Dominator ...

\[ M \text{ dom } N \]

\[ M \text{ sdom } N \text{ strict} \]

\[ M \text{ idom } N \text{ immediate} \]

iff \[ M \] is on all paths from START to \( N \)

iff \[ M \text{ dom } N \text{ and } M \neq N \]

iff \[ M \text{ sdom } N \text{ and} \]

\((P \text{ sdom } N) \implies (P \text{ dom } M)\)

**Proof:** assume \( M_1, M_2 \) \( M_1 \text{ dom } N, M_2 \text{ dom } N \) two immediate dominators?

\( M_1, M_2, \ldots, M_k \) must appear on all paths
P can be made acyclic.

Node, $H_j$ dom $N$ must appear on acyclic path.

So, $H_1, H_2, \ldots, H_k$ will appear in some order.

Last node $H_j$ (on path).

Node that precedes $N$.

$H_j$ would be immediate dominator.

But $H_i$ is also immediate dominator $\text{dom}(H_i \neq H_j)$.

Go directly $H_i \rightarrow N$, excludes $H_j$. 

\[ A \quad B \quad C \quad D \quad E \quad F \quad E \quad F \]
\[ \text{dom}(N) = \{ N \} \cup \left( \bigcup_{M \text{ predecessor of } N} \text{dom}(M) \right) \]

\[ \text{Dom In } (\text{START}) = \emptyset \]

\[ \text{Dom In } (b) = \bigcap_{c \text{ predecessor of } (b)} \text{Dom Out } (c) \]

\[ \text{Dom Out } (b) = \text{Dom In } (b) \cup \{ b \} \]
How should we initialize $\text{DomOut}(x)$, $x \neq \text{Start node}$?

Proposition 1: $\text{DomOut}(x) = \emptyset$

Consider the loop

\[
\text{START} \quad \downarrow \quad A \quad \downarrow \quad B \quad \downarrow \quad C
\]

\[
\text{dom}(B) = \text{domOut}(B \cup \{B\}) = \{B\} \cup \text{DomOut}(B)
\]

If we start iteration with $\text{DomOut}(B) = \emptyset$

\[
\text{domOut}(B) = \{B\}
\]
2 options:

a) Initial out set differently

\[ \text{init } \text{dom out}(x) = \{ x \} \]

set of all nodes

\( x \neq \text{START} \)

\[ \text{dom out}(B) = \{ B \} \cup \text{dom out}(A) \]

b) order computation and exclude nodes that have not had dom out computed from \( \cap \) of predecessors dom M

...
For option a) 
algorithm based on worklist 

For (each n ∈ NodeSet) 
I: set it all nodes 
    dom (n) = NodeSet; 

worklist = { START } 

dom (START) = Ø; 

while ( worklist ≠ Ø ) { 
    pick y from worklist, remove y; 
    New = 2y3 U ( ∪ dom (x)) 
        x ∈ pred (y) 
    if (New ≠ dom (y)) { 
        dom (y) = New; 
        for (each z ∈ succ (y)) { 
            worklist = worklist U z 
        }
    }
}
worklist alg:

$O(N^2)$ ops, $N$: number of nodes

$O(N \log N + E)$, $E$: number of edges

$O(N \cdot \alpha(N,E))$

\[\alpha\text{-Ackermann function grows slowly}\]

- Propensity of real programs helps
- Simple CFG allows optimization...
\[
\text{Dom}(B) = \{B\} \cup \left( i\text{Dom}(B) \right) \cup \\
\left( i\text{Dom}(i\text{Dom}(B)) \right) \cup \\
\left( i\text{Dom}(i\text{Dom}(i\text{Dom}(B))) \right) \cup \cdots \\
\cup \{\text{START}\}
\]

Depth - First Spanning Tree

For CFG \( \leq 1000 \) nodes

\( N^2 \quad 3 \times \text{speed} \quad N\log N + E \)
\[ DF_{\text{local}}(x) = \{ y \mid y \in \text{succ}(x) \} \]

\[ x \not\rightarrow y \]

\[ DF_{\text{up}}(z) = \{ y \mid y \in DF(z) \text{ and } \text{idom}(z) \not\rightarrow y \} \]

\[ DF(x) = DF_{\text{local}}(x) \cup \bigcup_{z \in \text{children}(x)} DF_{\text{up}}(z) \]
To make computation of \( DF_{\text{local}} \) more efficient,

\[
DF_{\text{local}} (x) = \{ y \mid y \in \text{succ} (x) \text{ and } \text{idom} (y) \neq x \}
\]

**Proof:**

Given \( y \in \text{succ} (x) \)

\[(x \gg y) \iff (\text{idom} (y) = x)\]

\[\leq^*\]

\[x = \text{idom} (y) : x \neq y, \ x \in \text{sdom} y \]

\[x \gg y\]
"\Rightarrow" assume \( X \) dom \( Y \) and hence some \( V \) if \( X \) that dominates \( Y \).

Can there be a \( V \)?

\( V \) appears on any path from \( \text{START} \) to \( Y \) (that path must include \( X \)).

Edge \( X \rightarrow V \rightarrow Y \) or \( V \rightarrow X \rightarrow Y \).

Either this means \( V \) dom \( X \) is possible iff \( V = X \).

\( V \) is child, so \( V \) dom \( X \) is not possible.

So \( \text{idom}(Y) = \text{idom}(V) = X \).
We remark for any node \( x \) with children \( t \) (child in DT),

\[
\text{DF}_{\text{up}}(z) = \{ y \mid y \in \text{DF}(z) \text{ and } \text{idom}(y) \neq x \}.
\]

Compute DF:
for (each \( x \) in NodeList, bottom-up traversal of DT,)

\[
\text{DF}(x) \leftarrow \emptyset
\]
for (each \( y \in \text{succ}(x) \) \&

// DF local

if (\( \text{idom}(y) \neq x \) \& \( \text{DF}(x) = \text{DF}(x) \cup \{y\} \))

}
\[\text{for (each } t \in \text{children}(x)) \{\]
\[\quad \text{DFup}\]
\[\quad \text{for (each } y \in \text{DF}(t)) \{\]
\[\quad \quad \text{if } (\text{idom}(y) \neq x) \{\]
\[\quad \quad \quad \text{DF}(x) = \text{DF}(x) \cup \{y\}\]
\[\quad \quad \}\]
\[\quad \}\]
\[\}\]