CS614: Advanced Compilers

SSA Construction. Sparse Simple CP

Manas Thakur
CSE, IIT Bombay

Spring 2025

Things we learnt in the last class

- Flow-sensitive constant propagation (IDFA)
  - Fits well within worklist-based implementation too
- Flow-insensitive constant propagation
  - Much faster than flow-sensitive but imprecise

We don't remember days,
we remember moments.

- Cesare Pavese

- SSA form as a program representation
  - Each variable is assigned to at a single statement
  - Each use has exactly one corresponding definition
  - Achieved using variable renaming and insertion of Φ functions

Insertion of Φ functions

- Intuitively:
  - If two paths in a CFG with a definition of a variable v converge at a node n, then we need a Φ function at node n.
  - The number of arguments of the Φ function is the same as the *in-degree* of n.

- A Φ function is also an assignment to the variable being addressed, so a Φ-insertion may lead to the insertion of more Φ-assignments at other nodes.

- Let's devise a *simple* forward algorithm to insert Φ functions.

Φ Insertion (Cont.)

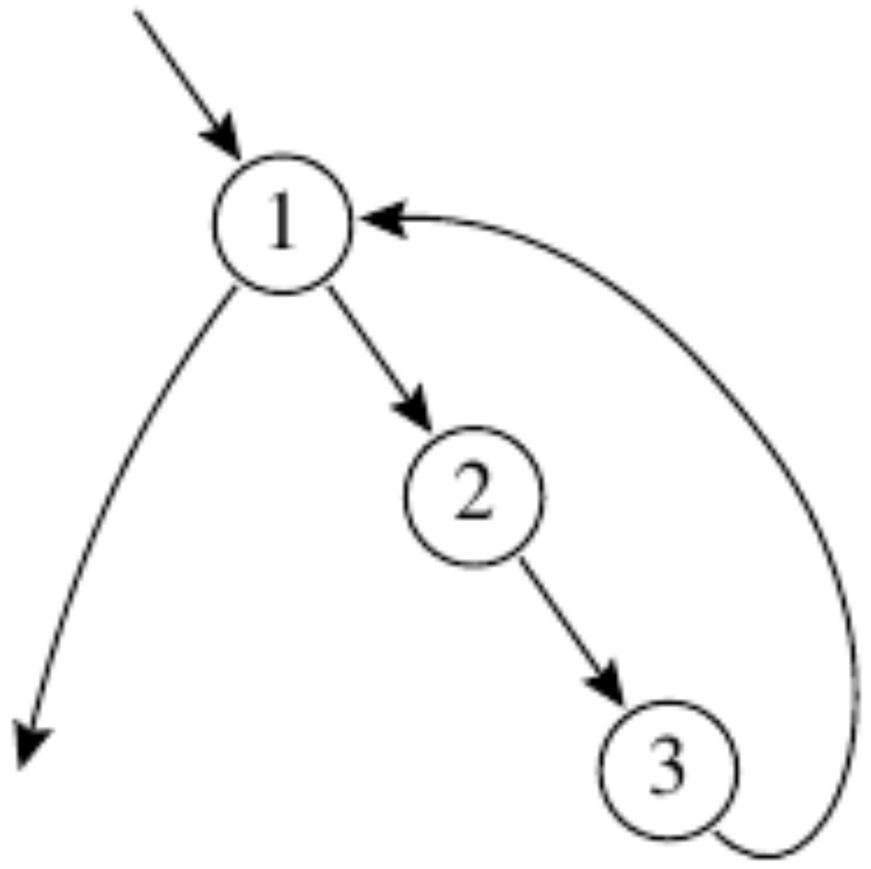
- Naive approach:
  - For each variable, insert a Φ function at the head of a basic block with multiple predecessors.
  - Too many unnecessary Φ functions.
- We truly need a Φ function for a variable *v* at a node *m* when:
  - There are distinct nodes *x* and *y* that define *v*; and
  - There are two non-empty paths *x* —*>*m* and *y* —*>*m* that are disjoint except the final node, such that *v* is defined only at *x* and *y*.

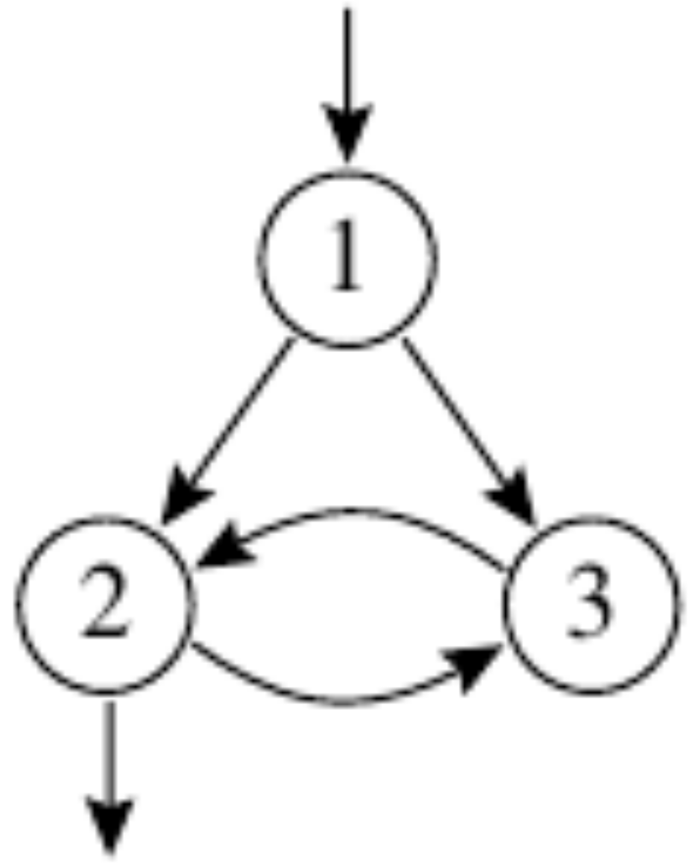
Path Convergence Criterion

Too expensive to compute.

Dominators

- A node *d* dominates a node *n* if every path from *entry* to *n* goes through *d*.
  - Every node dominates itself.
- Compute dominators of each node:

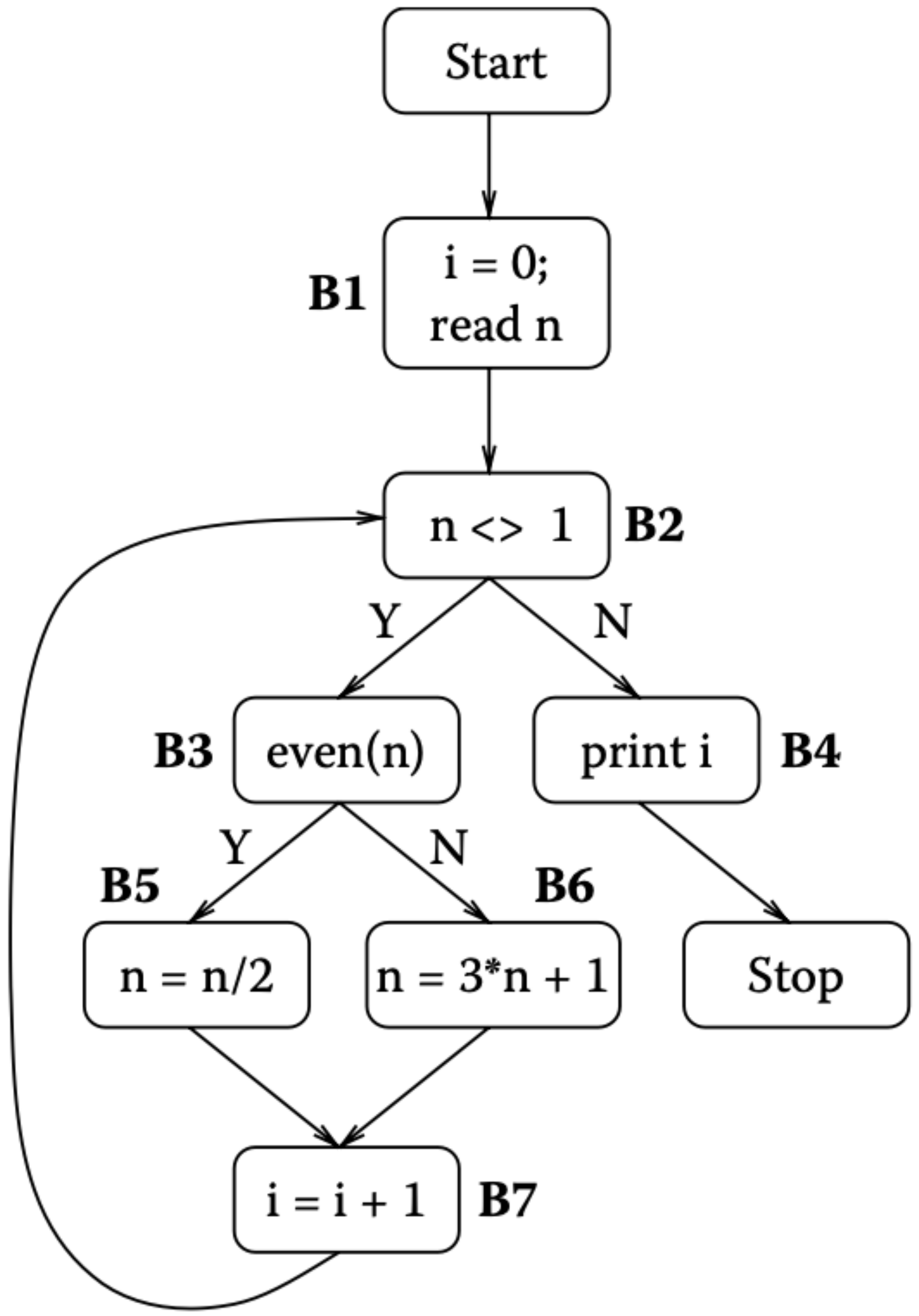


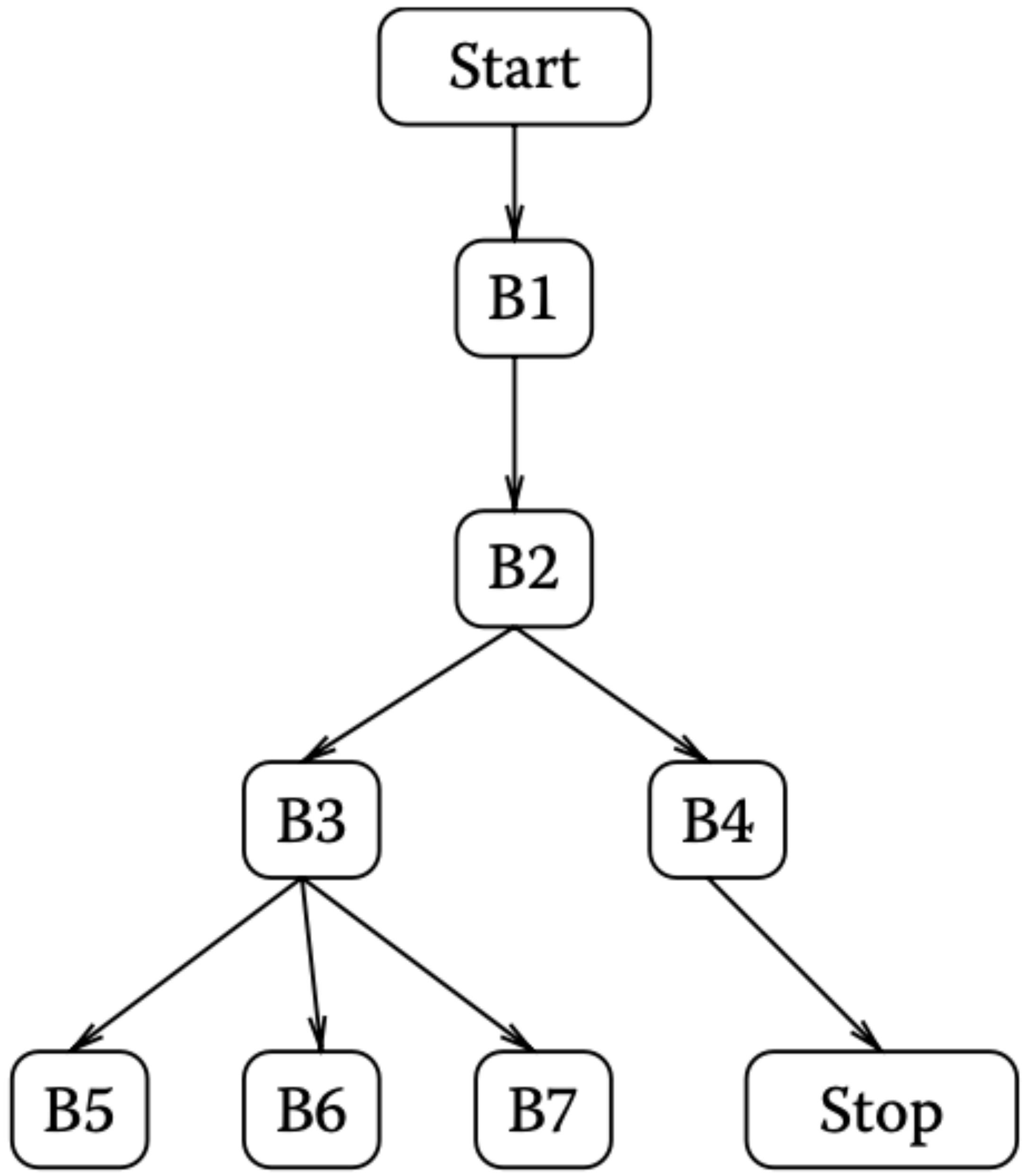


- Node x strictly dominates node y if it dominates y and $x \neq y$.
- Node x immediately dominates node y if x is the closest strict dominator of y.

Dominator Tree

- Parent-child relationship between a node and its immediate dominees.



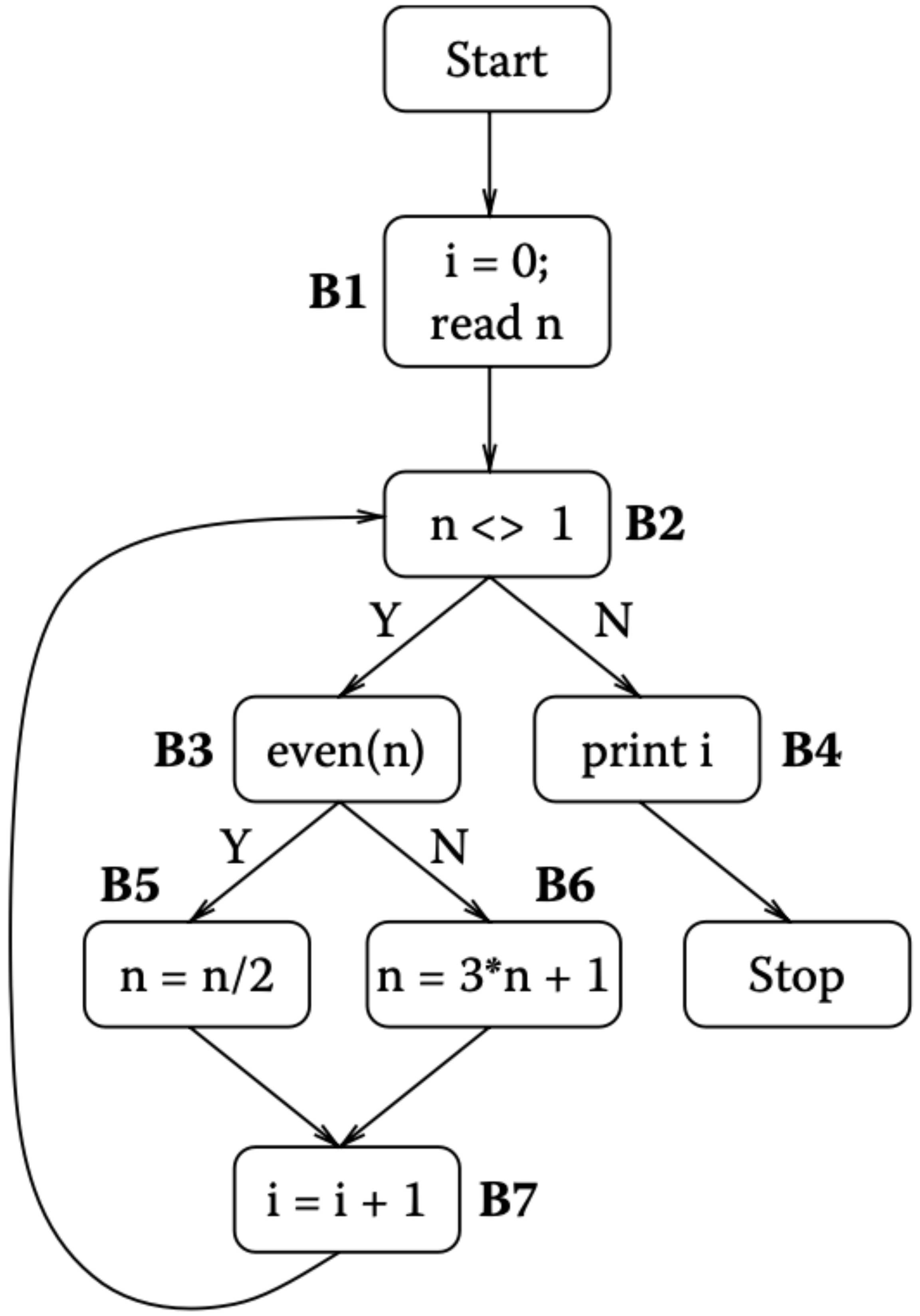


Dominance Frontier

- For a node x, the set of all nodes y such that x dominates a predecessor of y but does not strictly dominate y.

frontier: a border separating two regions.

DF(x): set of nodes that define *x*'s border of dominance.



- DF(B1) = {}
- DF(B2) = DF(B3) = {B2}
- DF(B4) = {}
- DF(B5) = {B7}
- DF(B6) = {B7}
- DF(B7) = {B2}

Iterated Dominance Frontier

- Iterated dominance frontier of a node x is the closure of $DF(x)$.

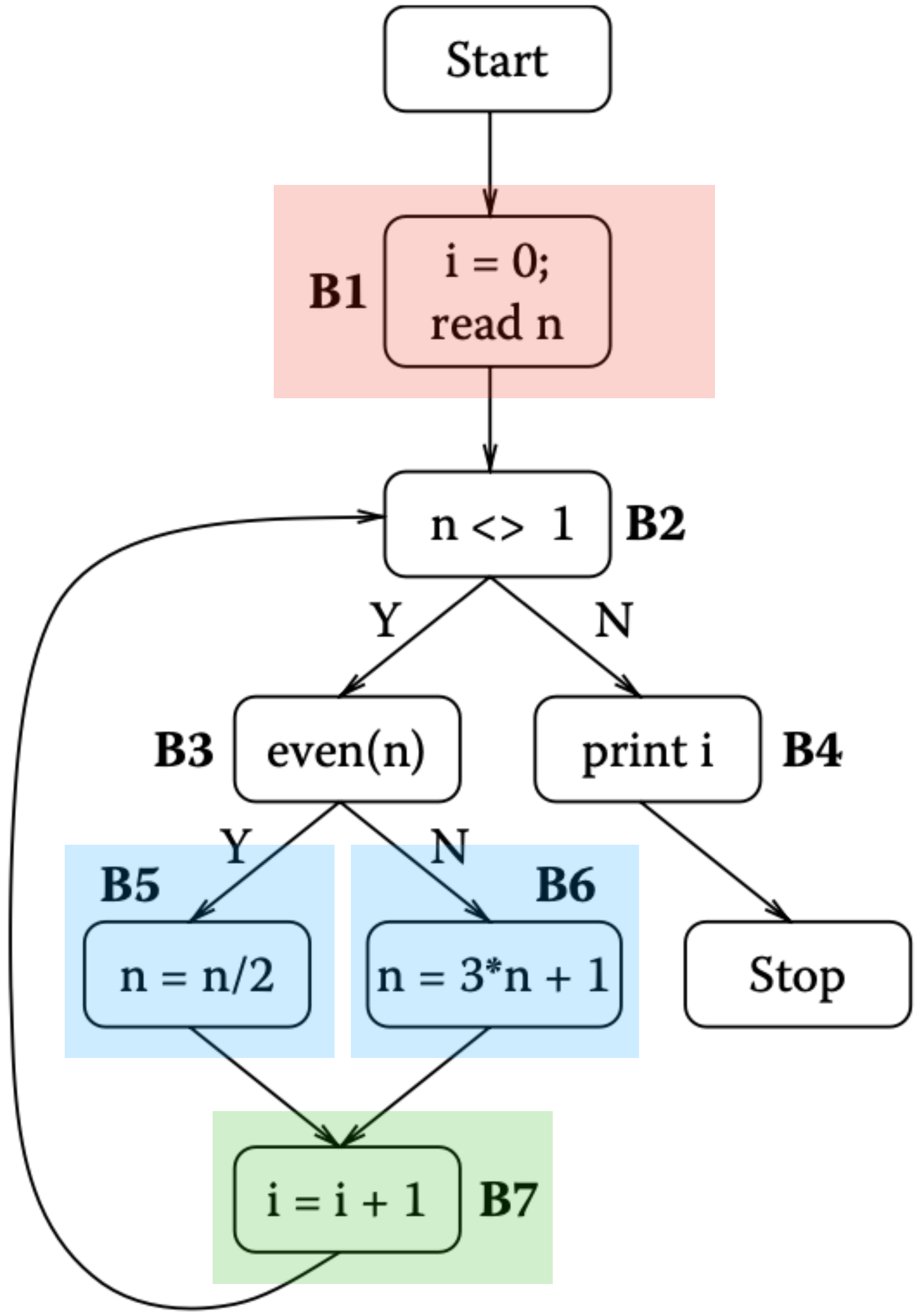
$$DF(\mathcal{S}) = \cup_{x \in \mathcal{S}} DF(x)$$
$$DF^{(1)}(\mathcal{S}) = DF(\mathcal{S})$$
$$DF^{(i+1)}(\mathcal{S}) = DF(\mathcal{S} \cup DF^{(i)}(\mathcal{S}))$$

- DF(B1) = {}
- DF(B2) = DF(B3) = {B2}
- DF(B4) = {}
- DF(B5) = {B7}
- DF(B6) = {B7}
- DF(B7) = {B2}

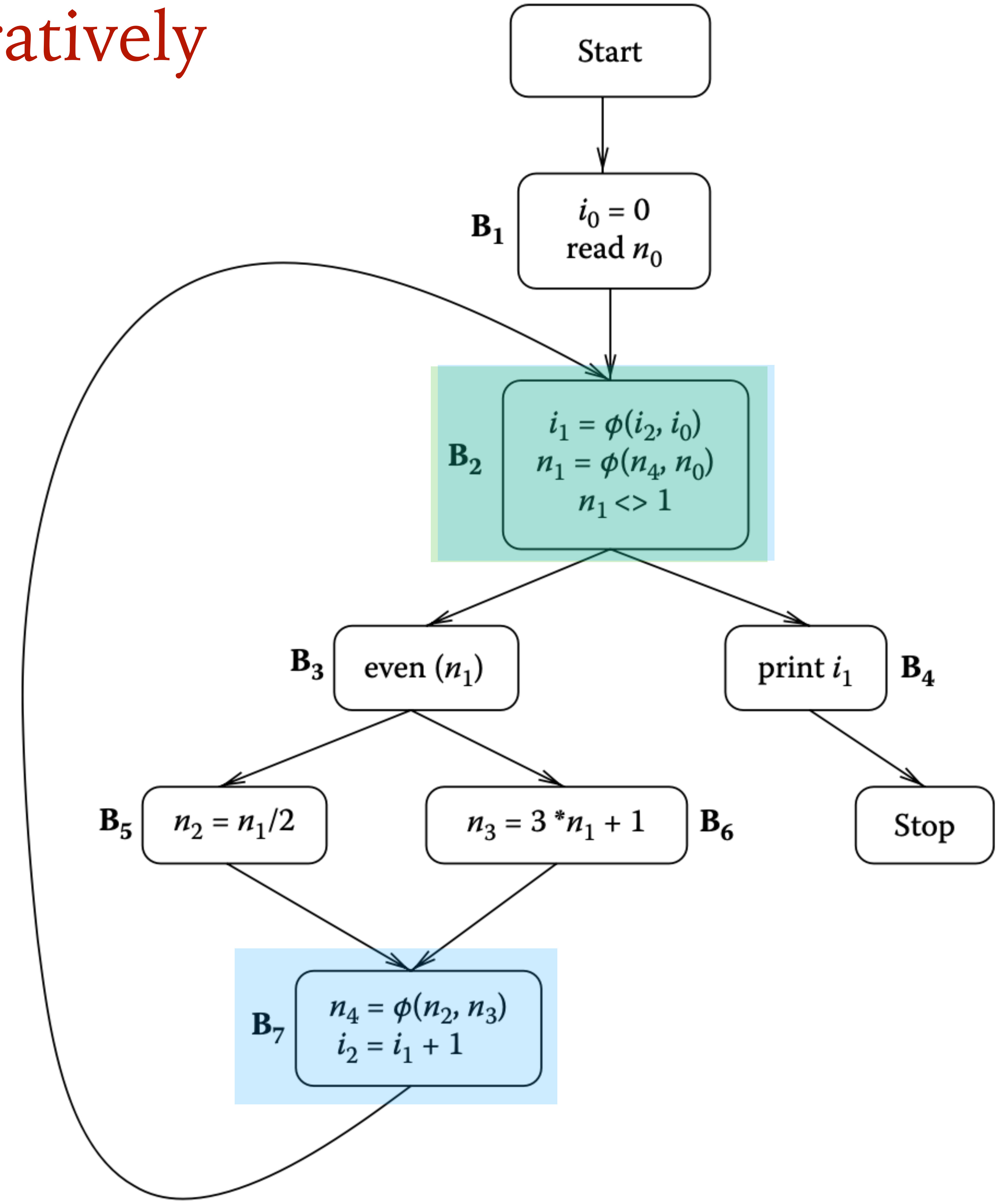
- For our example:
  - IDF(B1) = IDF(B4) = IDF(B5) = {}
  - IDF(B2) = IDF(B3) = {B2}
  - IDF(B5) = {B7, B2}
  - IDF(B6) = {B7, B2}
  - IDF(B7) = {B2}

Φ Insertion at IDFs

- For each variable v defined at a set S of nodes, iteratively insert Φ-functions at the nodes in IDF(S).



- IDF(B1) = {}
- IDF(B2) = IDF(B3) = {B2}
- IDF(B4) = IDF(B5) = {}
- IDF(B5) = {B7, B2}
- IDF(B6) = {B7, B2}
- IDF(B7) = {B2}



Path Convergence and Dominance Frontier

Path Convergence Criterion

- We truly need a Φ function for a variable v at a node m when:
  - There are distinct nodes x and y that define v; and
  - There are two non-empty paths x —*>m and y —*>m that are disjoint except the final node, such that v is defined only at x and y.

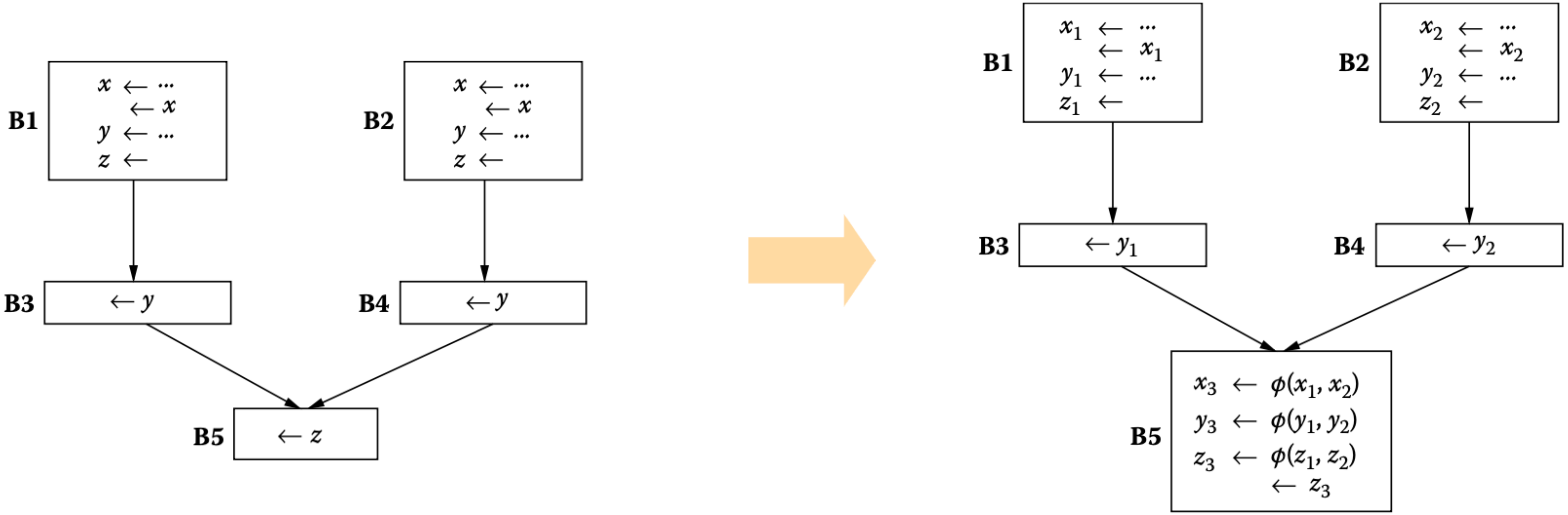
- Whenever a node x contains a definition of some variable v, then any node z in the dominance frontier of x needs a Φ function for v.

Dominance Frontier Criterion

- **Theorem.** The *iterated* path convergence criterion and the *iterated* dominance frontier criterion specify exactly the same set of nodes at which to insert Φ functions.

Φ Insertion — Reloaded

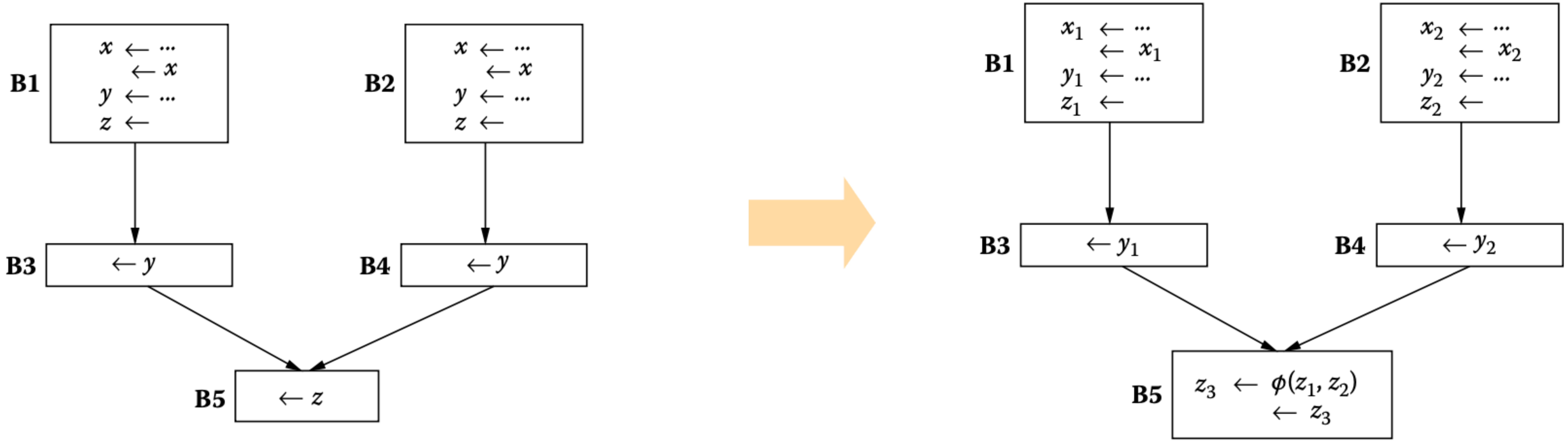
- Can our algorithm still insert redundant Φ functions?



- We don't need Φ functions for x and y at B5.

Pruned SSA Form

➤ Insert a Φ function for a variable v at node n only if v is *live(-in)* at n.



➤ Requires another IDFA, but liveness information is usually computed anyway.