

Static Single Assignment Form

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

SSA Form

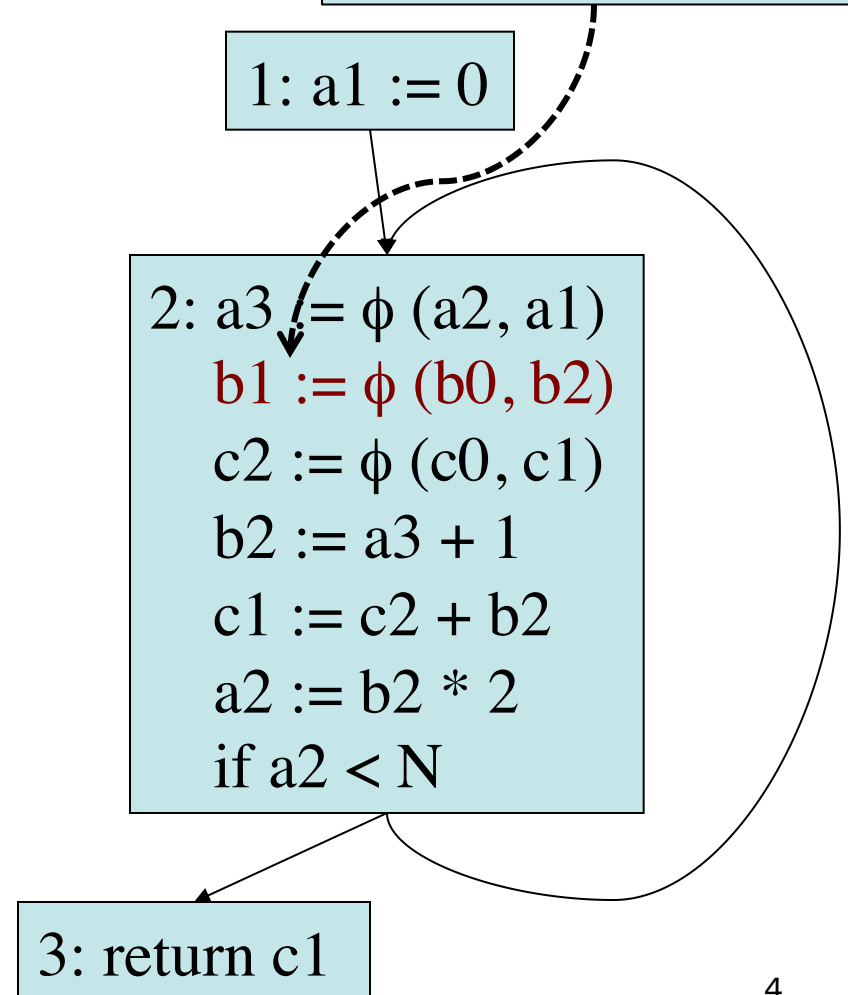
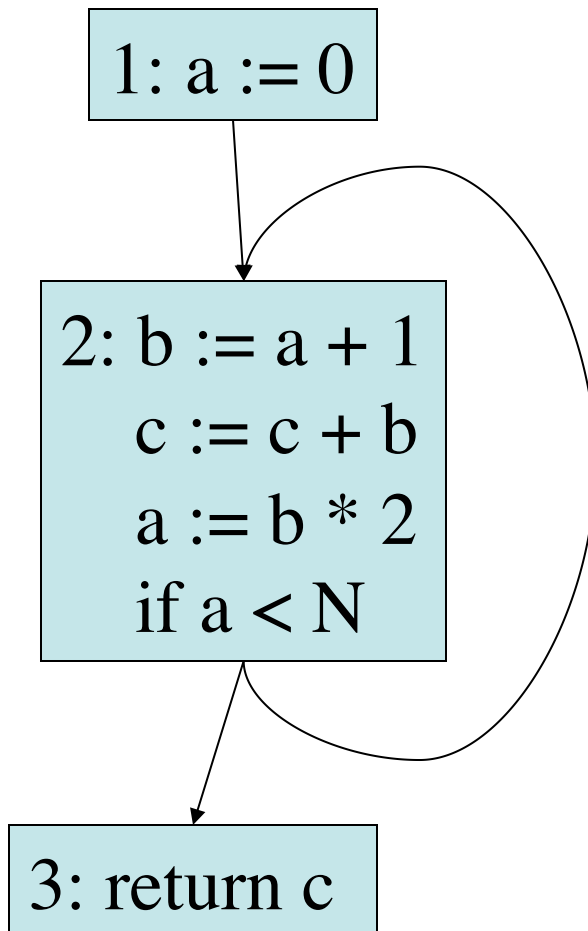
- Conversion from a Control Flow Graph (created from TAC) into SSA Form is not trivial
- SSA creation algorithms:
 - Original algorithm by Cytron et al. 1986
 - Lengauer-Tarjan algorithm (see the Tiger book by Andrew W. Appel for more details)
 - Harel algorithm

Conversion to SSA Form

- Simple idea: add a ϕ function for every variable at a join point
- A join point is any node in the control-flow graph with more than one predecessor
- But: this is wasteful and unnecessary.

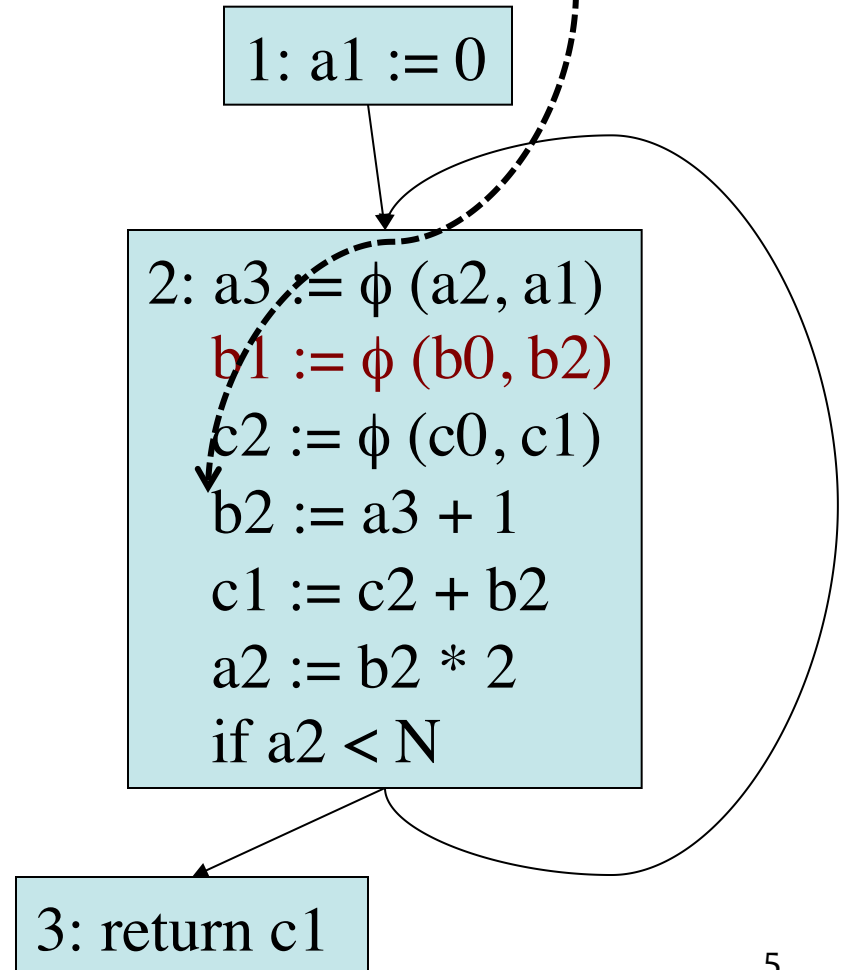
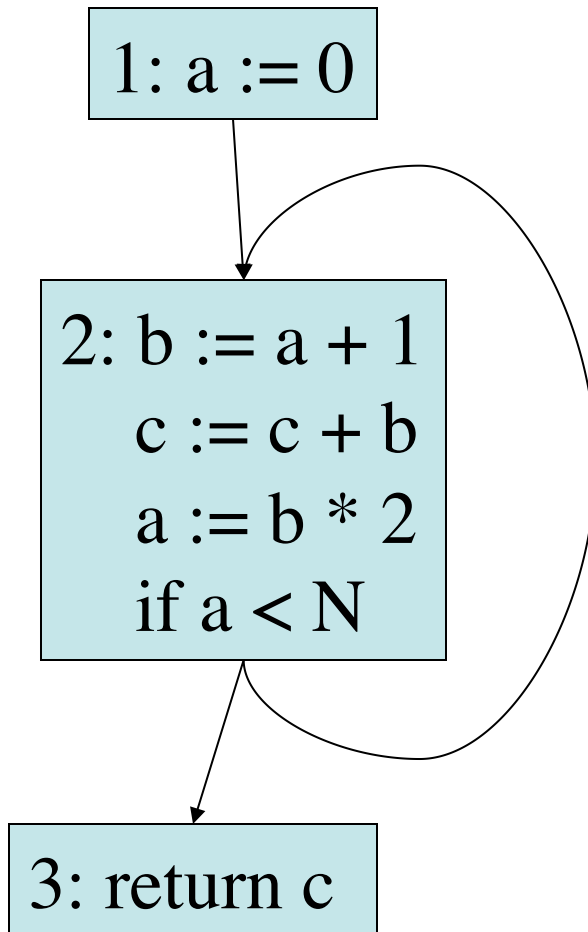
Conversion to SSA

b1 is never used,
stmt can be deleted



Conversion to SSA

b2 changes in each loop. SSA is **not** functional programming!

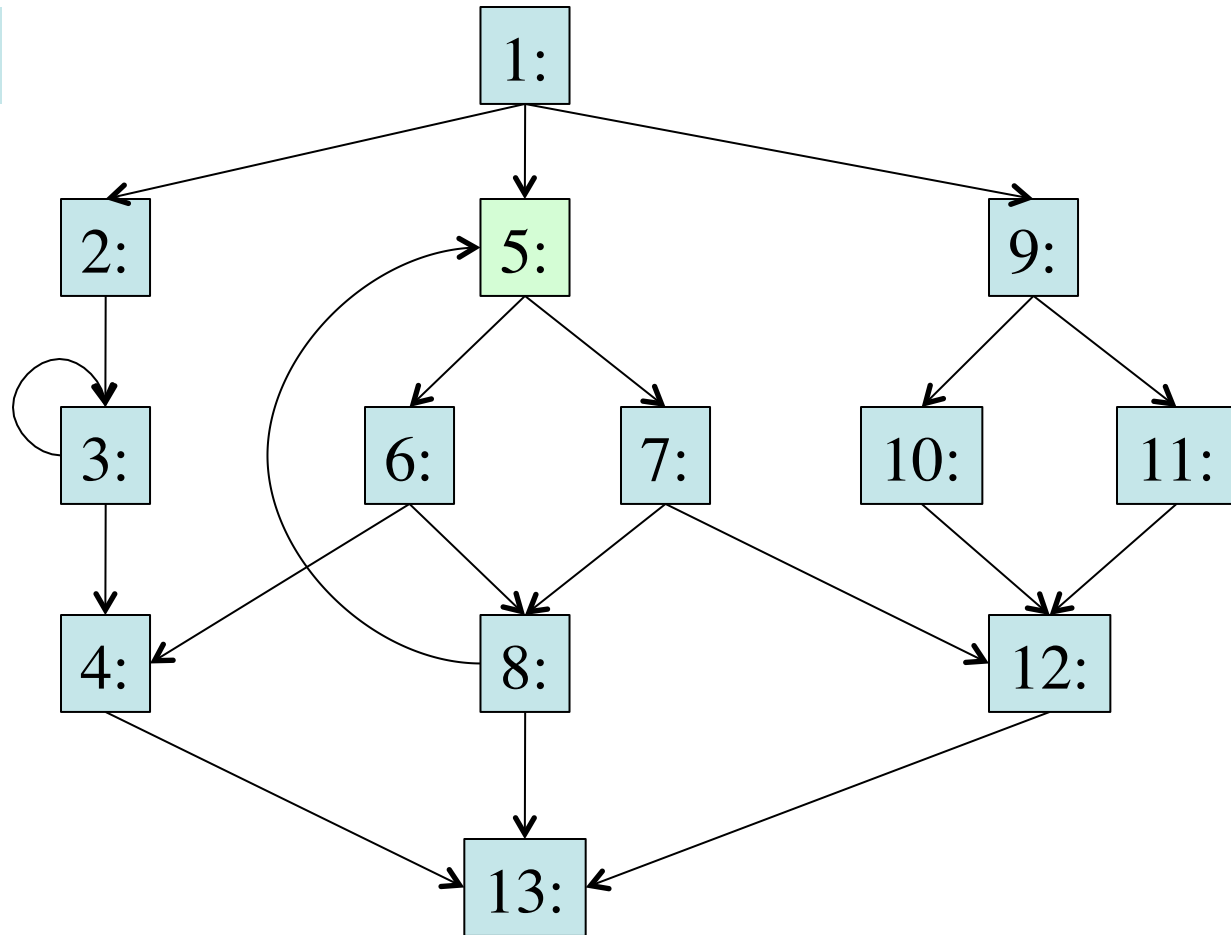


Dominance Relation

- X *dominates* Y if every path from the start node to Y goes through X
- $D(X)$ is the set of nodes that X dominates
- X *strictly dominates* Y if X dominates Y and $X \neq Y$

Dominance Relation

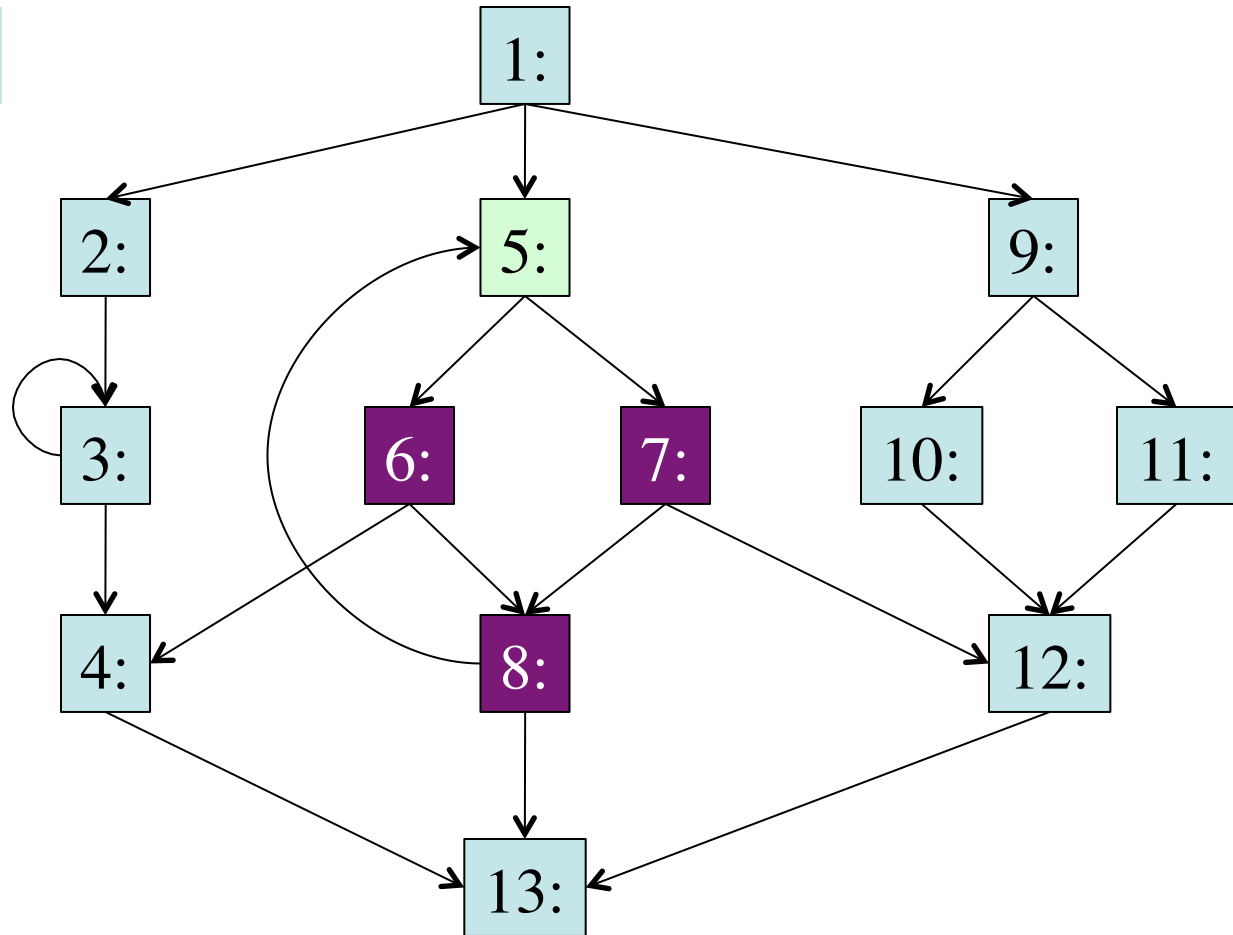
$D(5)=\{6,7,8\}$



5 strictly
dominates
6, 7, 8

Dominance Relation

$D(5) = \{6, 7, 8\}$



5 strictly
dominates
6, 7, 8

Dominance Property of SSA

- Essential property of SSA form is the definition of a variable must *dominate* use of the variable:
 - If X is used in a ϕ function in block n , then definition of X dominates every predecessor of n
 - If X is used in a non- ϕ statement in block n , then the definition of X dominates n .

Dominance Frontier

- *X strictly dominates Y* if X dominates Y and $X \neq Y$
- *Dominance Frontier (DF)* of node X is the set of all nodes Y such that:
 - X dominates a predecessor of Y, AND
 - X does not strictly dominate Y

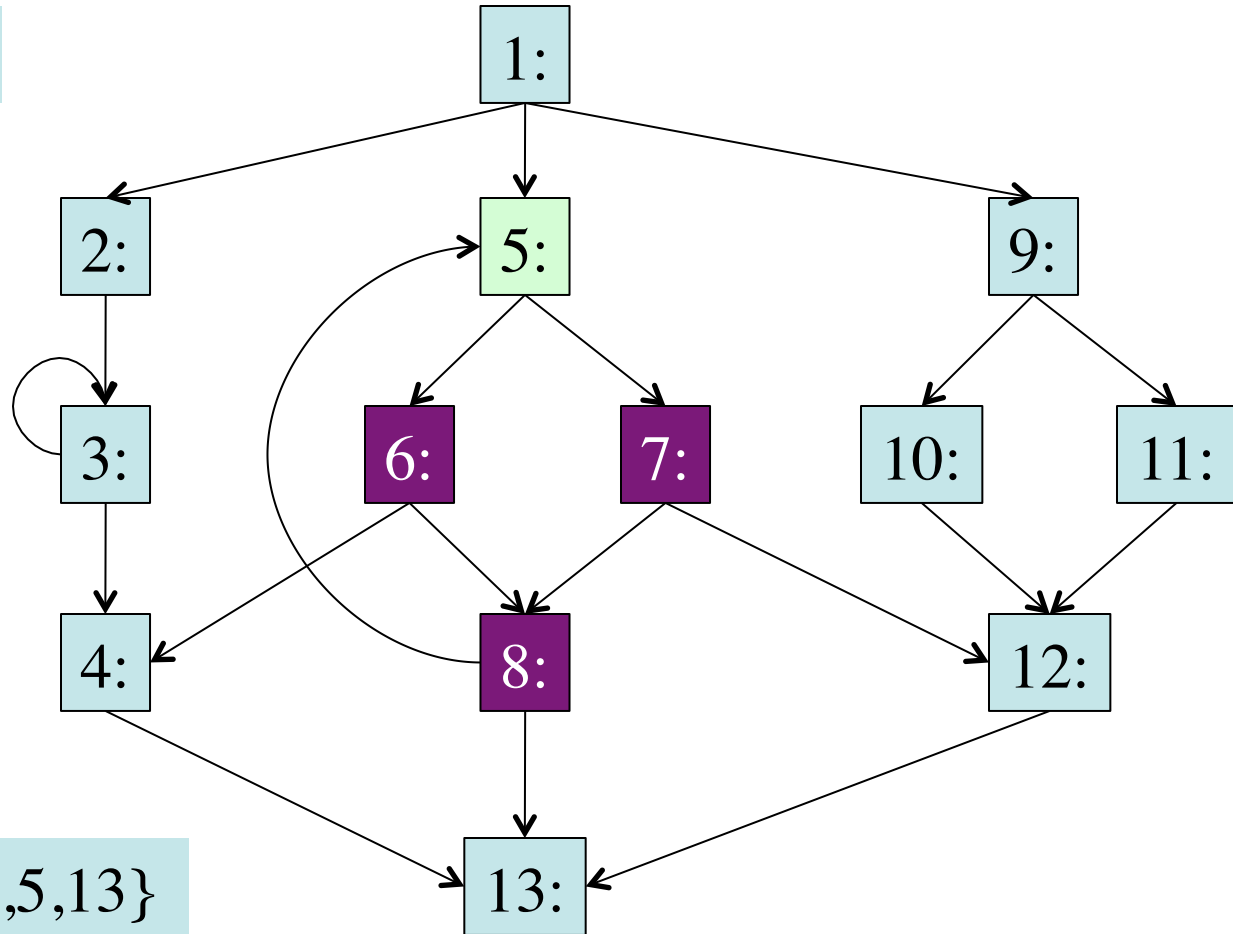
Dominance Frontier

$D(5) = \{6, 7, 8\}$

$S(6) = \{4, 8\}$

$S(7) = \{8, 12\}$

$S(8) = \{5, 13\}$



$DF(5) = \{4, 12, 5, 13\}$

Dominance Frontier

- Algorithm to compute $DF(X)$:
 - $Local(X) :=$ set of successors of X who do not immediately dominate X
 - $Up(X) :=$ set of nodes in $DF(X)$ that are not dominated by X 's immediate dominator.
 - $DF(X) :=$ Union of $Local(X)$ & (Union of $Up(K)$ for all K that are children of X)

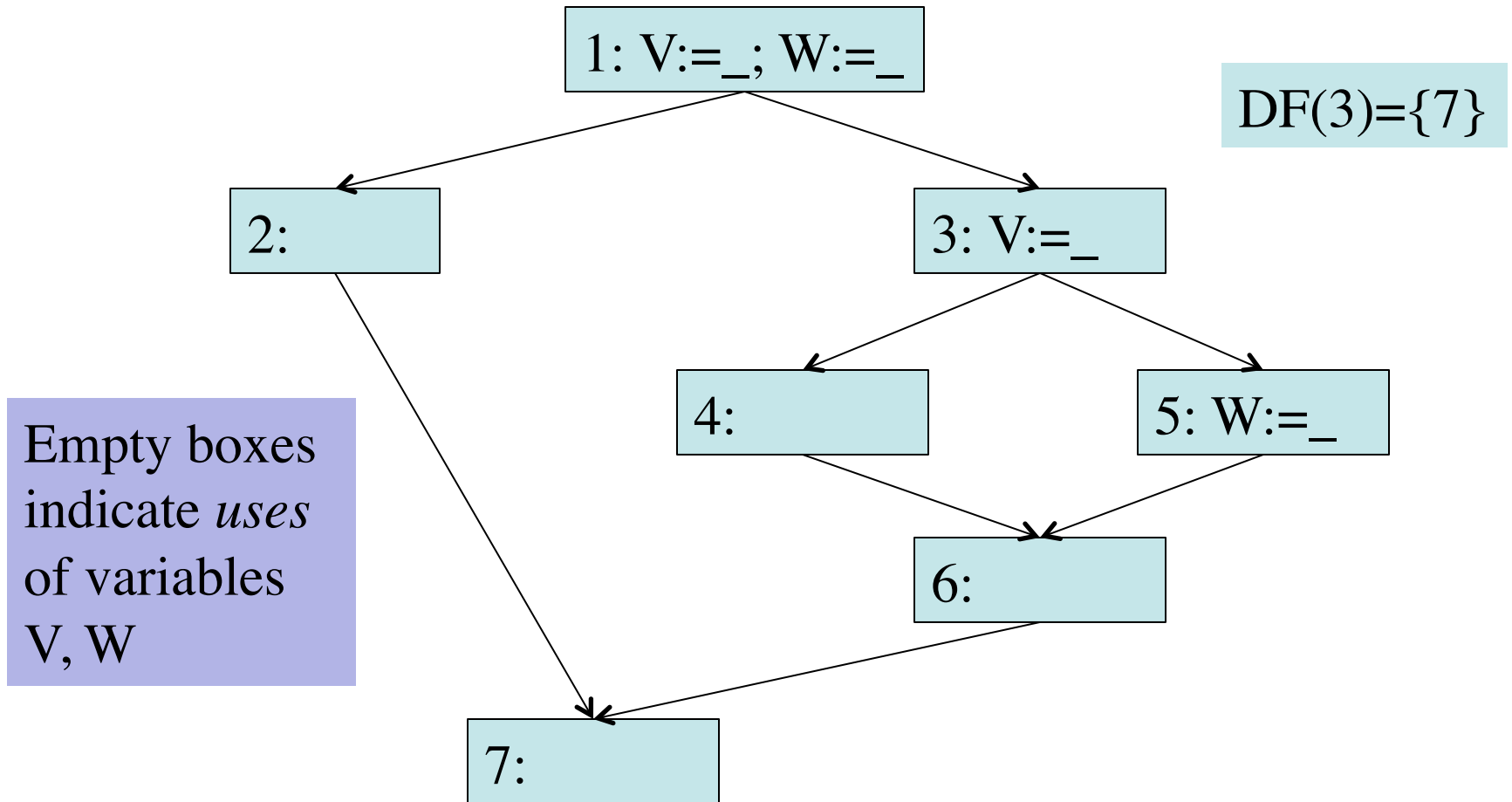
Dominance Frontier

- ComputeDF(X):
 - $S := \{\}$ // empty set
 - For each node Y in Successor(X):
 - If Y is not immediately dominating X:
 - $S := S + \{Y\}$ // this is Local(X), + means union
 - For each child K of X in D(X): // X dominates K
 - For each element Y in ComputeDF(K):
 - If X does not dominate Y,
 - $S := S + \{Y\}$ // this is Up(X)
 - DF(X) = S

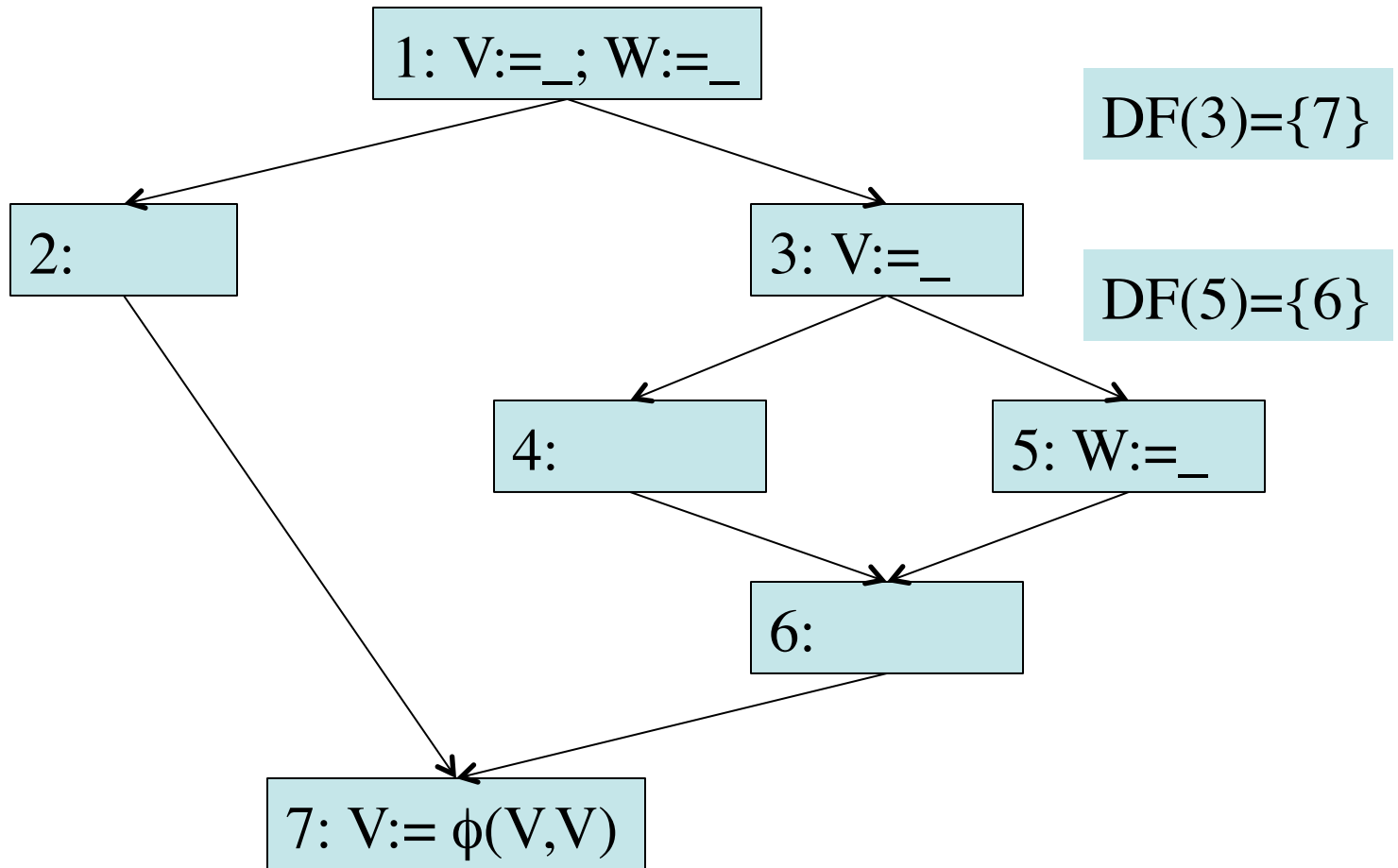
Dominance Frontier

- Dominance Frontier Criterion
 - If node X contains definition of some variable a , then any node Y in the $DF(X)$ needs a ϕ function for a .
- Iterated Dominance Frontier
 - Since a ϕ function is itself a definition of a new variable, we must iterate the DF criterion until no nodes in the CFG need a ϕ function.

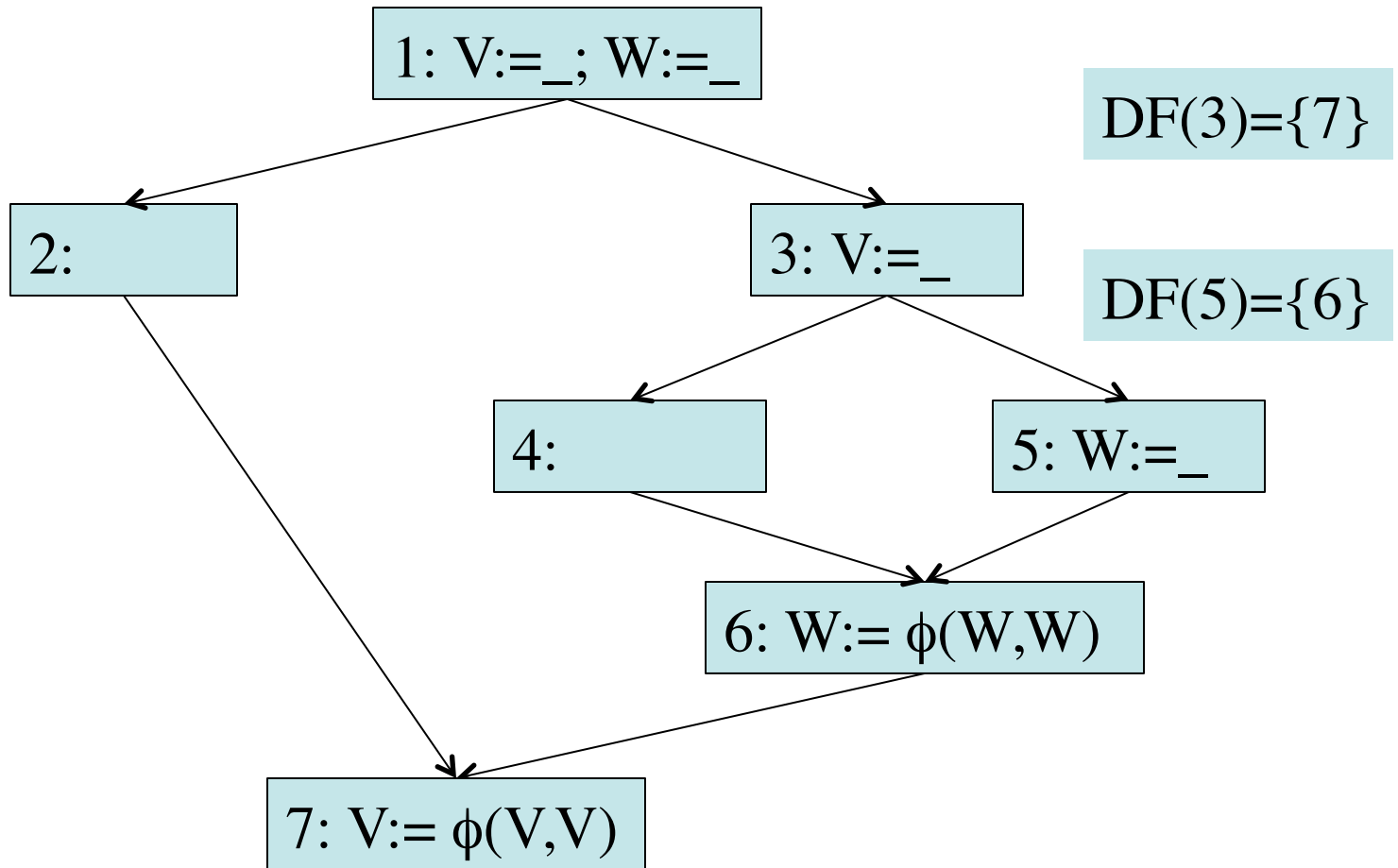
Placing ϕ Functions



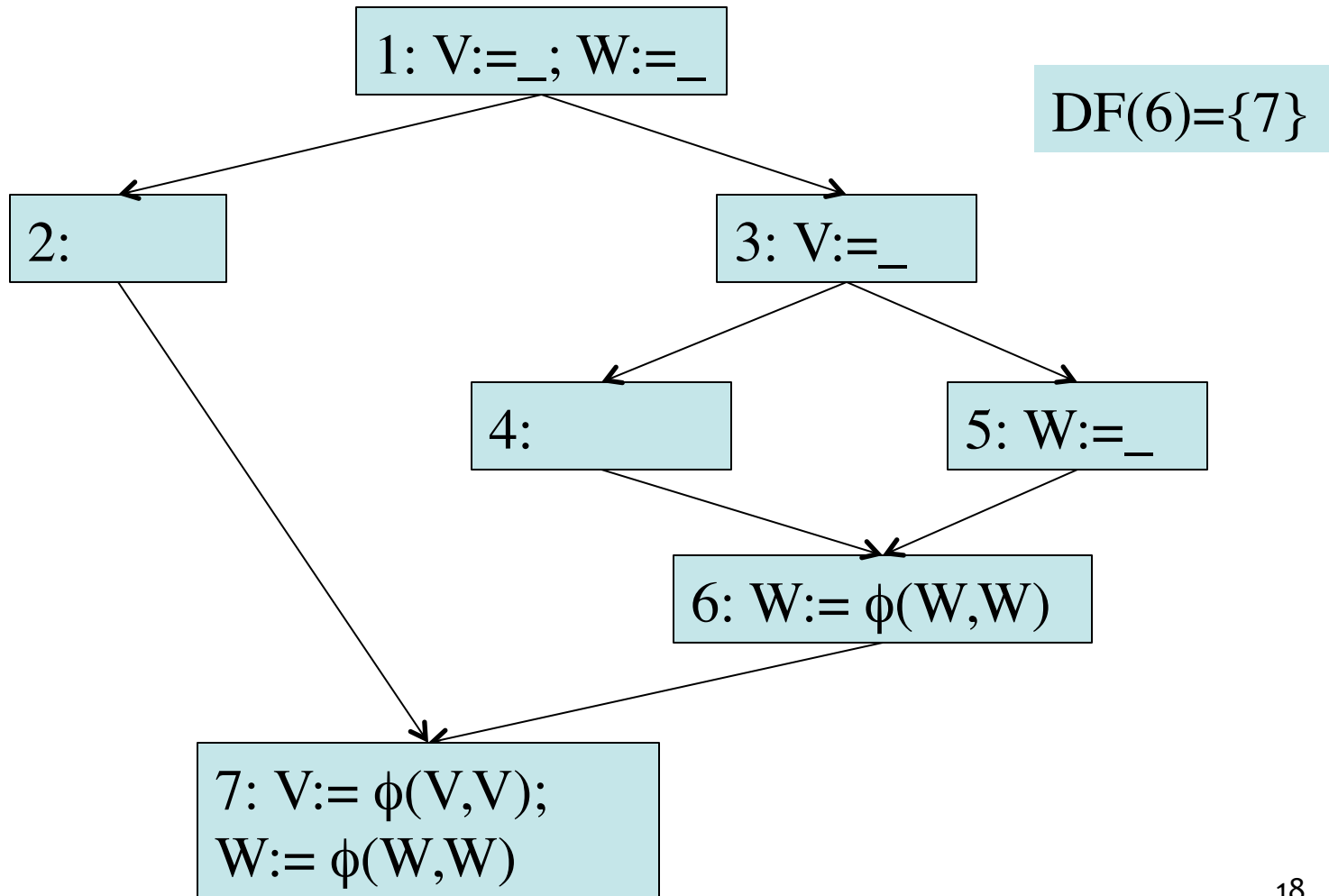
Placing ϕ Functions



Placing ϕ Functions



Placing ϕ Functions



Rename Variables

