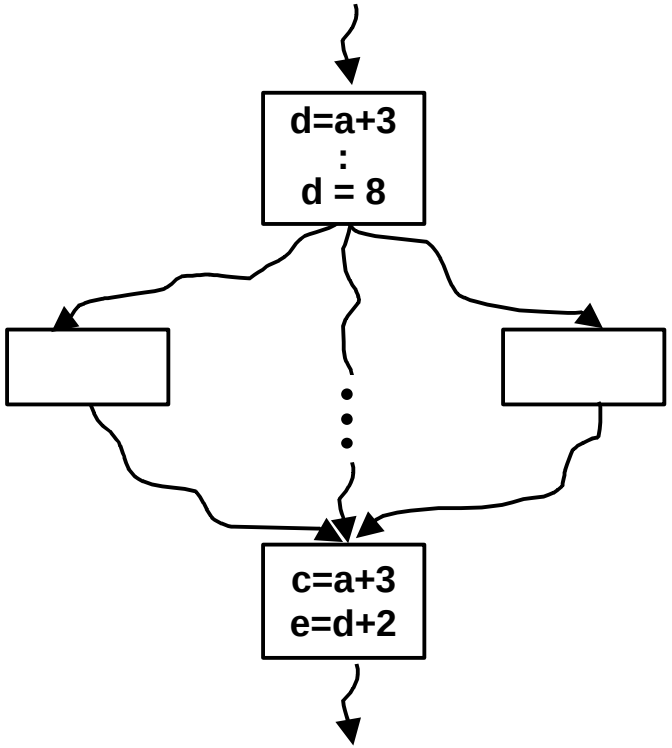
Data Flow Analysis

Data Flow Analysis

- Goal: make assertions about the data usage in a program
- Use these assertions to determine if and when optimizations are legal
- Local: within a single basic block
  - Analyze effect of each instruction
  - Compose effect at beginning/end of BB
- Global: within a procedure (across BBs)
  - Consider the effect of control flow
- Inter-procedural: across procedures



References:

Muchnick, Chapter 8.

Dragon Book, 608-611, 624-627, 631.

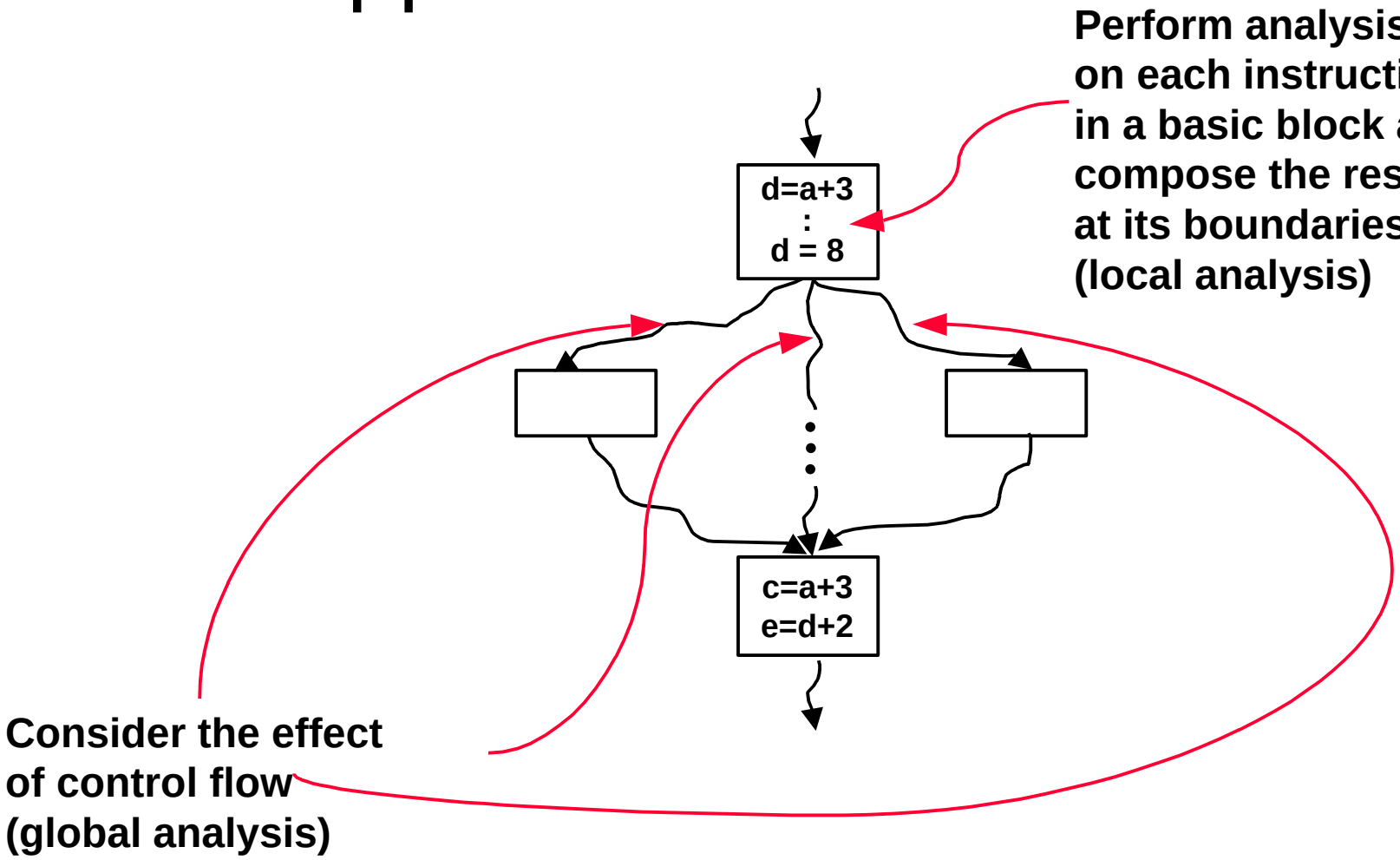
Data Flow Analysis

- *Compile-time* reasoning about the *run-time* flow of values in the program
- Represent facts about the run-time behavior
- Represent effect of executing each basic block
- Propagate facts around the control flow graph

Data Flow Analysis

- Formulated as a set of simultaneous equations
  - Sets attached to the nodes and edges
  - Lattice to describe the relation between values
  - Usually represented as a bit or bit vectors
- Solve equations using iterative framework
  - Start with initial guess of facts at each node
  - Propagate until stabilizes at *maximal fixed point*.
  - Would like *meet over all paths (MOP)* solution

Basic Approach:



Must be conservative!

Example: Reaching Definitions

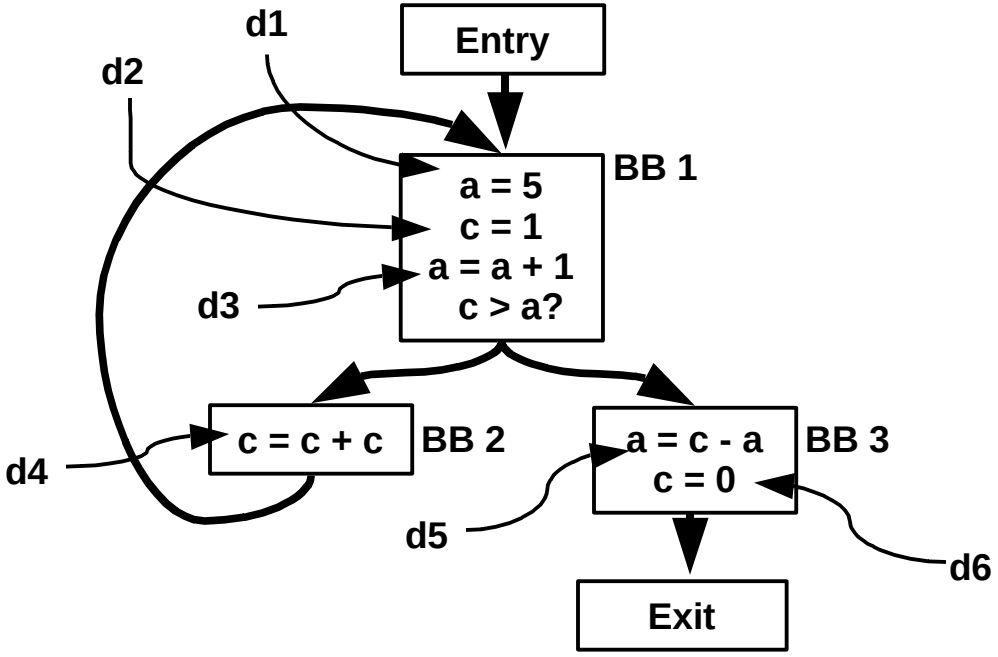
Problem statement: for each basic block b find which of all definitions in the program reach the boundaries of b.

Definition: A *definition* of a variable x is an instruction that assigns (or may assign) a value to x.

Reaches: A definition **d** of variable x ***reaches*** a point **p** in the program if there exists a path from the point immediately following **d** to **p** such that **d** is not ***killed*** by another definition of x along this path.

Reaching Definitions: Gen Set

Gen(b): the set of definitions that appear in a basic block b and reach its end.



Gen (BB 1) = {d2, d3}

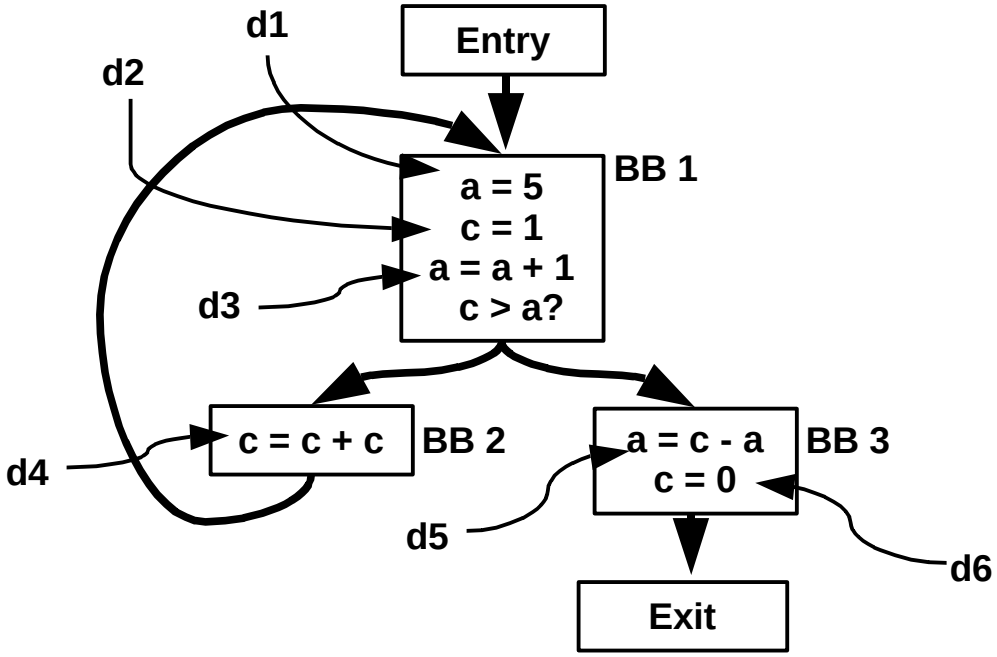
Gen (BB 2) = {d4}

Gen (BB 3) = {d5, d6}

Finding Gen(b) is doing *local* reaching definitions analysis.

Reaching Definitions: Kill Set

Kill(b): Set of definitions in other basic blocks that are killed in b (i.e., by instructions in b). For each variable v defined in b, the kill set contains all definitions of v in other basic blocks.



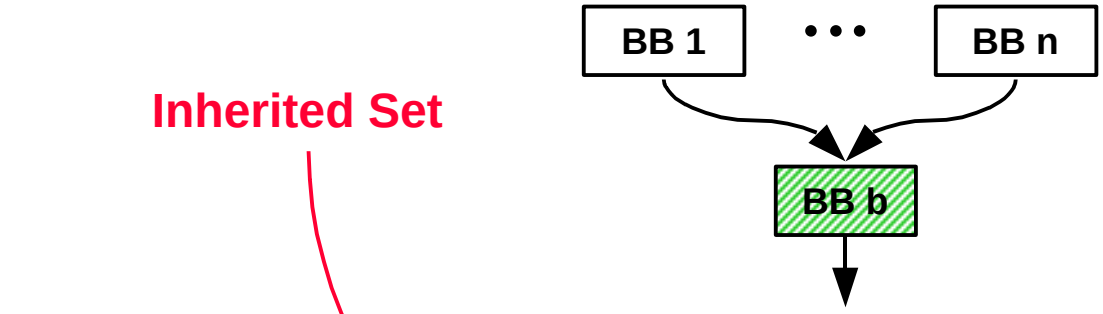
Kill (BB 1) = {d5, d4, d6}

Kill (BB 2) = {d2, d6}

Kill (BB 3) = {d1, d3, d2, d4}

Reaching Definitions: Data Flow Equations

RDin(b) : Set of definitions that reach the beginning of b.
RDout(b) : Set of definitions that reach the end of b.



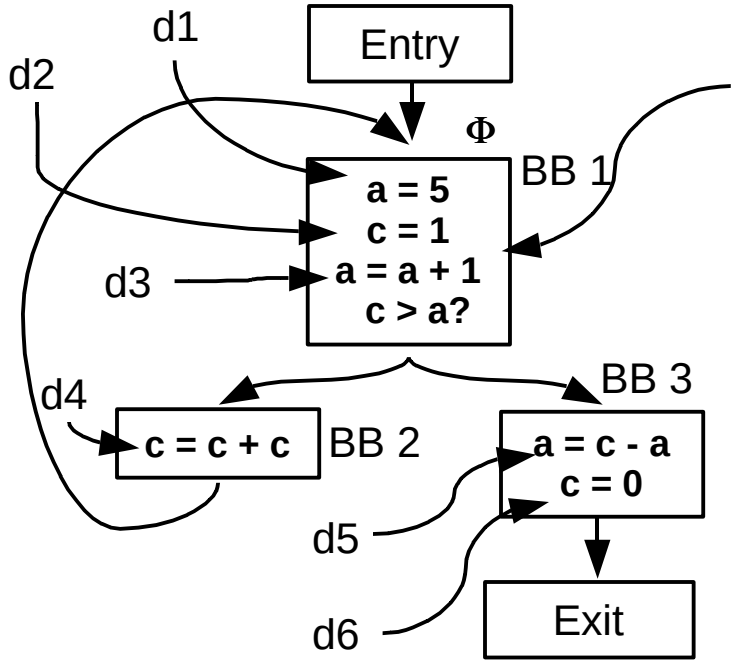
Inherited Set

$$RDin(b) = \bigwedge_{i \in Pred(b)} RDout(i)$$

Synthesized Set

$$RDout(b) = Gen(b) \wedge [RDin(b) - Kill(b)]$$

Reaching Definitions - Solving the Data Flow Equations



RDin(BB 1) = RDout(Entry) ∪ RDout(BB 2) = Φ

RDout(BB 1) = Gen(BB 1) ∪ [RDin(BB 1)-Kill (BB 1)]
= {d2,d3} ∪ [Φ-{d4,d5,d6}] = {d2,d3}

RDin(BB 1) = RDout(Entry) ∪ RDout(BB 2)={d3,d4}
RDout(BB 1) = Gen(BB 1) ∪ [RDin(BB 1)-Kill (BB 1)]
= {d2,d3} ∪ [{d3,d4}-{d4,d5,d6}]={d2,d3}

RDin(BB 3) = RDout(BB 1) = {d2,d3}

RDout(BB 3) = Gen(BB 3) ∪ [RDin(BB 3)-Kill (BB 3)]
= {d5,d6} ∪ [{d2,d3}-{d1,d2,d3,d4}]
= {d5,d6}

RDin(BB 2) = RDout(BB 1) = {d2,d3}

RDout(BB 2) = Gen(BB 2) ∪ [RDin(BB 2)-Kill (BB 2)]
= {d4} ∪ [{d2,d3}-{d2,d6}] = {d3,d4}

Repeat!

no change ⇒ done

Where do we start? Why?

Other data flow problems

- Reaching definitions
- Live variables
- Available expressions
- Very busy expressions