Putting Programs into SSA Form

Assume we have the CFG for a program, which we want to put into SSA form. We must:

- Rename all definitions and uses of variables
- Decide where to add phi functions

Renaming variable definitions is trivial—each assignment is to a new, unique variable.

After phi functions are added (at the heads of selected basic blocks), only one variable definition (the most recent in the block) can reach any use. Thus renaming uses of variables is easy.

Placing Phi Functions

Let b be a block with a definition to some variable, v. If b contains more than one definition to v, the last (or most recent) applies.

What is the first basic block following b where some other definition to v *as well as* b's definition can reach?

In blocks dominated by b, b's definition *must* have been executed, though other later definitions may have overwritten b's definition.

Domination Frontiers (Again)

Recall that the Domination Frontier of a block b, is defined as

$$DF(N) = \{Z \mid M \rightarrow Z \ \& \ (N \text{ dom } M) \ \& \ \neg(N \text{ sdom } Z)\}$$

The Dominance Frontier of a basic block N, DF(N), is the set of all blocks that are immediate successors to blocks dominated by N, but which aren't themselves strictly dominated by N.

Assume that an initial assignment to all variables occurs in b_0 (possibly of some special "uninitialized value.")

We will need to place a phi function at the start of all blocks in b's Domination Frontier.

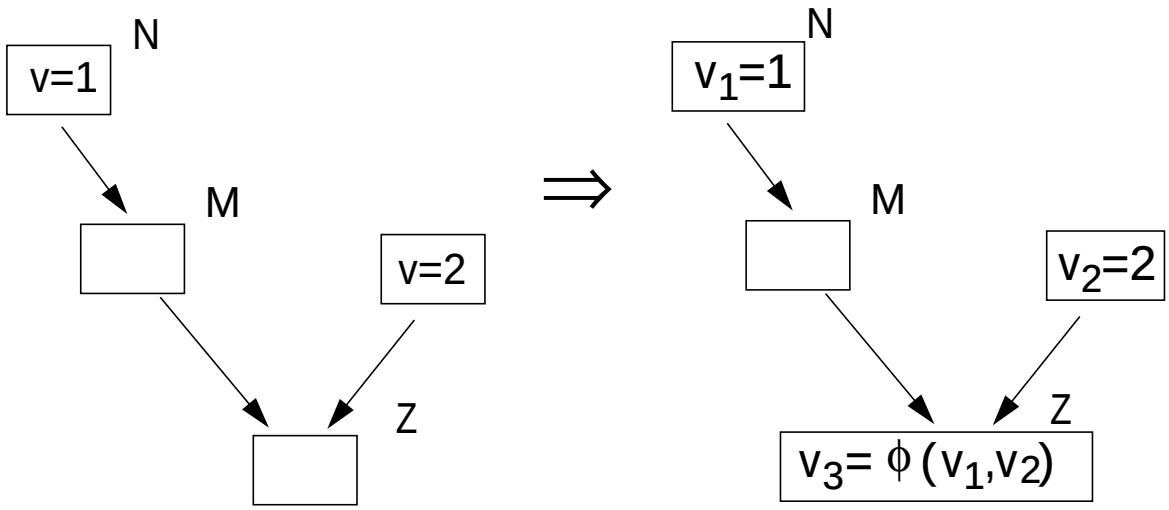
The phi functions will join the definition to v that occurred in b (or in a block dominated by b) with definitions occurring on paths that don't include b.

After phi functions are added to blocks in DF(b), the domination frontier of blocks with newly added phi's will need to be computed (since phi functions imply assignment to a new v_i variable).

Examples of How Domination Frontiers Guide Phi Placement

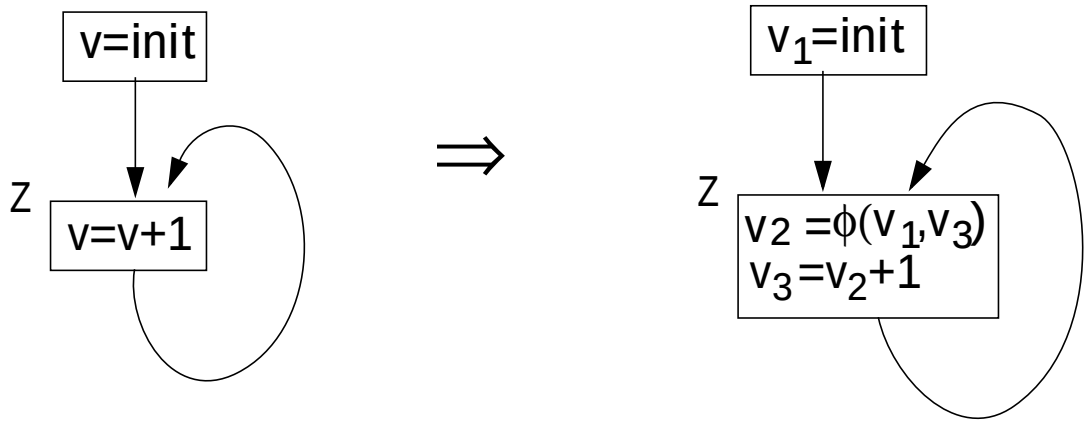
DF(N) =
{Z | M→Z & (N dom M) &
¬(N sdom Z)}

Simple Case:



Here, (N dom M) but ¬(N sdom Z),
so a phi function is needed in Z.

Loop:



Here, let M = Z = N. M→Z,

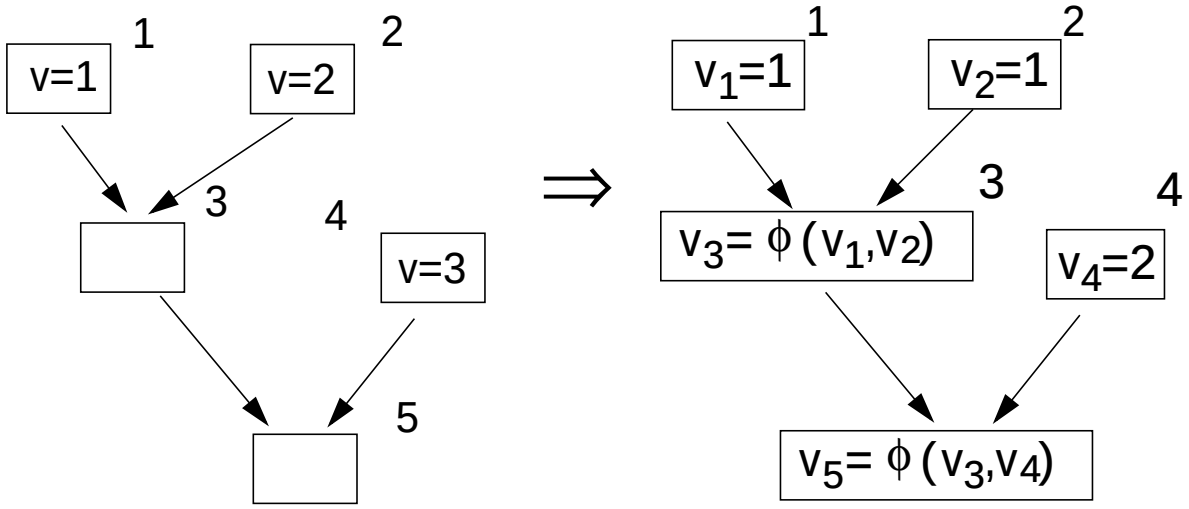
(N dom M) but ¬(N sdom Z),

so a phi function *is* needed in Z.

DF(N) =

{Z | M→Z & (N dom M) &
¬(N sdom Z)}

Sometimes Phi's must be Placed Iteratively



Now, DF(b1) = {b3}, so we add a phi function in b3. This adds an assignment into b3. We then look at DF(b3) = {b5}, so another phi function must be added to b5.