CS 412
Introduction to Compilers

Andrew Myers
Cornell University

Lecture 29: Data-flow, control-flow analysis
11 Apr 01

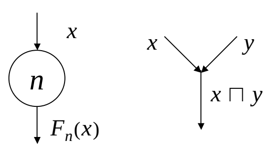
Administration

- HW4 due Friday the 13^{th}
- Prelim 2 next Tuesday

Last time

Dataflow analysis framework:

1. Lattice of dataflow information values L with order $\sqsubseteq$, top $\top$
2. Monotonic flow functions $F_n : L \rightarrow L$
3. Meet (GLB) operator $\sqcap$ on L



Solution quality

- MOP is best possible solution:

  $out[n] = \bigsqcap_{all\ paths\ (\ldots,\ p_2,\ p_1,\ n)} F_n(F_{p_1}(F_{p_2}(\ldots)))$
- Does iterative analysis

  $x_i = F_i(\bigsqcap_{j \in pred[i]} x_j)$

  produce the MOP solution?
- Yes, if flow functions *distribute* over the meet operator:

  $\bigsqcap_i F_n(x_i) = F_n\ (\bigsqcap_i x_i)$
- Not all analyses give MOP solution!

Other analyses

- Live variables, reaching definitions

  $F_n(l) = gen[n] \cup (l - kill[n]), \quad \sqcap = \cup$
- Available expressions

  $F_n(l) = gen[n] \cup (l - kill[n]), \quad \sqcap = \cap$

- Do they terminate?
- Compute MOP solutions?

Summary

- Analyses for standard optimizations fit into dataflow analysis framework
- Iterative analysis finds solution if flow function monotonic in $\sqsubseteq$, combining function $\sqcap$ is GLB of lower semilattice
- Solution is MOP if distribution condition

  $\bigsqcap_i F(x_i) = F(\bigsqcap_i x_i)$ holds

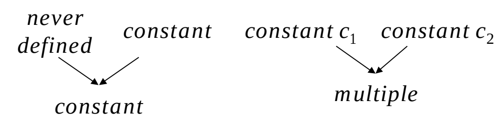
"classic" constant propagation

- Idea: propagate and fold integer constants in one pass

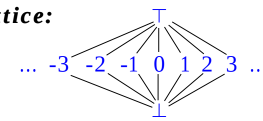
```
x = 1;        x = 1;
y = 5+x;  =>  y = 6;
z = y*y;      z = 36;
```

- Information about a single variable:
  - i. Variable never defined
  - ii. Variable has single constant value
  - iii. Variable has multiple values

One-variable Const. Prop.



Full lattice:



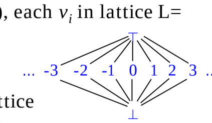
Rest of definition

- Flow function for $x = x$ OP c_1:

  $F_n(\top) = \top$
  $F_n(\bot) = \bot$
  $F_n(c_2) = c_2 \text{ OP } c_1$
- Flow function is monotonic, distributive: iterative solution works, gives MOP
- What about multiple variables $x_1 \ldots x_n$? Want tuple $(v_1, \ldots v_n)$,

Multiple vars

- Dataflow value is a tuple $(v_1, \ldots v_n)$, each v_i in lattice L=



- Set of tuples $(v_1, \ldots, v_n)$ is also a lattice under component-wise ordering:

  $$(v_1, \ldots, v_n) \sqsubseteq (v'_1, \ldots, v'_n) \Leftrightarrow \forall_i\, v_i \sqsubseteq v'_i$$
  $$(v_1, \ldots, v_n) \sqcap (v'_1, \ldots, v'_n) = (v_1 \sqcap v'_1, \ldots, v_n \sqcap v_n)$$
- For any two lattices L_1, L_2, have *product lattice* $L_1 \times L_2$

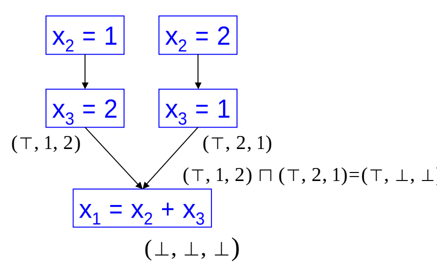
  $$(v_1, v_2) \sqsubseteq (v'_1, v'_2) \Leftrightarrow v_1 \sqsubseteq v'_1 \ \& \ v_2 \sqsubseteq v'_2$$
- Tuple dataflow values are in $L \times \ldots \times L = L^n$

Flow functions

- Consider $x_1 = x_2 \text{ OP } x_3$

  $F(x_1, \top, x_3) = (\top, \top, x_3)$
  $F(x_1, x_2, \top) = (\top, x_2, \top)$
  $F(x_1, \bot, x_3) = (\bot, \bot, x_3)$
  $F(x_1, x_2, \bot) = (\bot, x_2, \bot)$
  $F(x_1, c_2, c_3) = (c_2 \text{ OP } c_3, c_2, c_3)$
- Monotonic? Distributes over $\sqcap$?

Not MOP!



$$F((\top, 1, 2) \sqcap (\top, 2, 1)) \neq F(\top, 1, 2) \sqcap F(\top, 2, 1)$$

Loops

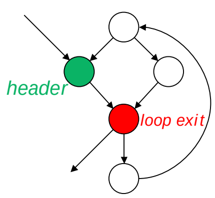
- Most execution time in most programs is spent in loops: 90/10 is typical
- Most important targets of optimization: loops
- Loop optimizations:
  - loop-invariant code motion
  - loop unrolling
  - loop peeling
  - strength reduction of expressions containing induction variables
  - removal of bounds checks
  - loop tiling
- When to apply loop optimizations?

High-level optimization?

- Loops may be hard to recognize in IR or quadruple form -- should we apply loop optimizations to source code or high-level IR?
  - Many kinds of loops: while, do/while, continue
  - loop optimizations benefit from other IR-level optimizations and vice-versa -- want to be able to interleave
- Problem: identifying loops in flowgraph

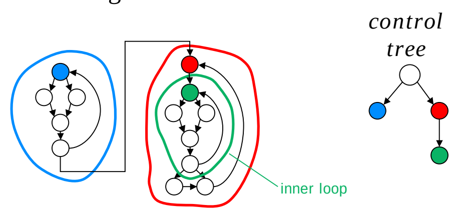
Definition of a loop

- A *loop* is a set of nodes in the control flow graph, with one distinguished node called the *header* (entry point)
- Every node is reachable from header, header reachable from every node: *strongly-connected component*
- No entering edges from outside except to header
- nodes with outgoing edges: *loop exit nodes*



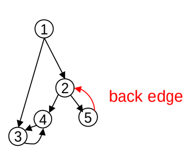
Nested loops

- Control-flow graph may contain many loops, and loops may contain each other
- *Control-flow analysis* : identify the loops and nesting structure:



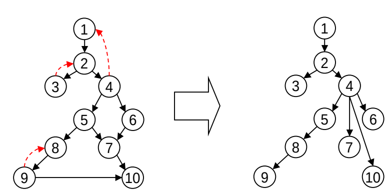
Dominators

- CFA based on idea of *dominators*
- Node A *dominates* node B if the only way to reach B from start node is through A
- Edge in flowgraph is a *back edge* if destination dominates source
- A loop contains at least one back edge



Dominator tree

- Domination is transitive; if A dominates B and B dominates C, then A dominates C
- Domination is anti-symmetric
- Every flowgraph has *dominator tree* (Hasse diagram of domination relation)



Dominator dataflow analysis

- Forward analysis; out[n] is set of nodes dominating n
- "A node **B** is dominated by another node **A** if **A** dominates *all* of the predecessors of **B**"

$$in[n] = \bigcap_{n' \in pred[n]} out[n']$$

- "Every node dominates itself"

$$out[n] = in[n] \cup \{n\}$$

- Formally: L = sets of nodes ordered by $\subseteq$, flow functions $F_n(x) = x \cup \{n\}$, $\sqcap = \cap$, $\top = \{\text{all } n\}$
  $\Rightarrow$ Standard iterative analysis gives best soln

Completing control-flow analysis

- Dominator analysis gives all back edges
- Each back edge $n \rightarrow h$ has an associated *natural loop* with h as its header: all nodes reachable from h that reach n without going through h
- For each back edge, find natural loop
- Nest loops based on subset relationship between natural loops
- Exception: natural loops may share same header; merge them into larger loop.
- Control tree built using nesting relationship