Dataflow Frameworks, conclusion
Announcements

- Welcome back!
- I’ve moved Quiz 1 to Thursday
- Office hour at 4:30pm today
- Recordings:
- Homework due January 31st
  - Questions?
  - Will push back deadline if necessary
Outline of Today’s Class

- Dataflow framework
  - Lattices
  - Transfer functions
  - Worklist algorithm

- MOP solution vs. MFP solution
Lattice Theory

- **Partial ordering** (denoted by \( \leq \) or \( \subseteq \))
  - Relation between pairs of elements
  - Reflexive: \( a \leq a \)
  - Anti-symmetric: \( a \leq b \) and \( b \leq a \) \( \implies a = b \)
  - Transitive: \( a \leq b \) and \( b \leq c \) \( \implies a \leq c \)

- **Partially ordered set** (poset) (set \( S, \leq \))
  - 0 element: \( 0 \leq a \), for every \( a \) in \( S \)
  - 1 element: \( a \leq 1 \), for every \( a \) in \( S \)

We don’t necessarily need 0 or 1 element
D = \{a, b, c\}
The poset is $2^D$, $\leq$ is set inclusion
Lattice Theory

- Greatest lower bound (glb)
  \( l_1, l_2 \) in poset \( S \), \( a \) in poset \( S \) is the \( \text{glb}(l_1, l_2) \) iff
  1) \( a \leq l_1 \) and \( a \leq l_2 \)
  2) for any \( b \) in \( S \), \( b \leq l_1, b \leq l_2 \) implies \( b \leq a \)

If glb exists, it is unique. Why? Called *meet* (denoted by \( \wedge \) or \( \cap \)) of \( l_1 \) and \( l_2 \).

- Least upper bound (lub)
  \( l_1, l_2 \) in poset \( S \), \( c \) in poset \( S \) is the \( \text{lub}(l_1, l_2) \) iff
  1) \( c \geq l_1 \) and \( c \geq l_2 \)
  2) for any \( d \) in \( S \), \( d \geq l_1, d \geq l_2 \) implies \( d \geq c \)

If lub exists, it is unique. Called *join* (denoted by \( V \) or \( \cup \)) of \( l_1 \) and \( l_2 \).
Definition of a Lattice \( (L, \Lambda, V) \)

- A lattice \( L \) is a poset under \( \leq \), such that every pair of elements has a \textit{glb} (meet) and \textit{lub} (join).

- A lattice need not contain a 0 or 1 element.
- A finite lattice must contain 0 and 1 elements.
- Not every poset is a lattice.
- If there is element \( a \) such that \( a \leq x \) for every \( x \) in \( L \), then \( a \) is the 0 element of \( L \).
- If there is \( a \) such that \( x \leq a \) for every \( x \) in \( L \), then \( a \) is the 1 element of \( L \).
A Poset but Not a Lattice

There is no \text{lub}(e_3, e_4) in this poset so it is not a lattice.

Suppose we add the \text{lub}(e_3, e_4), is it a lattice?
Is This Poset a Lattice

\[ D = \{a, b, c\} \]

The poset is \(2^D\), \(\leq\) is set inclusion
Examples of Lattices

- $H = (2^D, \cap, U)$ where $D$ is a finite set
  - $\text{glb}(s_1,s_2)$ denoted $s_1 \Lambda s_2$, is set intersection $s_1 \cap s_2$
  - $\text{lub}(s_1,s_2)$ denoted $s_1 \vee s_2$, is set union $s_1 \cup s_2$

- $J = (N_1, \gcd, \lcm)$
  - Partial order is integer divide on $N_1$
  - $\text{glb}(n_1,n_2)$ denoted $n_1 \Lambda n_2$ is $\gcd(n_1,n_2)$
  - $\text{lub}(n_1,n_2)$ denoted $n_1 \vee n_2$ is $\lcm(n_1,n_2)$

($N_1$ denotes natural numbers starting at 1)
A poset $C$ where for every pair of elements $c_1, c_2$ in $C$, either $c_1 \leq c_2$ or $c_2 \leq c_1$.

- E.g., $\{\} \leq \{a\} \leq \{a,b\} \leq \{a,b,c\}$
- E.g., from the lattice $J$ as shown here,
  
  $1 \leq 2 \leq 6 \leq 30$
  
  $1 \leq 3 \leq 15 \leq 30$

A lattice s.t. every ascending chain is finite, is said to satisfy the *Ascending Chain Condition*
Lattices in Dataflow Analysis

- Lattices define the property space
- Lattices lead to certain properties of the standard solution procedure for dataflow analysis (the worklist algorithm)
Dataflow Lattices: Reach

D = all definitions: {(x,1),(x,4),(a,3)}  {(x,1),(x,4),(a,3)}  
Poset is $2^D$, $\leq$ is the subset relation $\subseteq$

1. $x=a*b$

2. if $y<=a*b$

3. $a=a+1$

4. $x=a*b$

5. goto 3
Dataflow Lattices: \textit{Avail}

D = all expressions: \{a*b, a+1, y*z\}

Poset is $2^D$, $\leq$ is the superset relation $\supseteq$

1. $x := a*b$

2. if $y*z \leq a*b$

3. $a := a+1$

4. $x := a*b$

5. goto 2

CSCI 4450/6450, A Milanova
Dataflow Framework

Equations:

\[ \text{in}(j) = V \text{out}(i) \quad \text{out}(j) = f_j(\text{in}(j)) \]

\( i \) in pred\((j)\)

where:

- \( \text{in}(j), \text{out}(j) \) are elements of a property space
- \( f_j \) is the transfer function associated with node \( j \)
- \( V \) is the merge operator
Dataflow Framework (cont.)

- Analysis property space must be:
  1. A lattice \( L, \leq \)
  2. \( L \) satisfies the *Ascending Chain Condition*
     Requires that all ascending chains are finite
Dataflow Framework (cont.)

- **Merge operator** $V$ must be the join of $L$

- In dataflow, $L$ is often the lattice of the subsets over a finite set of dataflow facts $D$
  - Choose universal set $D$ (e.g., all definitions)
  - Choose ordering operation $\leq$
    - In many problems merge operator is the union, thus, $\leq$ is *subset*
    - In *must* problems merge is intersection, thus, $\leq$ is *superset*
Example: *Reach* Lattice

- Property space is the lattice of the subsets where
  - $D$ is the set of all definitions in the program
  - $\leq$ is the *subset* operation
    - Join is set union $U$, as needed for *Reach*, which is a *may* problem

- Lattice has a $0$ being $\emptyset$, and a $1$ being $D$
- Lattice satisfies the *Ascending Chain Condition*
Reach Lattice

D = all definitions: \{(x,1),(x,4),(a,3)\} \{(x,1),(x,4),(a,3)\}
Poset is \(2^D\), \(\leq\) is the subset relation \(\subseteq\)

1. \(x = a \cdot b\)
2. if \(y \leq a \cdot b\)
3. \(a = a + 1\)
4. \(x = a \cdot b\)
5. goto 3

\{(x,1),(x,4)\} \{(x,4),(a,3)\} \{(x,1),(a,3)\}
\{(x,1)\} \{(x,4)\} \{(a,3)\}
\{\}\n
Example: \textit{Avail} Lattice

- Property space is the lattice of the subsets where
  - $\mathbb{D}$ is the set of all expressions in the program
  - $\leq$ is superset
    - join of the lattice is set intersection, as needed for \textit{Avail}, which is a \textit{must} problem

- Lattice has a \textbf{0} being $\mathbb{D}$, and a \textbf{1} being $\emptyset$
- Lattice satisfies \textit{Ascending Chain Condition}
Dataflow Lattices: \textit{Avail}

\( D = \text{all expressions: \{a*b, a+1, y*z\}} \)

Poset is \(2^D, \leq\) is the superset relation \(\supseteq\)

1. \( x := a*b \)

2. if \( y*z \leq a*b \)

3. \( a := a + 1 \)

4. \( x := a*b \)

5. goto 2

\[ \begin{array}{c}
\{a*b\} & \{a+1\} & \{y*z\} \\
\{a*b, y*z\} & \{a*b, a+1\} & \{a+1, y*z\} \\
\{a*b, a+1, y*z\} & \{\} & 1 \\
0 & & \\
\end{array} \]
(Monotone) Dataflow Framework

- A problem fits into the dataflow framework if
  - its property space is a lattice $\mathbf{L}$, $\leq$ that satisfies the Ascending Chain Condition
  - its merge operator $V$ is the join of $\mathbf{L}$
  and
  - its transfer function space $F : \mathbf{L} \rightarrow \mathbf{L}$ is monotone

- We can make use of a generic solution procedure, known as the worklist algorithm.
Outline of Today’s Class

- Dataflow frameworks
  - Lattices
  - Transfer functions
  - Worklist algorithm

- MOP solution vs. MFP solution
Transfer Functions

The transfer functions: $f: L \rightarrow L$. Formally, function space $F$ is such that

1. $F$ contains all $f_j$
2. $F$ contains the identity function $\text{id}(x) = x$
3. $F$ is closed under composition
4. Each $f$ is monotone
Monotonicity Property

- **F: L → L** is monotone if and only if:
  1. \(a, b \in L, f \in F \) then \(a \leq b \iff f(a) \leq f(b)\)
  or (equivalently):
  2. \(x, y \in L, f \in F \) then \(f(x) \lor f(y) \leq f(x \lor y)\)

- Theorem: Definitions (1) and (2) are equivalent.
  - Show that (1) implies (2)
  - Show that (2) implies (1)
Monotonicity Property

- Show that (1) implies (2)
Distributivity Property

- $F : L \rightarrow L$ is **distributive** if and only if $x, y \in L$, $f \in F$ then $f(x \lor y) = f(x) \lor f(y)$

- Every distributive function is also monotone but not the other way around

- Distributivity is a very nice property!
Monotonicity and Distributivity

Is classical \textit{Reach} distributive?
- Yes

To show distributivity:
For each \( j \):
\[
( ( X \cup Y ) \cap \text{pres}(j) ) \cup \text{gen}(j) =
( (X \cap \text{pres}(j)) \cup \text{gen}(j) ) \cup ( (Y \cap \text{pres}(j)) \cup \text{gen}(j) )
\]

\[
( ( X \cup Y ) \cap \text{pres}(j) ) \cup \text{gen}(j) =
( ( X \cap \text{pres}(j) ) \cup ( Y \cap \text{pres}(j) ) ) \cup \text{gen}(j) =
( (X \cap \text{pres}(j)) \cup \text{gen}(j) ) \cup ( (Y \cap \text{pres}(j)) \cup \text{gen}(j) )
\]
Monotone Dataflow Framework

A problem fits into the dataflow framework if
- its property space is a lattice $\mathbf{L}, \leq$ that satisfies the Ascending Chain Condition
- its merge operator $V$ is the join of $\mathbf{L}$ and
- its transfer function space $F: \mathbf{L} \rightarrow \mathbf{L}$ is monotone

Thus, we can make use of a generic solution procedure, known as the worklist algorithm.
/* Initialize to initial values; 1 is entry node of CFG */
in(1) = InitialValue; out(1) = f_1(in(1))
for m = 2 to n do in(m) = 0; out(m) = f_m(0)
W = \{2,\ldots,n\} /* put every node but 1 on the worklist */

while W ≠ Ø do {
    remove j from W
    in(j) = V \{ out(i) | i is predecessor of j \}
    out(j) = f_j(in(j))
    if out(j) changed then
        W = W U \{ k | k is successor of j \}
}
Worklist Algorithm on Reach

D = all definitions: {(x,1),(x,4),(a,3)}
Poset is $2^D$, $\leq$ is the subset relation $\subseteq$

1. $x = a \cdot b$

2. if $y \leq a \cdot b$

3. $a = a + 1$

4. $x = a \cdot b$

5. goto 3
Termination Argument

Why does the algorithm terminate?

Sketch of argument:
At each “phase”, at least one $\text{out}(j)$ changes. Monotonicity of $f_j$ entails that change is up the chain: $\text{out}’(j) \geq \text{out}(j)$. Since $\text{out}(j)$ in $L$, and $L$ satisfies the Ascending Chain Condition, $\text{out}(j)$ changes at most $O(h)$ times where $h$ is the height of the lattice $L$. 
Correctness Argument

Theorem: The worklist algorithm computes a solution that satisfies the dataflow equations.

Why?

Sketch of argument:
- Argue that for each path, $\text{out}(i) \leq \text{in}(j)$ where $i$ is a predecessor of $j$. Thus, $V_{\text{out}}(i) \leq \text{in}(j)$
- Then argue that $\text{in}(j) \leq V_{\text{out}}(i)$
Theorem: The algorithm computes the least solution of the dataflow equations.

Historically though, this solution is called the maximal fixpoint solution (MFP).

I.e., For every node $j$, the worklist algorithm computes a solution $\text{MFP}(j) = \{\text{in}(j), \text{out}(j)\}$, such that every other solution $\{\text{in}'(j), \text{out}'(j)\}$ of the dataflow equations $\text{in}(j) \leq \text{in}'(j)$ and $\text{out}(j) \leq \text{out}'(j)$.
Example

1. \( z := x + y \)

2. if (\( z > 500 \))
   - \( \text{in}_{\text{Avail}}(1) = \emptyset \)
   - \( \text{out}_{\text{Avail}}(1) = (\text{in}_{\text{Avail}}(1) - E_z) \cup \{x+y\} \)
   - \( \text{in}_{\text{Avail}}(2) = \text{out}_{\text{Avail}}(1) \lor \text{out}_{\text{Avail}}(3) \)
   - \( \text{out}_{\text{Avail}}(2) = \text{in}_{\text{Avail}}(2) \)
   - \( \text{in}_{\text{Avail}}(3) = \text{out}_{\text{Avail}}(2) \)
   - \( \text{out}_{\text{Avail}}(3) = \text{in}_{\text{Avail}}(3) \)

3. skip

Equivalent to: \( \text{in}_{\text{Avail}}(2) = \{x+y\} \lor \text{in}_{\text{Avail}}(2) \)
and recall that \( \lor \) is \( \cap \) (i.e., set intersection).

Solution 1
- \( \emptyset \)
- \( \{x+y\} \)
- \( \{x+y\} \)
- \( \emptyset \)

Solution 2
- \( \emptyset \)
- \( \{x+y\} \)
- \( \emptyset \)
Outline of Today’s Class

- Dataflow frameworks, conclusion
  - Lattices (last time)
  - Transfer functions
  - Worklist algorithm

- MOP solution vs. MFP solution
Meet Over All Paths (MOP)

- Desired dataflow information at $n$ is obtained by traversing ALL PATHS from 1 (entry node) to $n$.

- For every path $p=(1, n_2, n_3, ..., n_k)$ we compute $f_{n_k}(...f_{n_2}(f_1(\text{Initial\ Value})))$

- The MOP at entry of $n$ is $\forall f_{n_k}(...f_{n_2}(f_1(\text{Initial\ Value})))$ over all paths $p$ from 1 to $n$
MOP vs. MFP

- MOP is an abstraction of the best solution computable with dataflow analysis
  - It is a common assumption in dataflow analysis that *all program paths are executable*
  - (Abstract interpretation and axiomatic semantics are more precise and rule out some infeasible paths)
- Recall that the MFP is the solution computed by the worklist algorithm
MOP vs. MFP

- For *distributive* problems $\text{MFP} = \text{MOP}$!

- Unfortunately, for *monotone* problems this is not true. But we still have a *safe* solution: it is a theorem that for monotone problems, $\text{MFP} \geq \text{MOP}$
Safety of a Dataflow Solution

- A safe (also, correct or sound) solution $X$ overestimates the “best” possible dataflow solution, i.e., $X \geq \text{MOP}$

- Historically, an acceptable solution $X$ is one that is better than what we can do with the MFP, i.e., $X \leq \text{MFP}$
Safe Solutions: Reach

\[ U = \text{all definitions:}\{(x,1),(x,4),(a,3)\} \quad \{(x,1),(x,4),(a,3)\} \]

Poset is \( 2^U \), \( \leq \) is the subset relation \( \subseteq \)

1. \( x=a*b \)

2. if \( y \leq a*b \)

3. \( a=a+1 \)

4. \( x=a*b \)

5. goto 3

CSCI 4450/6450, A Milanova
Safe Solutions: Avail

U = all expressions: \{a*b, a+1, y*z\}

Poset is $2^U$, $\leq$ is the superset relation $\supseteq$

1. $x := a*b$

2. if $y*z \leq a*b$

3. $a := a+1$

4. $x := a*b$

5. goto 2
Precision of a Dataflow Solution

- **Precise** solution is one that is “close” to MOP
  - A precise solution contains few spurious dataflow facts (spurious facts is what we call *noise*)
  - Unfortunately, for most problems even the MOP (an approximation itself!) is undecidable

- MOP \( \leq X \leq Y \): X is more precise than Y
  - Usually, we can compare two solutions X and Y
  - But, for most problems, we have no way of knowing the “ground truth”
Next class: real analyses

- Next time: non-distributive analyses
  - Constant propagation
  - Pointer analysis