Advanced Compilers
CMPSCI 710
Spring 2003
More data flow analysis

Emery Berger
University of Massachusetts, Amherst

More Data Flow Analysis

- Last time
  - Program points
  - Lattices
  - Max fixed point
  - Reaching definitions
- Today
  - Iterative Worklist Algorithm
    - actual algorithm, examples

From Last Time:
Informal Description
- Define lattice to represent facts
- Attach meaning to lattice values
- Associate transfer function to each node
- Initialize values at each program point
- Iterate through program until fixed point

Iterative Worklist Algorithm
for v 2 V
IN(v) = ∅
OUT(v) = Gen(v)
worklist Ą V
while (worklist ≠ ∅)
v Ą remove(worklist)
oldout(v) = OUT(v)
IN(v) = [p 2 PRED(v) \ OUT(p)]
OUT(v) = GEN(v) \ (IN(v) - KILL(v))
if (oldout(v) ≠ OUT(v))
  worklist Ą worklist \ SUCC(v)

Bounding Expected Runtime
- Order matters: visit nodes in reverse postorder
- Nodes visited roughly after its predecessors
- Intuition: accumulate as much info as possible before processing each node
Reverse Postorder

```plaintext
visited(n):
visited(n) = true
for s \in \text{SUCC}(n)
if not visited(s)
visited(s) = true
postorder(n) = count
count = count + 1
for each node n
visited(n) = false
```

Reverse Postorder: Examples

1. visited(1) = postorder(1) =
2. visited(2) = postorder(2) =
3. visited(3) = postorder(3) =
4. visited(4) = postorder(4) =

Loop Interconnectiveness

- Defined as \( d(G) = \max \{ \text{number of back edges on any acyclic path in graph } G \} \)
- up to \(|N|\)
- but usually 3 and often 1
- \( d(G) = 1 \) for reducible flow graphs

Loop Interconnectiveness: Examples

1. \( d(G) = 1 \)
2. \( d(G) = 2 \)
3. \( d(G) = 3 \)

Iterative Worklist Algorithm, Modified

```plaintext
for v \in V
IN(v) = \emptyset
OUT(v) = Gen(v)
worklist \rightarrow rPostorder(V), changed \rightarrow true
while (changed)
changed \rightarrow false
for v \in worklist
oldout(v) = OUT(v)
IN(v) = \{ u \in PRED(v) | OUT(u) \}
OUT(v) = GEN(v) \cup \{ u \in IN(v) - KILL(v) \}
if (oldout(v) \neq OUT(v))
changed \rightarrow true
```
**Reaching Definitions Example**

```
<table>
<thead>
<tr>
<th>Entry</th>
<th>visited(1)</th>
<th>postorder(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>visited(2)</td>
<td>postorder(2)</td>
</tr>
<tr>
<td></td>
<td>visited(3)</td>
<td>postorder(3)</td>
</tr>
<tr>
<td></td>
<td>visited(4)</td>
<td>postorder(4)</td>
</tr>
<tr>
<td></td>
<td>visited(5)</td>
<td>postorder(5)</td>
</tr>
<tr>
<td></td>
<td>visited(6)</td>
<td>postorder(6)</td>
</tr>
<tr>
<td></td>
<td>visited(7)</td>
<td>postorder(7)</td>
</tr>
</tbody>
</table>
```

```
0: a = a + 1
0: a = a + 1
0: a = a + 1
0: a = a + 1
0: a = a + 1
0: a = a + 1
```

**For Reaching Definitions**

- For reaching `defs`, `u = {` (\`d\`)}
  - `Gen(d; v = exp) = \{d\}`
    - "on exit from block \`d\`, generate new definition"
  - `Kill(d; v = exp) = \{d\}`
    - "on exit from block \`d\`, definitions of \`v\` are killed"
- Computing \`In(S)` and \`Out(S)`
  - \`In(S) = \{P in PRED(S) \ Out(P)\}`
  - \`Out(S) = Gen(v) \ \{In(v) \ Kill(v)\}`

**Iterative Worklist Algorithm: Revised Analysis**

- Stabilizes in at most \(d(G) + 2\) iterations
  - \(d(G) + 1\) iterations to propagate data
  - 1 iteration to detect stability
- As noted, \(d(G) \approx 3\), often 1
- Each pass computes:
  - \(O(E)\) meets (sets of size \(|defs|\))
  - \(O(N)\) other operations
- Effectively \(O(N)\) complexity
- Note: for backwards analysis, use postorder

**Other Data Flow Problems**

- Definitely uninitialized variables
  - `Gen(S) =` (\`if\ (a == 2)\`) 
  - `Kills(S) = c = 1`
  - `Out(E, entry) = else b = 2`
  - `u =`
- Possibly uninitialized variables
  - `Gen(S) =`
  - `Kills(S) =`
  - `Out(E, entry) = u =`

**Next Time – Even More Data Flow!**

- Live variable analysis
- Backwards problem
- Constant propagation
- Supplementary paper available:
  - Wegman & Zadeck, TOPLAS 1991