F, maps an edge to a corresponding transfer function that describes data flow effect of traversing that edge

Monotone DF Frameworks

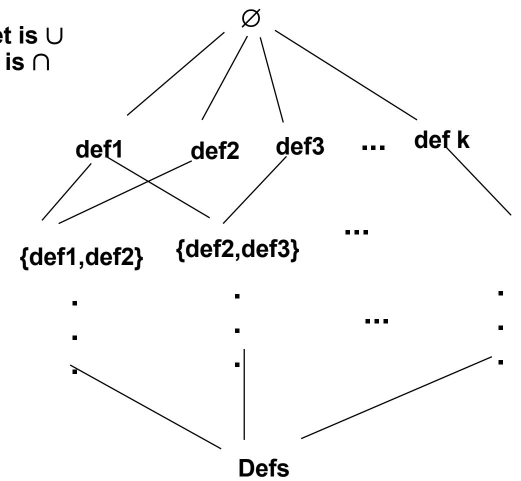
- Desirable to have F contain *monotone* functions so can used fixed point iteration as solution procedure for dataflow equations
- Can ensure finite chains by L being of finite height or a finite lattice
- Often use L as a lattice instead of a semi-lattice
- Kildall's meet semilattice formulation leads to some *unnatural* lattice formulations of problems
  - Always starting at a 1 of meet semilattice and taking meets down to a maximal fixed point (MFP)

Reaching Defns

- e.g., REACH meet operation is set union with partial order is $\supseteq$ superset inclusion
  - Why? recall that the 0 element a is such that $a \leq x = a$, $\forall$ x which means a is a superset of x!
- Defs = {<node,var>}, all defs in program
- Basis lattice is 2^{Defs}
- Cartesian product lattice = (2^{Defs}, 2^{Defs} ..., 2^{Defs})
- partial order on product is ≤' = ≤ component-wise
- 1 element (∅,…, ∅)
- 0 element (Defs, Defs, ..., Defs)

Reaching Defns

meet is ∪
join is ∩



Apply fixed point theorem to find max fixed point on the cross product of this lattice. But this all seems "upside down".

Alternative View

- Form natural basis lattice for a join problem
- Find minimum fixed point on this lattice
- Can consider literature discussions of meet semilattices and then use dual results for the corresponding natural join semilattices
- Means find max fixed point for AVAIL, VeryBusy and constant propagation whereas find min fixed point for REACH, LIVE

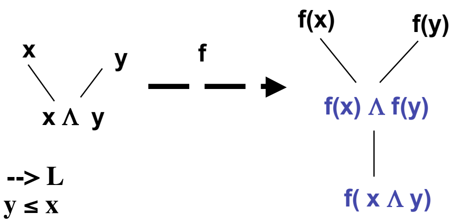
Meet vs Join Semilattice

- Meet semilattice has Λ operation; if finite, has a 0 element,
  - e.g., AVAIL with meet $\cap$ and order $\subseteq$
  - Initialization for
    - forward df problem is $X_\rho = 0$ element and for node $n \neq \rho$, $X_n = 1$ element.
    - backward df problem is $X_{exit} = 0$, all other nodes n, $X_n = 1$
- Join semilattice has υ operation; if finite, has 1 element,
  - e.g., LIVE with join $\cup$ and order $\subseteq$
  - Initialization for
    - forward df problem is $X_\rho = 0$ element and for node $n \neq \rho$, $X_n = 0$ element
    - backward df problem is $X_{exit} = 0$, all other nodes n, $X_n = 0$

Function Properties

- *Montonicity*
  - Defined as $x \leq y \Rightarrow f(x) \leq f(y)$.
  - Equivalent formulation of definition
    $f(x \Lambda y) \leq f(x) \Lambda f(y)$
- *Distributivity*
  - If $f(x \Lambda y) = f(x) \Lambda f(y)$ then f *distributive*
  - Distributivity implies monotonicity
  - Four classical bitvector problems are distributive

Monotonicity



f: L --> L
$x \wedge y \le x$
$x \wedge y \le y$
by defn of meet of x, y; and

$f(x) \wedge f(y) \le f(x)$
$f(x) \wedge f(y) \le f(y)$
by defn of meet of f(x), f(y).

$f(x \wedge y) \le f(x)$
$f(x \wedge y) \le f(y)$
by monotonicity of f

$f(x \wedge y) \le f(x) \wedge f(y)$
by defn meet of f(x), f(y)

Therefore, $x \le y \Rightarrow f(x) \le f(y)$ (1) implies $f(x \wedge y) \le f(x) \wedge f(y)$ (2).

Monotonicity, cont.

Show $f(x \wedge y) \le f(x) \wedge f(y)$ (2) implies $x \le y \Rightarrow f(x) \le f(y)$ (1)
Then we know these two definitions of monotonicity are equivalent.

Assume $x \le y$. Then $x \wedge y = x$ by defn of meet.

$f(x \wedge y) = f(x) \le f(x) \wedge f(y)$ which is given.

Then $f(x) \wedge (f(x) \wedge f(y)) = f(x)$ by defn of meet.

But $(f(x) \wedge f(x)) \wedge f(y) = f(x) \wedge f(y) = f(x)$ by associativity of meet

Therefore, $f(x) \le f(y)$ by defn of meet.

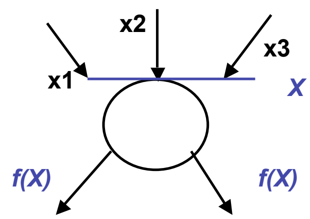
So (2) implies (1).

Therefore, these definitions of monotonicity are equivalent.

Function Properties

- Relation of function space properties to fixed point iterative algorithms for DFA
- *Distributivity* means you can take meets in the domain and then apply f OR you can apply f and then take meets in the range -- you calculate the same answer for these functions

$f(x1 \wedge x2 \wedge x3) = f(x1) \wedge f(x2) \wedge f(x3)$



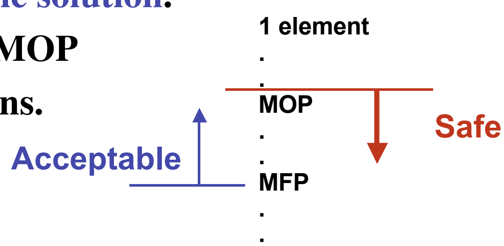
MOP vs MFP

- For *distributive* functions define the DF problem, to obtain data flow solution at node n, can gather information on paths (e.g., P1, P2) simultaneously without loss of precision.
  - e.g., $f_{P1}(0)$, $f_{P2}(0)$ needn't be calculated explicitly
- However, Kam and Ullman showed that this is not true for all *monotone* functions; Kam, Ullman, 1976,1977
- Therefore, MFP only approximates MOP for general monotone functions that are not distributive.

Safety of Dataflow Solution

- Safe solution underestimates the actual dataflow solution; $x \leq MOP$ is an approximate solution
- Acceptable solution is one that contains a fixed point of the function, $y \geq z$ where z is any fixed point.
- If they exist, MOP is largest safe solution and MFP is smallest acceptable solution.
- Between MFP and MOP

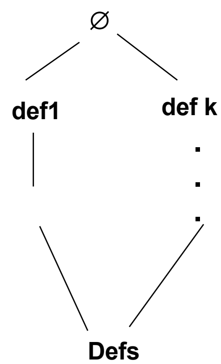
are interesting solutions.

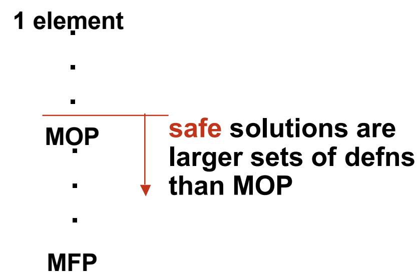


Safe Solutions

- AVAIL is meet semilattice; it is safe to err by saying an expression is NOT AVAILABLE when it might be. This inhibits *cse* transformations. Therefore, safe solutions are smaller than MOP here.
- REACH is join semilattice; it is safe to err by saying a definition reaches when it DOES NOT REACH. This inhibits dead code elimination transformations. Therefore, safe solutions are larger than MOP here.

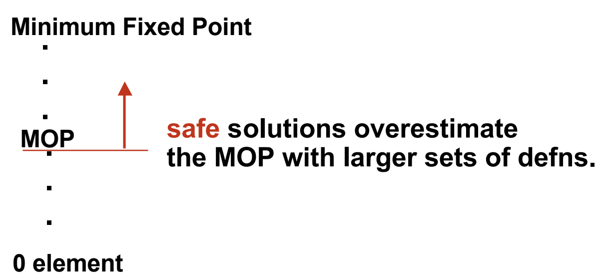
Reaching Defns - view 1





Meet semilattice formulation

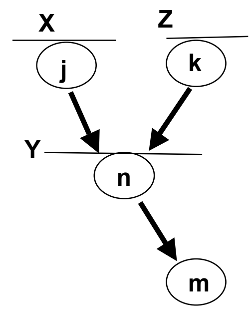
Reaching Defns - view 2



Join semilattice formulation

Nodes vs Edges Formulations

On Edges (MOP formulation) for Reaching Defs



Effect of path <j,n,m> is

$M(n,m) \circ M(j,n)(X) = f_n \wedge f_j(X)$ where $X \subseteq$ Defs

Dataflow information at node m is
$M(n,m) \circ M(j,n)(X) \wedge M(n,m) \circ M(k,n)(Z) =$

$f_n \circ f_j(X) \wedge f_n \circ f_k(Z) = f_n(f_j(X)) \wedge f_n(f_k(Z)) =$

$f_n(f_j(X) \wedge f_k(Z))$ by *distributivity* of f;

Therefore by nodes (df equations) and by edges (MOP-like) formulations are equivalent here

Kam and Ullman Results

- **On monotone data flow framework, iterative algorithms converges to MFP of dataflow equations**
- **A monotone problem that is not distributive for which MOP≠MFP is constant propagation**
- **MFP ≤ MOP always**
- **There is no way by using subroutines to apply functions to lattice elements and take meets, one can produce an algorithm to obtain MOP solution for an arbitrary instance of a monotone framework. MOP is undecidable (use a substring identification problem (MPCP) to prove this)**

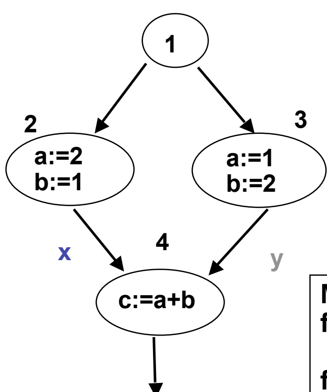
Constant Propagation

- Flowgraph nodes contain **`A:=B op C`** where **`A,B,C`** are vars and op ∈ {+ - / *} or **`A`** := integer constant
- Basis lattice is sets of {<var, integer value>} pairs with partial order of subset inclusion and meet ∩
- Monotone flow functions will be:

  $f_{A:= B\ op\ C}$ (X) = Y, where Y differs from X only in terms of a <A,_> pair.
  - if <B,b>, <C,c> ∈ X, then add <A, b op c> and remove any other tuples <A,_> to form Y
  - otherwise remove <A,_> from X to form Y

  $f_{A\ :=\ r}$ (X) = Y, where Y is formed from X with <A,r> added and all previous <A,_> removed.

Constant Propagation



Dataflow equations formulation
x = {<a,2> <b,1> }
y = {<a,1> <b,2>}
x Λ y = x ∩ y = ∅
therefore, f_4 (x Λ y) = ∅ as well

MOP formulation
f_4 (x) = f_4({<a,2> <b,1>}) = {<c,3> <a,2> <b,1>}

f_4 (y) = f_4({<a,1> <b,2>}) = {<c,3> <a,1> <b,2>}

f_4 (x) Λ f_4 (y) = f_4 (x) ∩ f_4 (y) = {<c,3>}

These are different answers!!
So the f's are not distributive functions.

Heuristic Fix

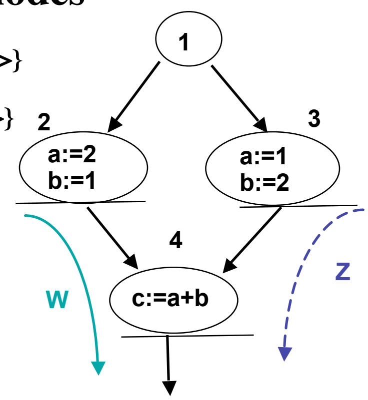
- **Setup df eqns at exit of nodes**

$W = f_4(\{\langle a,2\rangle\ \langle b,1\rangle\}) = \{\langle c,3\rangle\ \langle a,2\rangle\ \langle b,1\rangle\}$

$Z = f_4(\{\langle a,1\rangle\ \langle b,2\rangle\}) = \{\langle c,3\rangle\ \langle a,1\rangle\ \langle b,2\rangle\}$

Constants on exit of node 4 =
$W \wedge Z = \{\langle c,3\rangle\}$

This only shows one can get better approximations to MOP, but this trick of solving on exit of nodes, doesn't always work.



Iteration and Convergence Properties

- ***Monotonicity*** **and** ***distributivity*** **guarantee the existence of a fixed point solution and affect its precision when solved by fixed point iteration**
- **Orthogonal function properties govern the speed of convergence of the iteration**

Convergence Properties

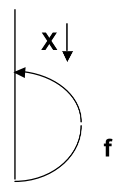
K-boundedness

Monotone function space is *k-bounded* if

$\forall f \in F, f^k = i \wedge f \wedge f^2 \wedge \ldots \wedge f^{k-1}$

where *i* is the identity function and this function is evaluated pointwise on the lattice.

- **2-bounded is termed *fast*** (Graham-Wegman 1981)



k-boundedness

- **Intuitively, if f describes dataflow effect of once around the cyclic path, then**

  $x \wedge f(x) \wedge f(f(x)) \wedge \ldots$

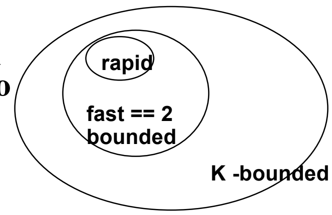
  is the effect of a loop where the number of iterations is *a priori* indeterminate.
- **For fast functions $f^2 \geq i \wedge f$. This means**

  $f^2(x) \geq i(x) \wedge f(x) = x \wedge f(x)$

Therefore, $x \wedge f(x) \wedge f^2(x) = x \wedge f(x)$ so you needn't apply f twice to find out the dataflow effect of executing the loop.

Convergence Properties

K-bounded: all contributions to MFP solution occur prior to Kth iteration
Fast: 1 pass around a cycle is enough to summarize its contribution to the dataflow solution (e.g., reflexive transitive closure is fast but not rapid)
Rapid: contribution of a cycle is independent of value at entry node; 1 pass around the cycle is enough. All classical bitvector problems are rapid
Sometimes, effect of a cycle must be approximated in order to have an effective solution procedure.



Categorizing DF Problems

- **REACH, AVAIL, LIVE, VERYBusy** are all distributive and rapid
- **Reflexive, transitive closure** (assign to a vertex v, the set of all transitive edges ending in v) is distributive and fast, but not rapid
- **Bound Set** (calculates formals linked through a chain of calls) is distributive, *k*-bounded for a particular instance where *k* depends on the length of call chains and permutation orders of parameters in recursive calls (related to aliasing in Fortran)

Categorizing DF Problems

- **Constant propagation is monotone and not distributive, but fast**
- **Interprocedural Must-define is monotone but not distributive**
- **Interprocedural May-modify and Interprocedural Must-preserve are distributive**
- **Pointer aliasing, formulated on an approximation lattice (Weihl, 1981), is monotone**