

So far been talking about various dataflow problems (e.g. reaching definitions, live variable analysis) in very informal terms. Now we will discuss a more fundamental approach to handle many of the dataflow analysis problems in an uniform manner

We define a dataflow framework.

A dataflow framework consists of:

- 1) a semi lattice  $(L, \wedge)$  which includes a domain of values  $V$  and a meet operator (confluence operator)
- 2) a family  $F$  of transfer functions:  $F = \{f \mid f : V \rightarrow V\}$
- 3) a direction of the data flow; either FORWARD or BACKWARD

## Semi lattice

A semi lattice is a pair  $(L, \wedge)$  where:

- $L$  is a non empty set of values  $V$
- $\wedge$  is a binary meet operator on  $L$  such that for all  $x, y, z$  in  $L$ :
  - $x \wedge x = x$  (idempotent)
  - $x \wedge y = y \wedge x$  (commutative)
  - $x \wedge (y \wedge z) = (x \wedge y) \wedge z$  (associative)

a semi lattice has a top element, denoted  $\top$ , such that: for all  $x$  in  $V$ ,  $\top \wedge x = x$

Optionally, a semi lattice may have a bottom element  $\perp$ , such that for all  $x$  in  $V$ ,  $\perp \wedge x = \perp$

**For example:**

**Reaching definitions:**

$L = \{ x \mid x = \text{values at the top of nodes in a flow graph} \} = 2^D$   
(where  $D$  is the set of all defs in the program)

e.g. for reaching defs:  $\text{In}[B] = \{L\}$

$F = \{f \mid f = \text{transfer function of blocks in flowgraphs}\}$

e.g. for for reaching defs:  $F = \{f(x) \mid f(x) = A \cup (X-B)\}$  where  $A, B$  are sets of defs in  $L$  representing Gen and Kill sets respectively.

$\wedge$  = meet operator

e.g. for reaching defs: meet operator is  $\cup$  (union)

### **Available expressions:**

$L = 2^D$      $D$  is the set of all expressions computed by the program

$F = \{f(x) \mid f(x) = A \cup (X-B)\}$  where  $A, B$  are sets of expressions in  $L$  representing Gen and Kill sets respectively.

$\wedge = \cap$  (intersection)

## constant Propagation:

In Reach Def:  $x = x + 1$ ,  $x$  may be redefined  $\Rightarrow$  not constant

In Monotone Data Flow:  $x = x + 1 \Rightarrow \text{const} \Rightarrow \text{const}$

In monotone data flow:

$L = \{x \mid x = \text{function } \psi\}$

$\psi : V \rightarrow \mathbf{R} \cup \{\text{nonconst}, \text{undef}\},$

where  $V = \{\text{set of variables in program}\}$

i.e.  $L$  is a mapping from variable to value, where the value of the variable can be constant (a constant value in  $\mathbf{R}$ ), nonconst (e.g. two possible assignments), or undef (no information is known)

## constant Propagation:

### Transfer function:

$f$  is transfer function of a block (top  $\rightarrow$  bottom) and is created from the type of operation in the flowgraph:

- no definition:  $f$  is simply the identity function
- assignment statement
  - assignment statement to a constant ( $x=c$ ):  $f(m) = m'$ , with  $m'(w) = m(w)$  for all  $w \neq x$  and  $m'(x) = c$
  - assignment statement ( $x=y+z$ ):  $f(m) = m'$  with
    - $m'(w)=m(w)$  for all  $w \neq x$
    - $m'(x) = m(y) + m(z)$  if  $m(y)$  and  $m(x)$  are constant
    - $m'(x) = \text{nonconst}$  if either  $m(y)$  or  $m(z)$  is nonconst
    - $m'(x) = \text{undef}$  otherwise
- read statement ( $\text{read}(x)$ ):  $f(m) = m'$  with  $m'(w) = m(w)$  for all  $w \neq x$ ,  $m'(x) = \text{nonconst}$

**constant Propagation:**

Meet operator:

suppose  $u, v$  in  $L$ , i.e. mappings  $\text{var} \rightarrow \text{value}$ .  $f = u \wedge v$  is function for all variable  $x$  such that  $f(x) = f(u(x), v(x))$ , and can be defined with the following table:

$v(x)/u(x)$	nonconst	c	$d \neq c$	undef
nonconst	nonconst	nonconst	nonconst	Nonconst
c	nonconst	c	nonconst	c
undef	nonconst	c	d	undef

## More formally: Axioms of Dataflow

About  $(L, F, \wedge)$  where:

1.  $F$  is family of transfer functions from  $L$  to  $L$  where:
  - $F$  must contain the identity function  $I(x) = x$
  - $F$  must be closed under composition
2. The meet operator  $\wedge$  has the following properties
  - Associative:  $u \wedge (v \wedge w) = (u \wedge v) \wedge w$
  - Commutative:  $u \wedge v = v \wedge u$
  - Idempotent:  $u \wedge u = u$
3. There is at least at TOP element  $\top$  in  $L$  such that for all  $u$  in  $L$ :  $\top \wedge u = u$

## Example: Reaching defs

*Recall transfer function has identity function and is closed under composition*

Let  $f_1(X) = G_1 \cup (X - K_1)$  and  $f_2(X) = G_2 \cup (X - K_2)$  be two transfer functions in  $F$

Proof Identity function in  $F$ :

Obvious if you pick Gen and Kill sets to be the empty set  $f(X) = X$

Proof closed under composition:

$$\begin{aligned} f_2(f_1(X)) &= G_2 \cup ((G_1 \cup (X - K_1)) - K_2) \\ &= (G_2 \cup (G_1 - K_2)) \cup (X - K_1 \cup K_2) \end{aligned}$$

Now let  $K = K_1 \cup K_2$  and  $G = G_2 \cup G_1 - K_2$  then the composition of  $f_1$  and  $f_2$ , which is  $f(x) = G \cup (x - K)$ , is in  $F$

## Example: available expressions

The meet operator is  $\cap$  (intersection)

The TOP element  $\top$  is the universal set

## Example: constant propagation

- F is closed under composition and has Identity function
- Meet operator (use table to prove):
  - Idempotent  $u \wedge u = u$  (from table diagonal can infer this is true)
  - Commutative  $u \wedge v = v \wedge u$  (from symmetry in table can infer true)
- TOP element  $\top(x)$  is undef for all x and  $\top \wedge u = u$  (last column in table)

## Monotonicity and Distributivity

Definition:  $\leq$  on L:

$$u \leq v \text{ iff } u \wedge v = u \text{ for } u, v \text{ in } L$$

Example: Reaching defs:

$\wedge$  is  $\cup$  (union)

L is sets of definitions in program,  $x, y$  in L

$x \leq y$  means  $x \cup y = x$ , i.e.  $x$  is superset of  $y$

(seems counter intuitive  $x$  superset of  $y$  and still  $x \leq y$ )

Example: Available expressions:

$\wedge$  is  $\cap$  (intersection)

L is sets of all the expressions in the program,  $x, y$  in L

$x \leq y$  means  $x \cap y = x$ , i.e.  $x$  is subset of  $y$

(intuitively makes more sense)

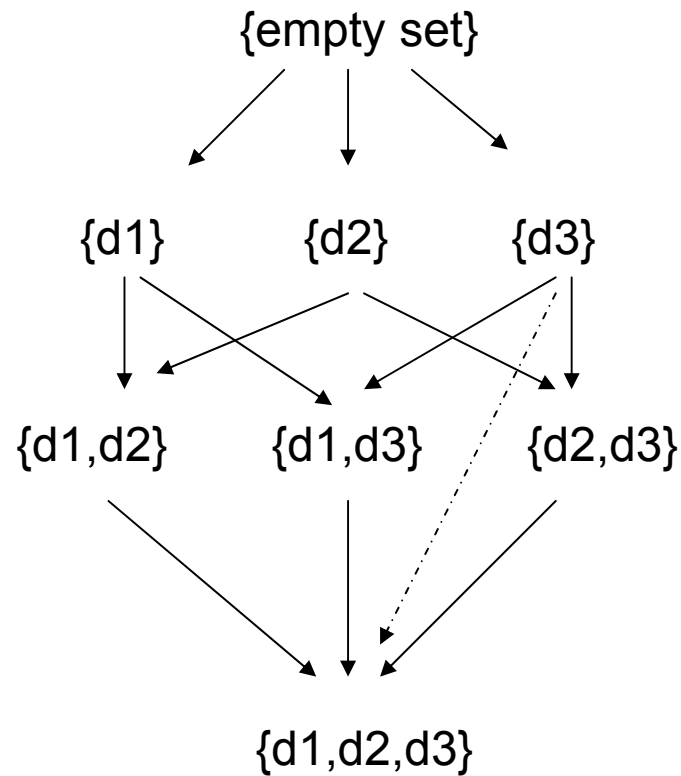
“ $\leq$ ” is called the partial order

$\leq$  is transitive (but e.g. available expressions  $x \cap y = \text{empty} \Rightarrow$  cannot say  $x \leq y$  or  $y \leq x$ )

# Lattice diagram

Arcs from x to y if  $y \leq x$  (y superset of x)

Example: Reaching defs:



$\{d1,d2,d3\} \leq \{d2\}$   
 $\leq \{d1\}$   
:  
:

*Note: transitivity not shown (e.g. dashed arc)*

*cont:*

$$\begin{array}{ccc} \text{e.g. } x = \{d1\} & \longrightarrow & z = \{d1, d2\} \\ y = \{d2\} & & \wedge = U \end{array}$$

From TOP  $\top$  there is a path to every element

Framework  $(L, F, \wedge)$  is monotone iff

1.  $u \leq v \Rightarrow f(u) \leq f(v)$ , for all  $u, v$  in  $L$  and all  $f$  in  $F$ , or equivalent
2.  $f(u \wedge v) \leq f(u) \wedge f(v)$ , for all  $u, v$  in  $L$  and all  $f$  in  $F$

$$(1) \Rightarrow (2)$$

$$(2) \Rightarrow (1)$$

Framework  $(L, F, \wedge)$  is distributive iff

$$f(u \wedge v) = f(u) \wedge f(v), \text{ for all } u, v \text{ in } L \text{ and all } f \text{ in } F$$

Distributive  $\Rightarrow$  monotone

Example: Reaching defs:

x,y are defs

f:  $f(z) = G \cup (z-K)$

$G \cup ((x \cup y) - K) = (G \cup (x-K)) \cup (G \cup (y-K))$

i.e. **distributive**

*Note: not all data analysis frameworks are distributive*

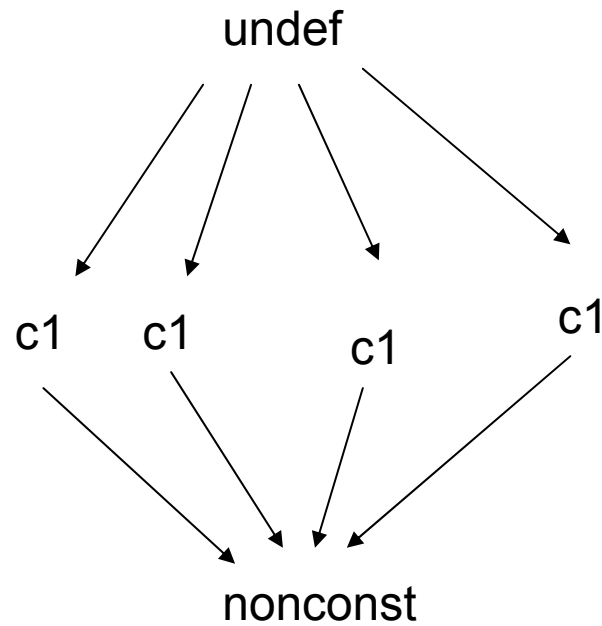
Example: Constant propagation: monotone but not distributive:

using the table again:

- $\text{nonconst} \wedge c = \text{nonconst}$ , for all c
- $c \wedge d = \text{nonconst}$ , for all  $c \neq d$
- $c \wedge \text{undef} = c$ , for all c
- $\text{nonconst} \wedge \text{undef} = \text{nonconst}$
- $x \wedge x$ , for all x in L

i.e.  $f(a) = u(a) \wedge v(a) \Rightarrow$

- $\text{nonconst} \leq c$
- $c \leq \text{undef}$
- $\text{nonconst} \leq \text{undef}$



$u \leq v$  iff  $u(a) \leq v(a)$ , for all  $a$  in  $L$

$u \leq v$  iff  $u(a) = c \Rightarrow v(a) = c$  or  $undef$

$u \leq v \Rightarrow f(u) \leq f(v)$ , i.e. monotone

e.g.  $f: a = b + c$

$u(a)$   $v(b)$  change

$u \leq c$  i.e.  $u(x) \leq v(x)$ , for all  $x \Rightarrow [f(u)](a) \leq [f(v)](a)$

all values of  $u(b)$ ,  $u(c)$ ,  $v(b)$ ,  $v(c)$  with

$u(b) \leq v(b)$  and  $u(c) \leq v(c)$

Example:

$u(b) = nonconst$

$u(c) = 3$

$v(b) = 2$

$v(c) = undef$

$\Rightarrow$

$[f(u)](a) = nc$

$[f(v)](a) = undef$

$\Rightarrow nonconst \leq undef$

not distributive

$$u(b) = 2 \quad v(b) = 3$$

$$u(c) = 3 \quad v(c) = 2$$

$$f = u \wedge v \Rightarrow u(b) \wedge v(b) = 2 \wedge 3 = \text{nonconst}$$

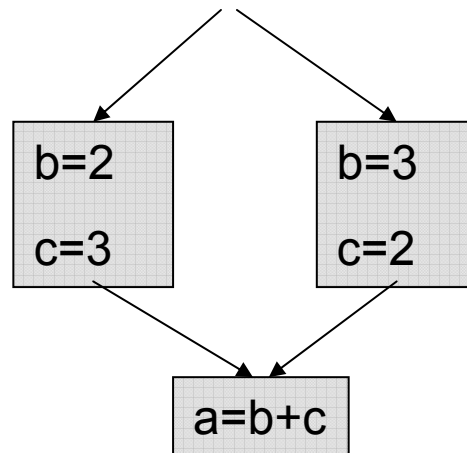
$$u(c) \wedge v(c) = 3 \wedge 2 = \text{nonconst}$$

$$\neg f(b) = f(c) = \text{nonconst} \Rightarrow [f(u)](a) = \text{nonconst}$$

$$[f(u)](a) = 5$$

$$[f(v)](a) = 5 \quad \Rightarrow \quad [f(u) \wedge f(v)] = 5$$

$$f(a) = [f(u \wedge v)](a) \neq [f(u) \wedge f(v)](a) \quad \Rightarrow \quad \text{NOT DISTRIBUTIVE}$$



## Meet over path solutions (MOP)

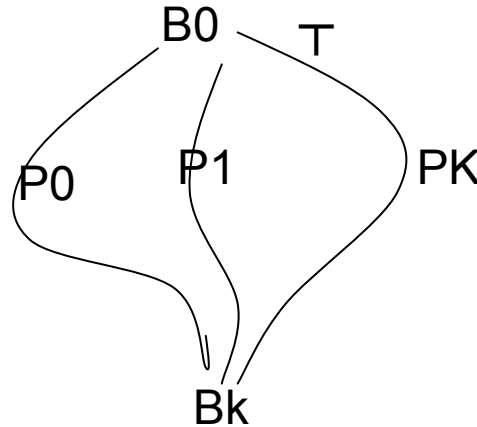
B is Basic block

$f_B$  transfer function

$P = \text{path} = B_0 \rightarrow B_1 \rightarrow \dots \rightarrow B_k$  /  $B_0$  is entry

$f \text{ of path} = f_0 \cdot f_1 \cdot f_2 \cdot \dots \cdot f_{k-1}$

TOP  $\top = \text{no info}$



$MOP(B) = \wedge (\text{paths } P \text{ from } B_0 \text{ to } B) \text{ fp}(\top)$

Conservative solution:  $IN[B] \leq MOP[B]$  for all B

(same paths or subset of paths)

Suppose:

$x = f_p(\top)$  all *real* paths on some execution

$y = f_p(\top)$  on all other paths

$MOP(B) = x \wedge y$

$x = \text{true}$

$MOP(B)$  larger, suppose  $x \wedge y$

$x \wedge y \leq y$

$MOP(B) \leq x$  (true solution)

Decent solution  $\leq MOP(B) \leq$  true solution

easier to obtain if only monotone

if framework is distributive  $\rightarrow$  solution = MOP

## Iterative solution for general frameworks:

Input:  $(L, F, \wedge)$   $f$  in  $F$  for all nodes element of  $G$

Output:  $IN[B]$  in  $L$  for all  $B$  element of  $G$

for each node  $B$  do

$out[B] = f_B[\perp]$

done

while (changes to any  $out[B]$ ) do

    for each  $B$ , in DFS order do

$in[B] = \wedge(\text{all pred } P \text{ of } B) out[P]$

$out[B] = f_B[in[B]]$

    done

endwhile

## properties of iterative alg:

Let P is path from B0 to Bk = B

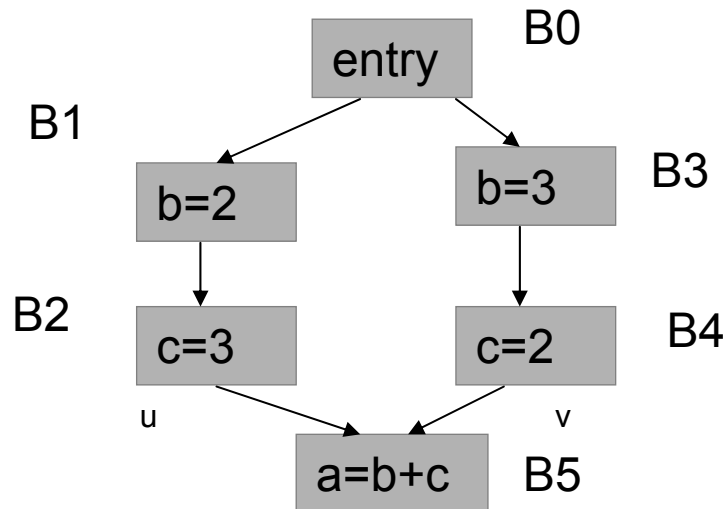
$IN[B] \leq f_p(\top)$  after k iterations for every P

$x \leq y, x \leq z \Rightarrow x \leq y \wedge z$

$IN[B] \leq MOP[B]$  (if monotone)

if distributive:  $IN[B] = MOP[B]$

If only monotone (not distributive) e.g. constant computation  
example:



looks like

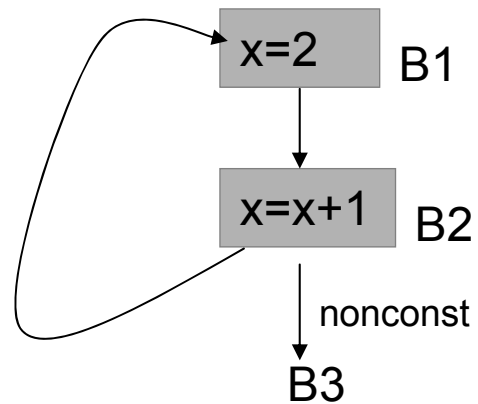
$B0 \rightarrow B1 \rightarrow B4 \rightarrow B5$  and

$B0 \rightarrow B3 \rightarrow B2 \rightarrow B5$  are  
real. so not accurate

## convergence of iterative alg:

if we use only acyclic paths  $2 + \text{depth of } G$  iterations

if we need cyclic paths:



new value  $\leq$  old value

suppose  $n$  nodes and  $v$  variables

OUT[B] can go from undef  $\rightarrow$  c  $\rightarrow$  nonconst

$\Rightarrow$  max number of iterations until nochange  $2*n*v$