CS 4120
Introduction to Compilers

Ross Tate
Cornell University

Lecture 31: Control-flow analysis

Loops

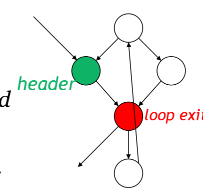
- Most execution time in most programs is spent in loops: 90/10 is typical
- Most important targets of optimization: loops
- Loop optimizations:
  - loop-invariant code motion
  - loop unrolling
  - loop peeling
  - loop-induction-variable strength reduction
  - removal of bounds checks
  - loop tiling
- 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 CFG

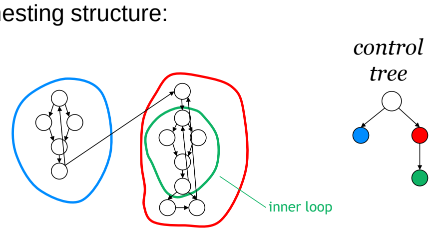
Definition of a loop

- A *loop* is a set of nodes in the CFG
- We will assume each loop has a unique entry point, called the *header*
- 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*



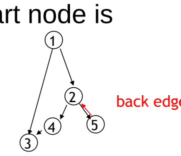
Nested loops

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



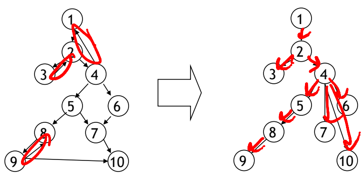
Dominators

- 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 CFG 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 CFG has *dominator tree*



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**

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

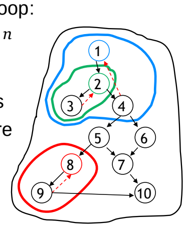
- Every node dominates itself

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

- Formally
  - preorder = sets of nodes ordered by $\supseteq$
    - $\bot$ = {all n}
  - flow functions $F_n(x) = x \cup \{n\}$, ~~$\top$ = {all n}~~
  - Standard iterative analysis gives best solution

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
- For each back edge, find natural loop:
  - all nodes reachable from h that reach n without going through h
- 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



Inserting ϕ Nodes