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: \(F = \{f | f : V \rightarrow V\}\)
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 \(\bot\), such that for all \(x\) in \(V\), \(\bot \wedge x = \bot\)
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 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.

$^\land = \text{meet operator}$
e.g. for reaching defs: meet operator is $\cup$ (union)
Available expressions:

\[ L = 2^D \quad D \text{ is the set of all expressions computed by the program} \]

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

\[ \wedge = \cap \text{ (intersection)} \]
**constant Propagation:**

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

In Monotone Data Flow:  \( x = x + 1 \) \( \Rightarrow \) const \( \Rightarrow \) const

In monotone data flow:

\[
L = \{ x \mid x = \text{function } \psi \}
\]

\[
\psi : V \rightarrow \mathbb{R} \cup \{\text{nonconst, undef}\},
\]

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 \( \)), nonconst (e.g. two possible assignments), or undef (no information is known)
**constant Propagation:**

**Transfer function:**

f is transfer function of a block (top -> 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 ≠ x and m’(x) = c
  - assignment statement (x=y+z): f(m) = m’ with
    - m’(w)=m(w) for all w ≠ x
    - m’(x) = m(y) + m(z) if m(y) and m(x) are constant
    - m’(x) = nonconst if either m(y) or m(z) is nonconst
    - m’(x) = undef otherwise
- read statement (read(x)): f(m) = m’ with m’(w) = m(w) for all w ≠ x, m’(x) = nonconst
**constant Propagation:**

Meet operator:
suppose $u, v$ in $L$, i.e. mappings var -> value. $f = u \land v$ is function for all variable $x$ such that $f(x) = f(u(x), v(x))$, and can be defined with the following table:

<table>
<thead>
<tr>
<th>$v(x)/u(x)$</th>
<th>nonconst</th>
<th>c</th>
<th>d$\neq$c</th>
<th>undef</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonconst</td>
<td>nonconst</td>
<td>nonconst</td>
<td>nonconst</td>
<td>Nonconst</td>
</tr>
<tr>
<td>c</td>
<td>nonconst</td>
<td>c</td>
<td>nonconst</td>
<td>c</td>
</tr>
<tr>
<td>undef</td>
<td>nonconst</td>
<td>c</td>
<td>d</td>
<td>undef</td>
</tr>
</tbody>
</table>
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:
$f_2(f_1(X)) = G_2 \cup ((G_1 \cup (X-K_1)) - K_2)$
$= (G_2 \cup (G_1-K_2)) \cup (X-K_1 \cup K_2)$
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 $u \land u = u$ (from table diagonal can infer this is true)
  - Commutative $u \land v = v \land u$ (from symmetry in table can infer true)
- TOP element $\top(x)$ is undef for all $x$ and $\top \land u = u$ (last column in table)
**Monotonicity and Distributivity**

Definition: \( \leq \) on \( L \):

\[
u \leq v \iff u \wedge v = u \quad \text{for } u,v \text{ in } L
\]

Example: Reaching defs:

\( \wedge \) is \( U \) (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 \text{ superset of } x)

Example: Reaching defs:

\[
\begin{array}{c}
\{\text{empty set}\} \\
\{d1\} & \{d2\} & \{d3\} \\
\{d1,d2\} & \{d1,d3\} & \{d2,d3\} \\
\{d1,d2,d3\}
\end{array}
\]

\[
\{d1,d2,d3\} \leq \{d2\} \leq \{d1\}
\]

Note: transitivity not shown (e.g. dashed arc)
cont:
e.g. \( x = \{d1\} \)
\( y = \{d2\} \)
\( z = \{d1,d2\} \)
\( ^\ = U \)

From TOP \( \top \) there is a path to every element

Framework \((L,F,^)\) 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. \( f(u^v) \leq f(u) ^ f(v) \), for all \( u,v \) in \( L \) and all \( f \) in \( F \)

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

Framework \((L,F,^)\) is distributive iff

\( f(u^v) = f(u) ^ f(v) \), for all \( u,v \) in \( L \) and all \( f \) 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:
   - 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
   - nonconst \( ^\wedge \ \text{undef} = \text{nonconst} \)
   - \( x \ ^\wedge \ x = \text{nonconst} \), for all x in L

i.e. \( f(a) = u(a) ^\wedge v(a) \Rightarrow \)
   - nonconst \( \leq c \)
   - \( c \leq \text{undef} \)
   - nonconst \( \leq \text{undef} \)
\[ u \leq v \text{ iff } u(a) \leq v(a), \text{ for all } a \in L \]
\[ u \leq v \text{ iff } u(a) = c \Rightarrow v(a) = c \text{ or } \text{undef} \]
\[ u \leq v \Rightarrow f(a) \leq f(v), \text{ i.e. monotone} \]

E.g. \( f: a = b + c \)

\[ u(a) \text{ v(b) change} \]
\[ u \leq c \text{ i.e. } u(x) \leq v(x), \text{ for all } x \Rightarrow [f(u)](a) \leq [f(v)](a) \]

All values of \( u(b), u(c), v(b), v(c) \) with
\[ u(b) \leq v(b) \text{ and } u(c) \leq v(c) \]

Example:
\[ u(b) = \text{nonconst} \]
\[ u(c) = 3 \]
\[ v(b) = 2 \]
\[ v(c) = \text{undef} \]

\[ [f(u)](a) = \text{nc} \]
\[ [f(v)](a) = \text{undef} \]

\[ \Rightarrow \text{nonconst} \leq \text{undef} \]
not distributive

\[ u(b) = 2 \quad v(b) = 3 \]
\[ u(c) = 3 \quad v(c) = 2 \]

\[ f = u \land v \Rightarrow u(b) \land v(b) = 2 \land 3 = \text{nonconst} \]
\[ u(c) \land v(c) = 3 \land 2 = \text{nonconst} \]

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

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

\[ f(a) = [f(u \land v)](a) \neq [f(u) \land f(v)](a) \quad \Rightarrow \text{NOT DISTRIBUTIVE} \]
Meet over path solutions (MOP)

B is Basic block
\( f_B \) transfer function

\[ P = \text{path} = B_0 \to B_1 \to \ldots \to B_k \quad / \ Bo \text{ is entry} \]

\( f \) of path = \( f_0 \cdot f_1 \cdot f_2 \cdot \ldots \cdot f_{k-1} \)

\[ \text{TOP} \; T = \text{no info} \]

\[ \text{MOP}(B) = \bigwedge (\text{paths } P \text{ from } B_0 \text{ to } B) \; f_p(T) \]

Conservative solution: \( \text{IN}[B] \leq \text{MOP}[B] \) for all \( B \)
(same paths or subset of paths)
Suppose:
\[ x = f_p(\top) \text{ all real paths on some execution} \]
\[ y = f_p(\top) \text{ on all other paths} \]

\[ \text{MOP}(B) = x \land y \]
\[ x = \text{true} \]
\[ \text{MOP}(B) \text{ larger, suppose } x \land y \]
\[ x \land y \leq y \]
\[ \text{MOP}(B) \leq x \quad \text{(true solution)} \]

Decent solution \( \leq \text{MOP}(B) \leq \text{true solution} \)
easier to obtain if only monotone

if framework is distributive \( \Rightarrow \) solution = MOP
Iterative solution for general frameworks:

Input: \((L, F, ^) f \) in F for all nodes element of G
Output: \(\text{IN}[B] \) in L for all B element of G

for each node B do
  \(\text{out}[B] = f_B[\pi]\)
done
while (changes to any \(\text{out}[B]\)) do
  for each B, in DFS order do
    \(\text{in}[B] = ^{(\text{all pred P of B}) \text{out}[P]}\)
    \(\text{out}[B] = f_B[\text{in}[B]]\)
done
endwhile
**properties of iterative alg:**

Let $P$ is path from $B_0$ to $B_k = B$

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

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

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

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

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

---

![Graph Diagram]

- Looks like $B_0 \rightarrow B_1 \rightarrow B_4 \rightarrow B_5$ and $B_0 \rightarrow B_3 \rightarrow B_2 \rightarrow B_5$ are real, so not accurate
convergence of iterative alg:

if we use only acyclic paths 2 + depth of G iterations

if we need cyclic paths:

\[
\begin{align*}
x &= 2 \\
x &= x + 1
\end{align*}
\]

new value \leq old value

suppose n nodes and v variables

OUT[B] can go from undef -> c -> nonconst

\[\Rightarrow\] max number of iterations until no change 2*n*v