Program Representation

Last Time

- Live variable analysis
- Constant propagation
  leads us to SSA and how to connect uses and def

Today

- Finish constants
- Goal: understand control flow more deeply to build SSA
- Dominator relationships
- DOM, IDOM, DOM^{-1}, DOM!, post-dominators
- Control Dependence

Dominator Relationships

Dominators

x dominates y, x DOM y, in a *CFG* if $\forall$ paths from *Entry* to y include x.

DOM(v) = the set of all **vertices that dominate** v.

- All vertices dominate themselves, $v \in$ DOM(v).
- *Entry* dominates every vertex in the graph, $\forall v$ *Entry* $\in$ DOM(v).
- DOM is reflexive, antisymmetric, and transitive.

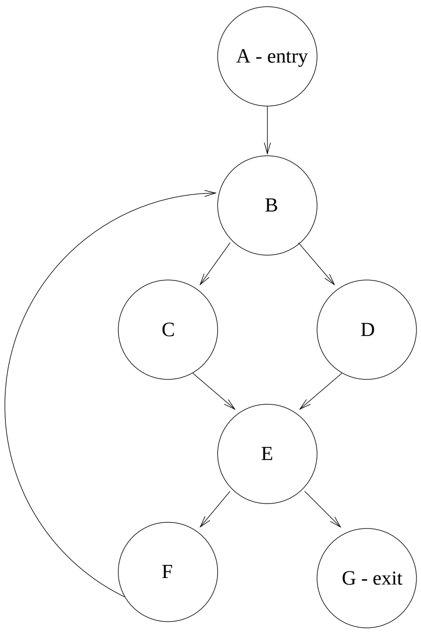
Strict Dominators

- DOM!(v) = DOM(v) - $\{v\}$, strictly dominates v
- antisymmetric and transitive

Immediate Dominator

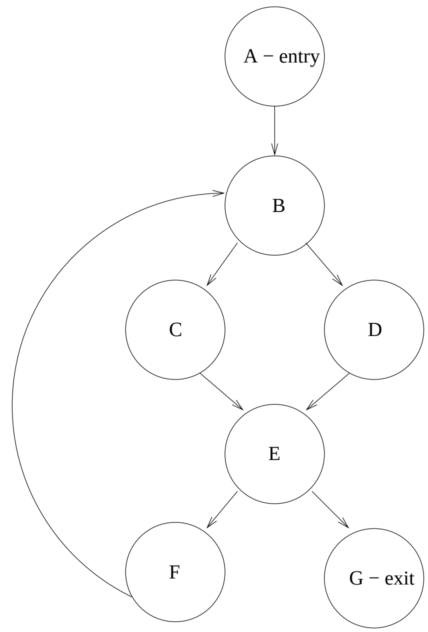
- IDOM(v) = the closest, strict dominator of v.
  d IDOM v if
  d DOM! v and ($\forall w \in w$ DOM! v) $[w$ DOM $d]$
- antisymmetric

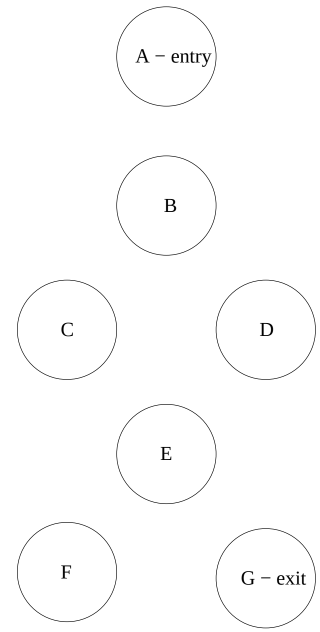
Dominator Example



v	DOM(v)	DOM! (Strict)	IDOM(v)
A			
B			
C			
D			
E			
F			
G			

Dominator Tree





Dominator Relationships

Theorem: IDOM(v) is unique, *i.e.*, a singleton.

Proof: by contradiction. Suppose c IDOM v and d IDOM v. By definition, $c \neq v$ and $d \neq v$, so c DOM! v and d DOM! v. By definition of IDOM,

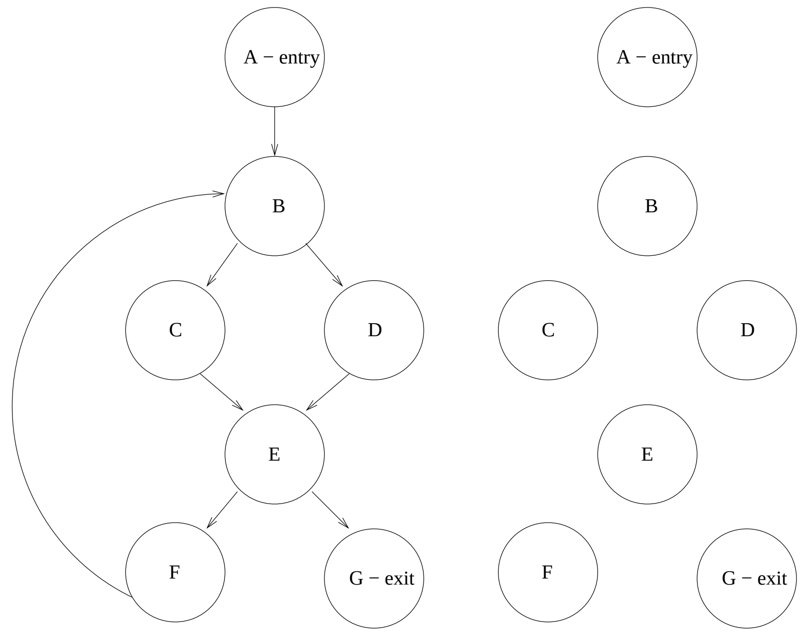
$$d \text{ DOM! } v \text{ and } (\forall w \in w \text{ DOM! } v)\ [w \text{ DOM } d].$$

Thus, c DOM d and d DOM c, but DOM is antisymmetric, a contradiction if $c \neq d$. c and d must therefore be the same vertex.

Inverse Dominators

- DOM$^{-1}(v)$ = the set of all **vertices dominated by** v.
- reflexive, antisymmetric, and transitive

Inverse Dominator Example



v	DOM(v)	DOM$^{-1}(v)$
A	$\{A\}$	
B	$\{A,B\}$	
C	$\{A,B,C\}$	
D	$\{A,B,D\}$	
E	$\{A,B,E\}$	
F	$\{A,B,E,F\}$	
G	$\{A,B,E,G\}$	

Finding Dominators

DOM(v) = the set of all **vertices that dominate** v.

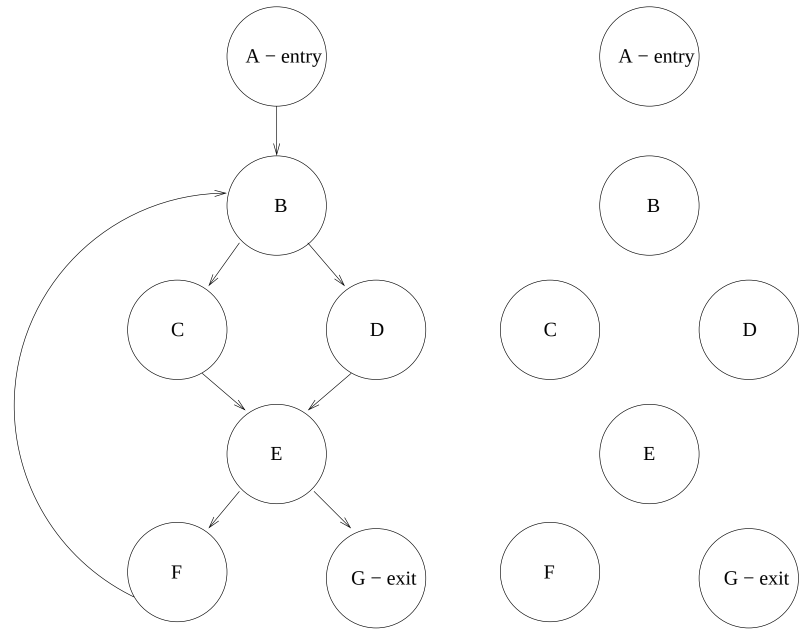
$$\mathrm{DOM}(v) = \{v\} \cup \bigcap_{p \in \mathrm{PRED}(v)} \mathrm{DOM}(p)$$

Algorithm:

DOM(*Entry*) = { *Entry* }
for $v \in V - \{$ *Entry* $\}$ **do** DOM(v) = V
repeat
  changed = **false**
  for $n \in V - \{$ *Entry* $\}$ **do**
    olddom = DOM(n)
    $\mathrm{DOM}(n) = \{n\} \cup \bigcap_{p \in \mathrm{PRED}(v)} \mathrm{DOM}(p)$
    if DOM(n) $\neq$ *olddom* **then** *changed* = **true**
  endfor
until *changed* = **false**

Complexity: O(N^2)

Dominator Algorithm Example



DOM(v) iteration:	0	1	2
A	$\{A\}$		
B	$\{A,B,C,D,E,F,G\}$		
C	$\{A,B,C,D,E,F,G\}$		
D	$\{A,B,C,D,E,F,G\}$		
E	$\{A,B,C,D,E,F,G\}$		
F	$\{A,B,C,D,E,F,G\}$		
G	$\{A,B,C,D,E,F,G\}$		

Post-Dominators

CFG = $\langle V, E,$ *Entry, Exit* $\rangle$

$(\forall v \in V)[v \xrightarrow{*}$ *Exit*]

Exit is reachable from all other nodes

PDOM(v): all nodes that post-dominate v

p post-dominates v, if every path from v to *Exit* includes p

- p PDOM v implies $v \xrightarrow{*}$ *Exit* can be split into $v \xrightarrow{*} p$ and $p \xrightarrow{*}$ *Exit*
- reflexive, antisymmetric, and transitive
- PDOM on *CFG* is the same as DOM on the reverse *CFG*

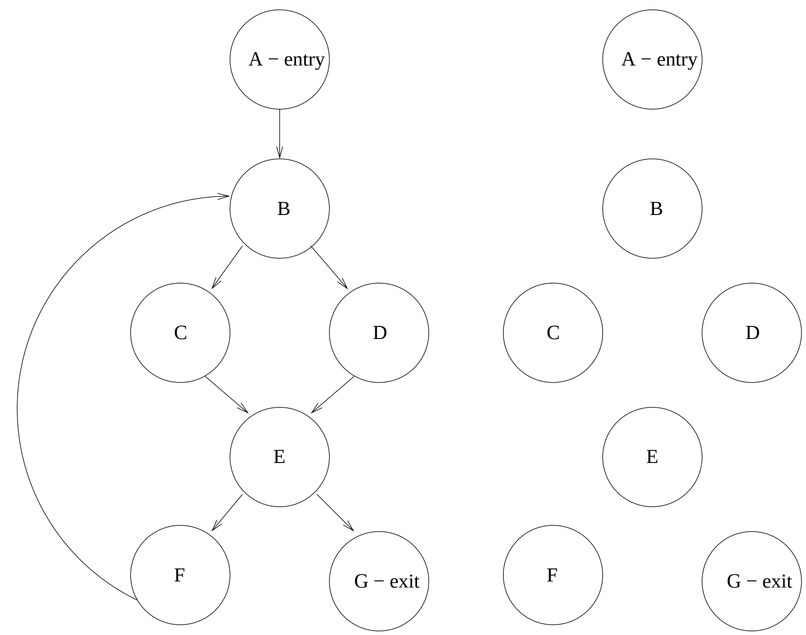
strict post-dominators

- p PDOM! $v \Longleftrightarrow p$ PDOM v & $p \neq v$

post-dominance frontier

- $v \in$ PDF(p) if p PDOM SUCC(v) but p is not p PDOM! v

Post-Dominator Example



Control Dependence Graph - G_{cd}

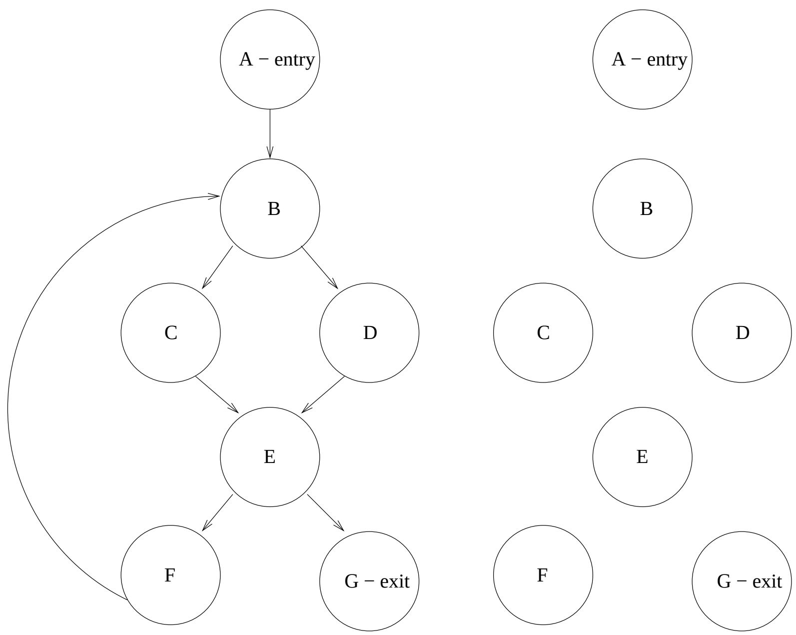
y is control dependent on x, x and y in *CFG*, if:

- $\exists\ x \stackrel{*}{\rightarrow} y$, y post-dominates every vertex p in $x \stackrel{*}{\rightarrow} y$, $p \neq x$, and
- y does not strictly post-dominate x.
- $(x,y)_l$ has label l, the first edge on $x \stackrel{*}{\rightarrow} y$.

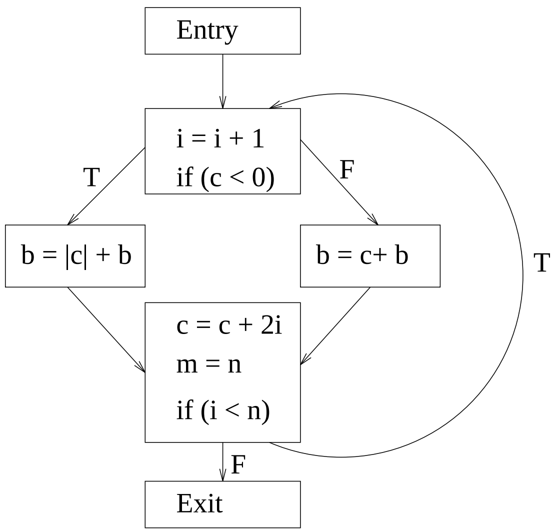
CDPRED(y) = $\{x \mid y$ is control dependent on $x\}$
CDSUCC(x) = $\{y \mid y$ is control dependent on $x\}$

Note: add edge (*entry, exit*) in *CFG*

Control Dependence Example

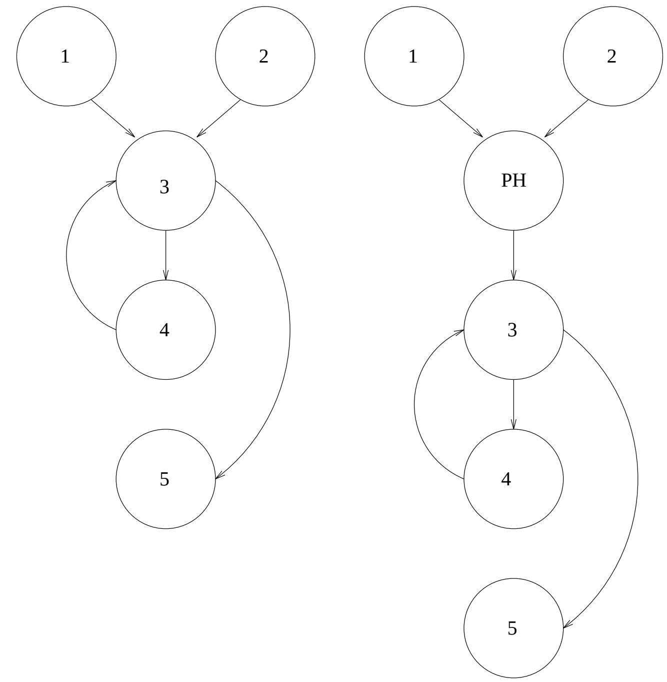


Aside: we need a basic block for code motion



- landing pad
- control dependence graph

Landing pad (Preheaders)



Next Time

Static Single Assignment

Read: Cytron et al. "Efficiently Computing Static Single Assignment Form and the Control Dependence Graph, TOPLAS 13(4), Oct 1991, pp. 451-490.