CS293S Redundancy Removal: SVN & DVN

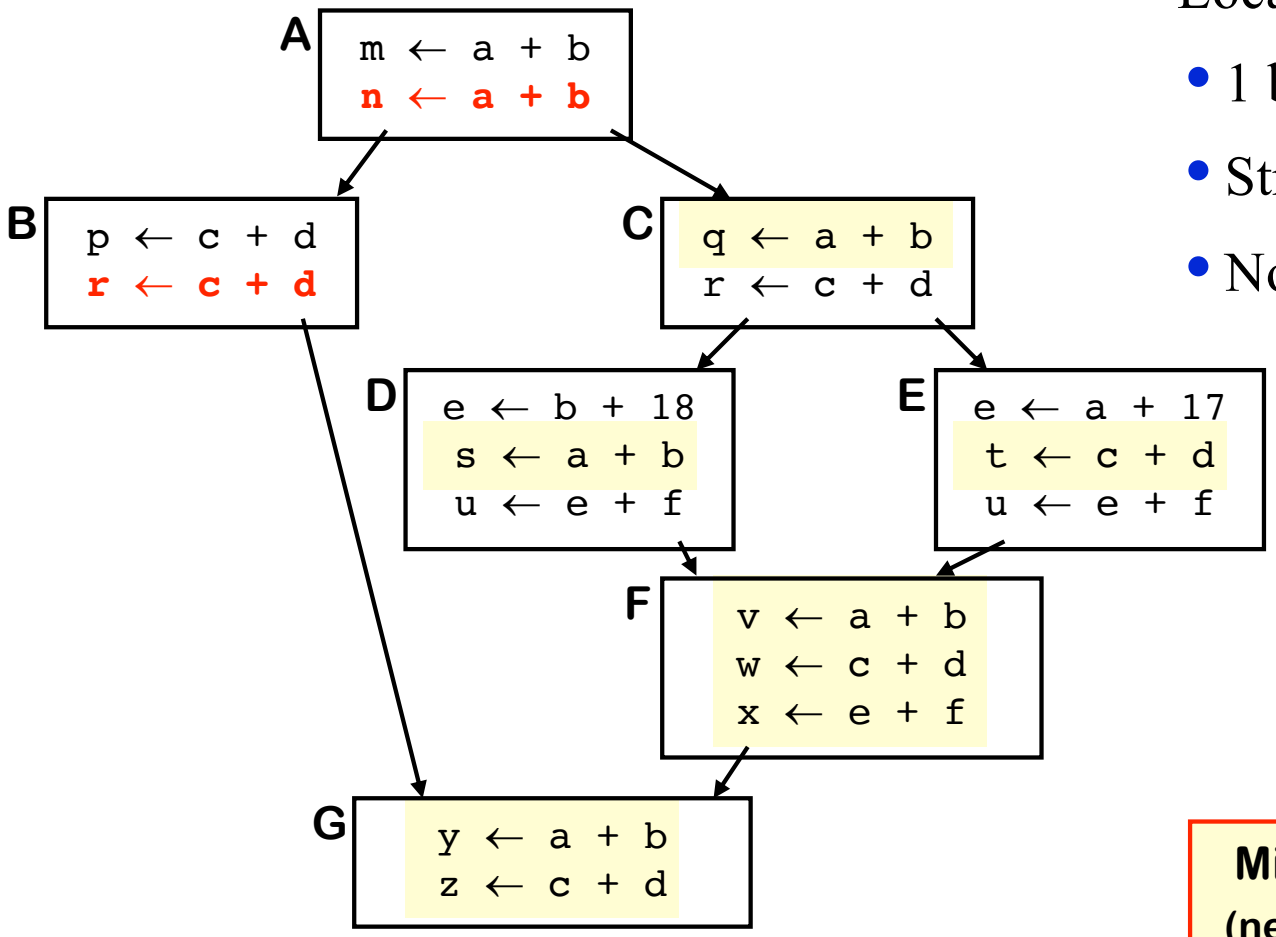
Yufei Ding

Review of Last Class

Redundancy Elimination

- **Goal:** Removing redundant expressions
- **Data Structure** to encode our target program: types of intermediate representations
- **Two methods** for removing redundant expressions
  - DAG: version tracking
  - Linear representation: value numbering

Local Value Numbering <-> Linear IR

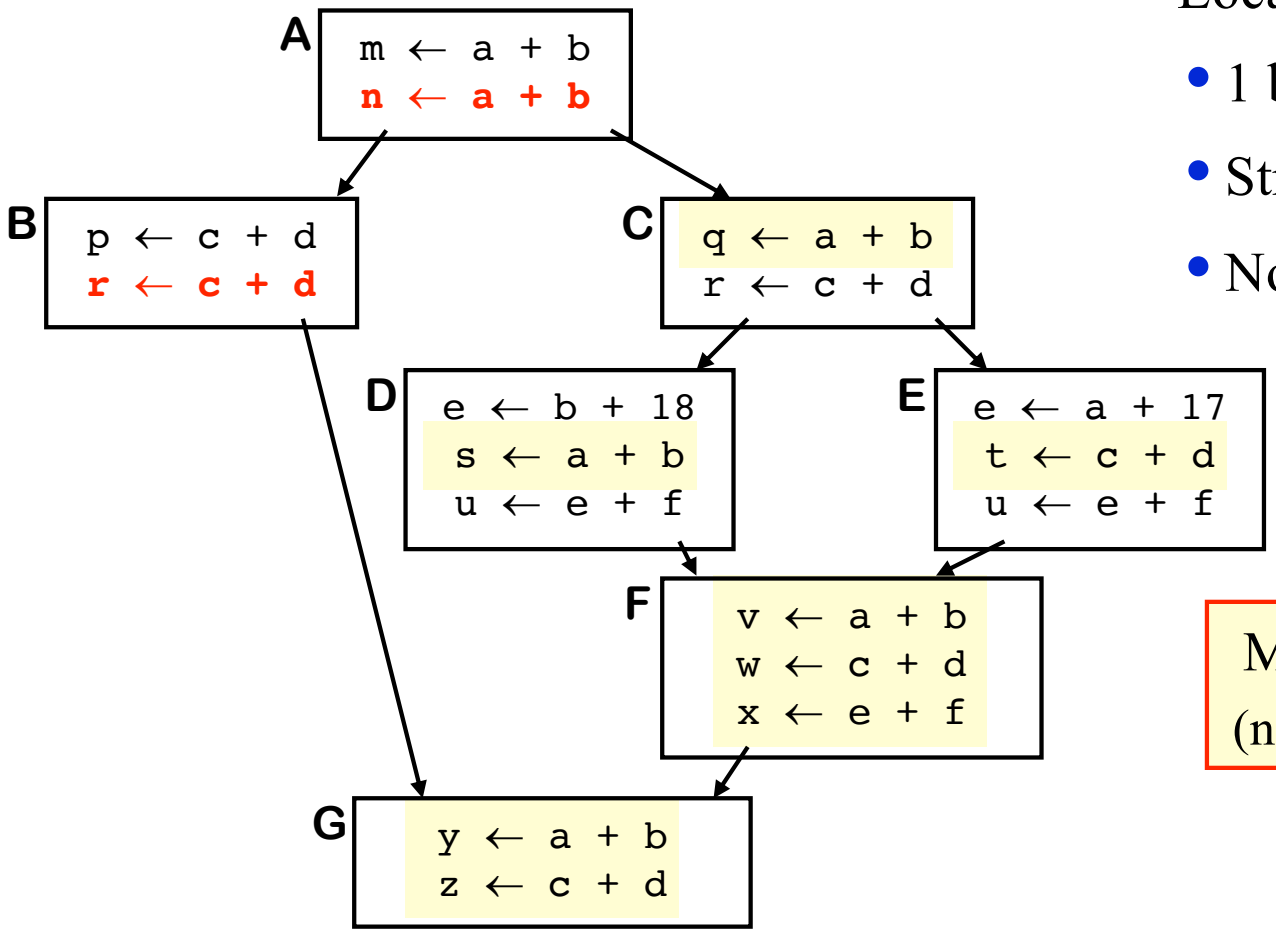


Local Value Numbering

- 1 block at a time
- Strong local results
- No cross-block effects

Missed opportunities
(need stronger methods)

Local Value Numbering <-> Linear IR



Local Value Numbering

- 1 block at a time
- Strong local results
- No cross-block effects

Missed opportunities
(need stronger methods)

Can we find set of blocks that also ensures the sequential execution order in the basic block?

Topics of This Class

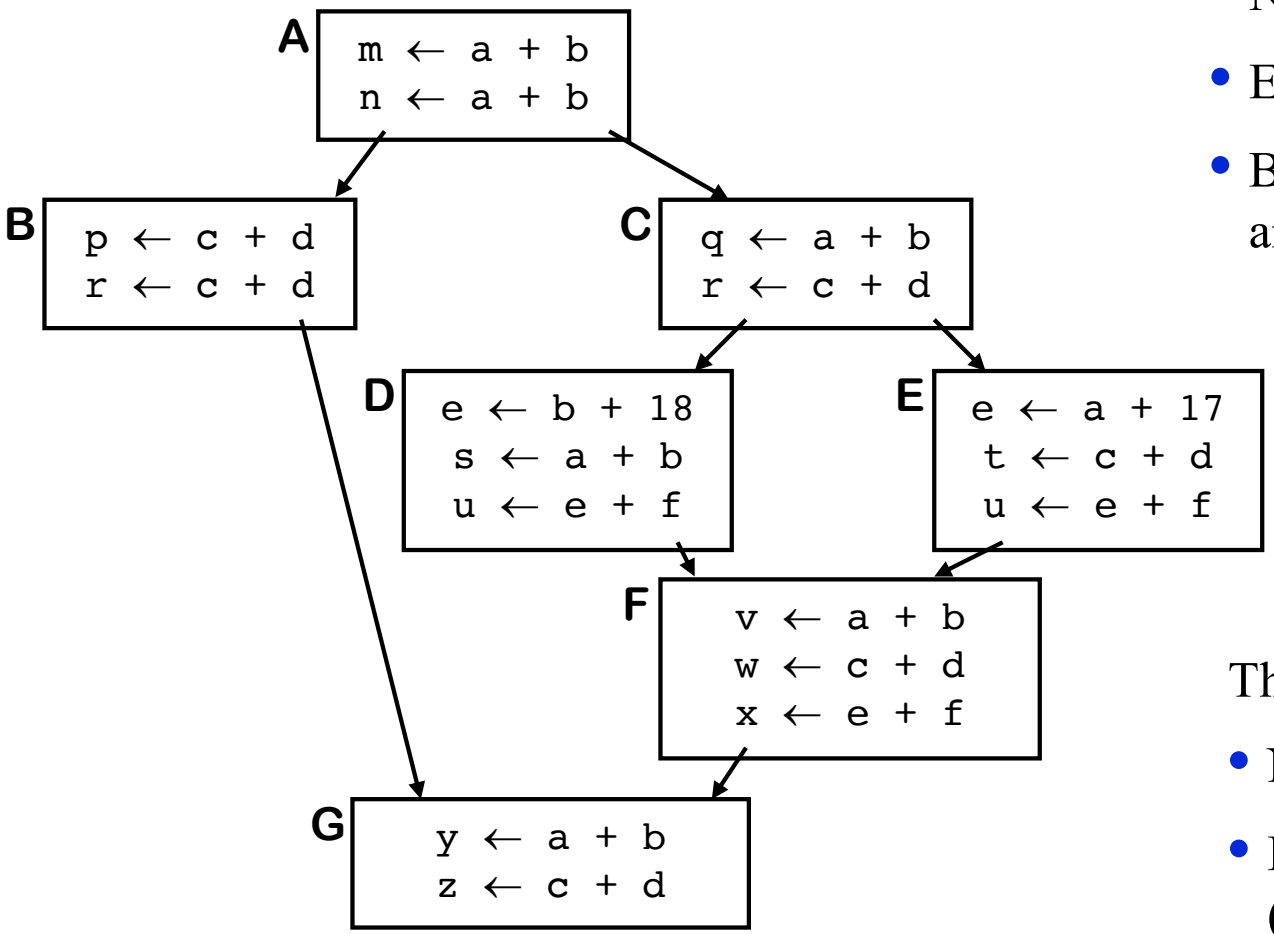
- Scope of optimization
  - Basic block -> Local value numbering
  - Extended basic block -> Superlocal value numbering (SVN)
  - Dominator -> Dominator-based value numbering (DVN)
- Global Common Subexpression Elimination (GCSE)
  - More close to DAG-based methods
  - Work on lexical notation instead of expression values.

Basic blocks

- A basic block is a maximal-length segment of straight-line, unpredicated code. In another word, it has one entry point (i.e., no code within it is the destination of a jump instruction), one exit point and no jump instructions contained within it.
- Example

```
        m = 2;
L2:     c = m + n;
        if(c>0) goto L1;
        d = 4;
        goto L2;
L1:     c = 5;
```

CFG



Control-flow graph (CFG)

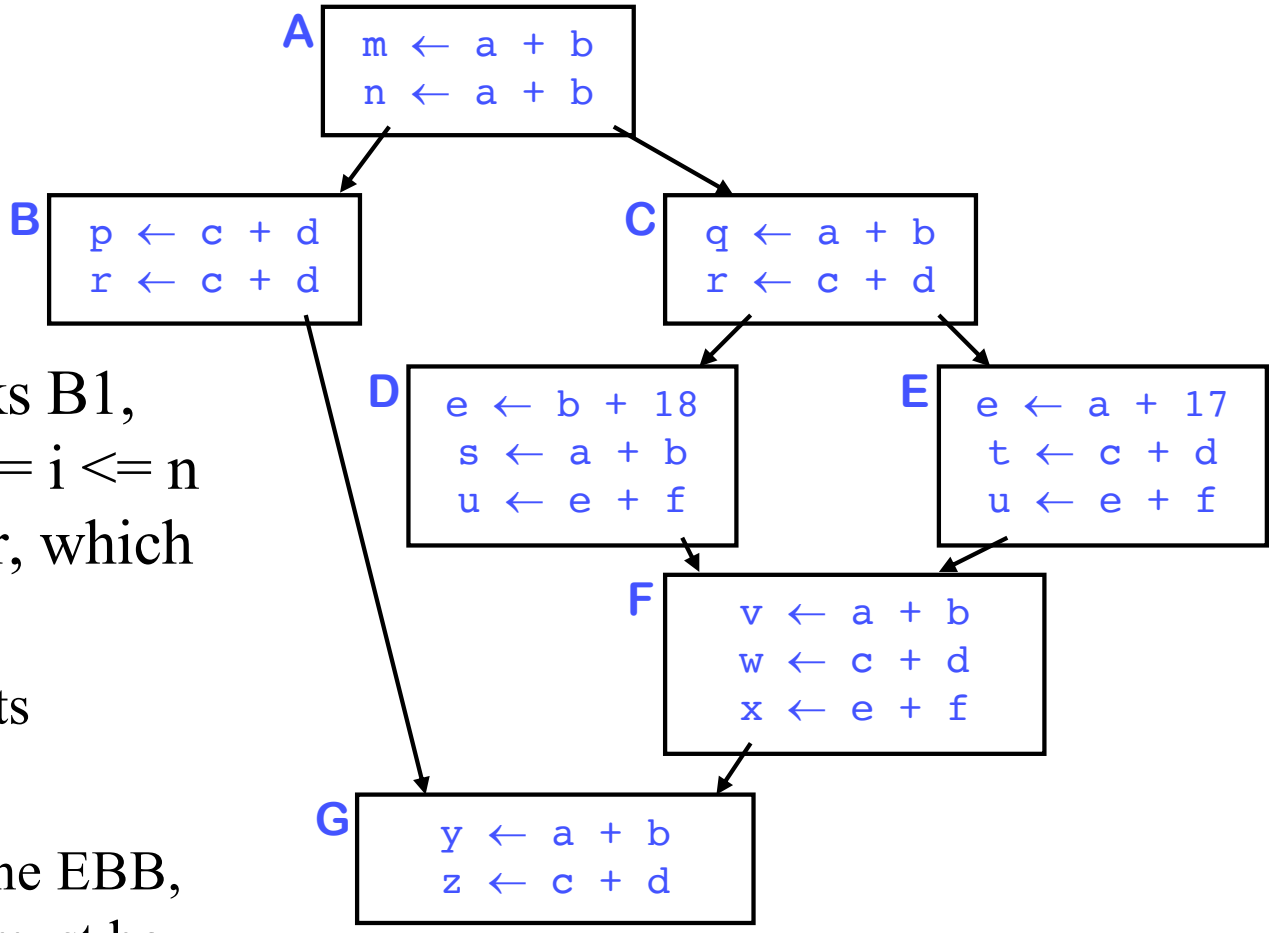
- Nodes for basic blocks
- Edges for branches
- Basis for many program analysis & transformation

This CFG, G = (N,E)

- N = {A,B,C,D,E,F,G}
- E = {(A,B),(A,C),(B,G),(C,D), (C,E),(D,F),(E,F),(F,E)}
- |N| = 7, |E| = 8

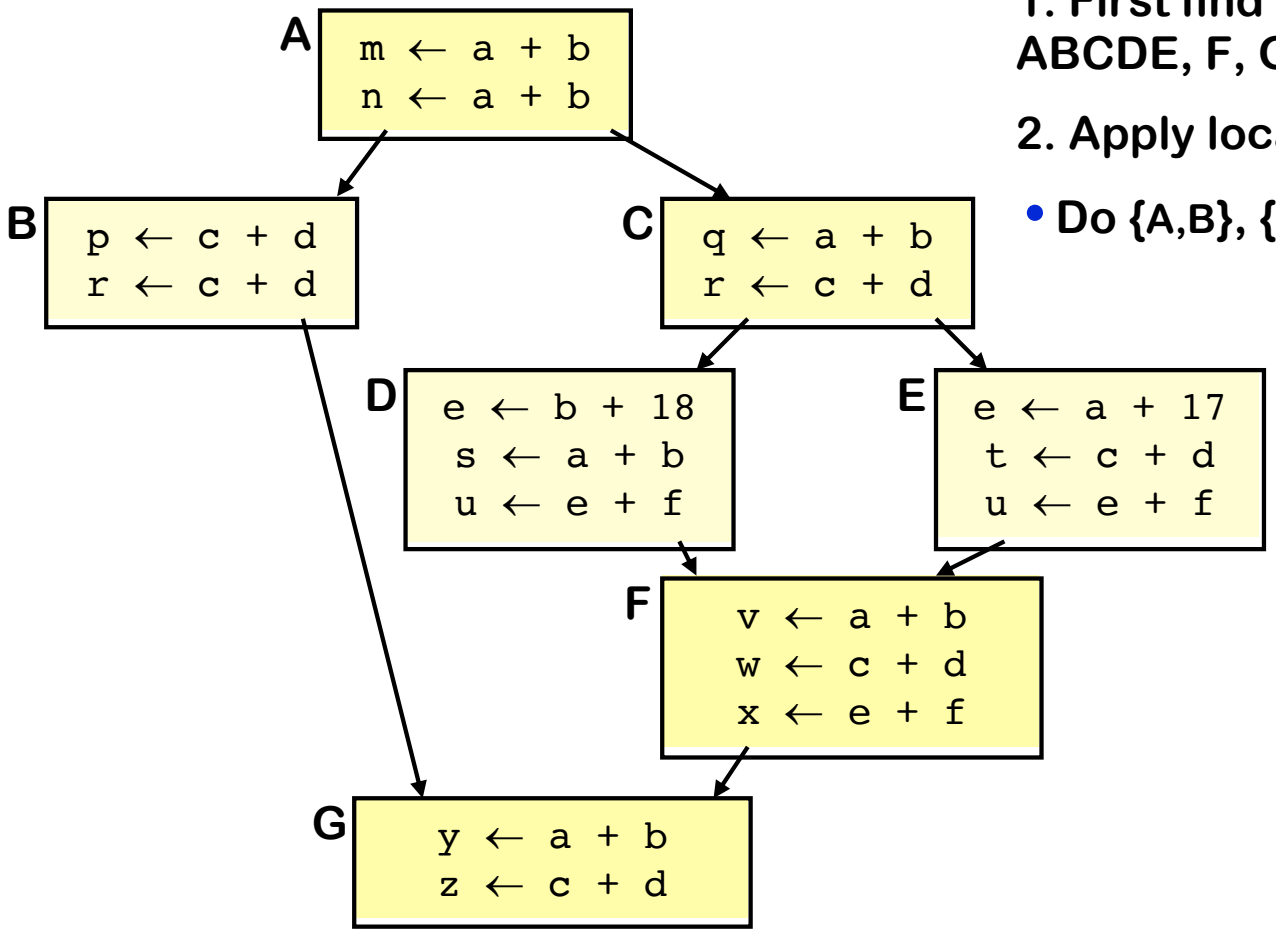
Extended basic block (EBB)

- An EBB is a set of blocks B1, B2, ..., Bn, where Bi, 2<= i <= n has a unique predecessor, which is in the EBB.
  - May have multiple exits
  - A tree structure
  - If a block is added to the EBB, all of its predecessors must be included. Bi is the one with on predecessor, i.e., the root of the EBB.



Can you find the maximum EBB ?

Superlocal Value Numbering



1. First find the maximum EBB: ABCDE, F, G

2. Apply local method to EBBs' paths

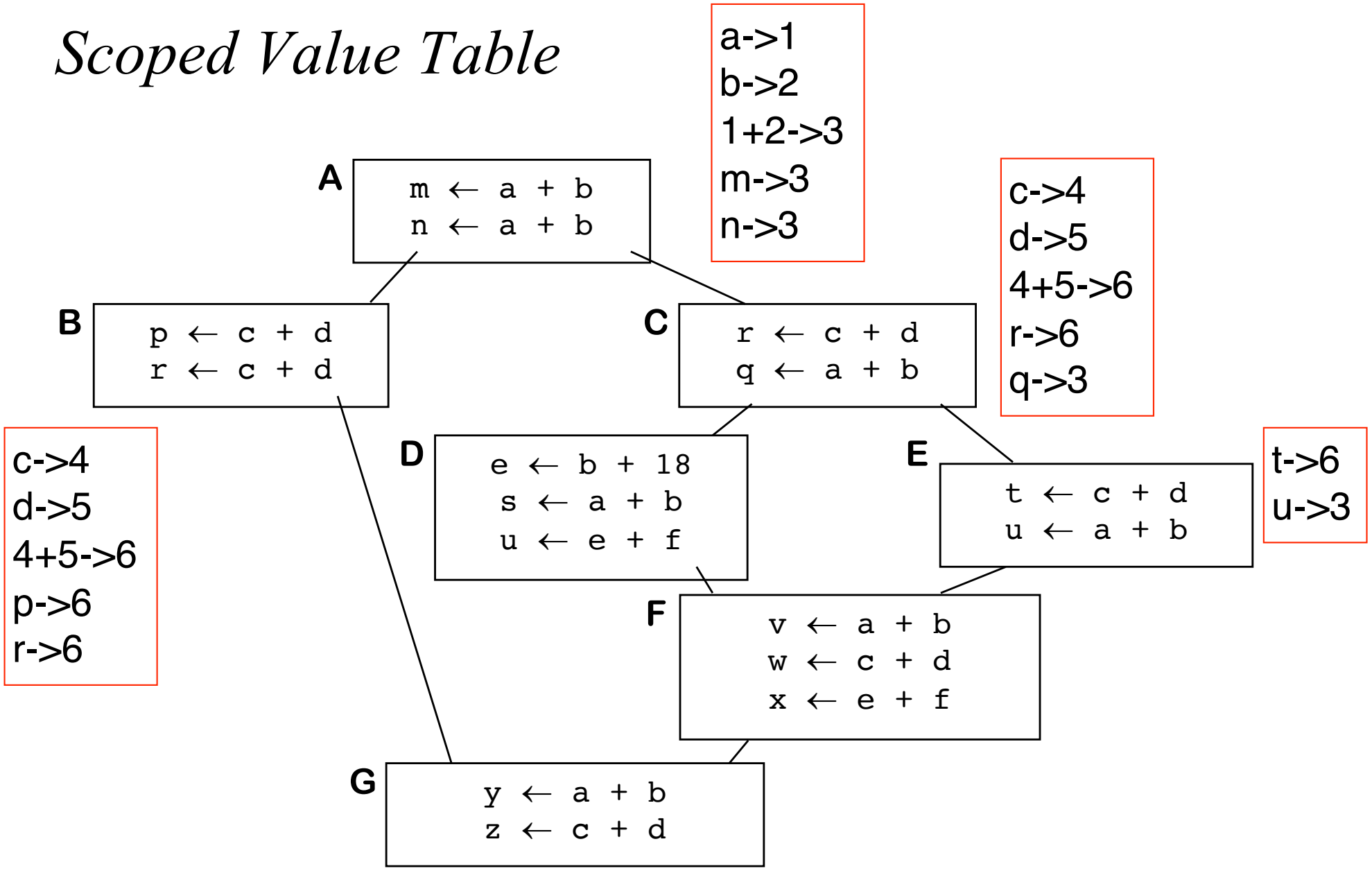
- **Do {A,B}, {A,C,D}, {A,C,E}, {F}, {G}**

Implementation

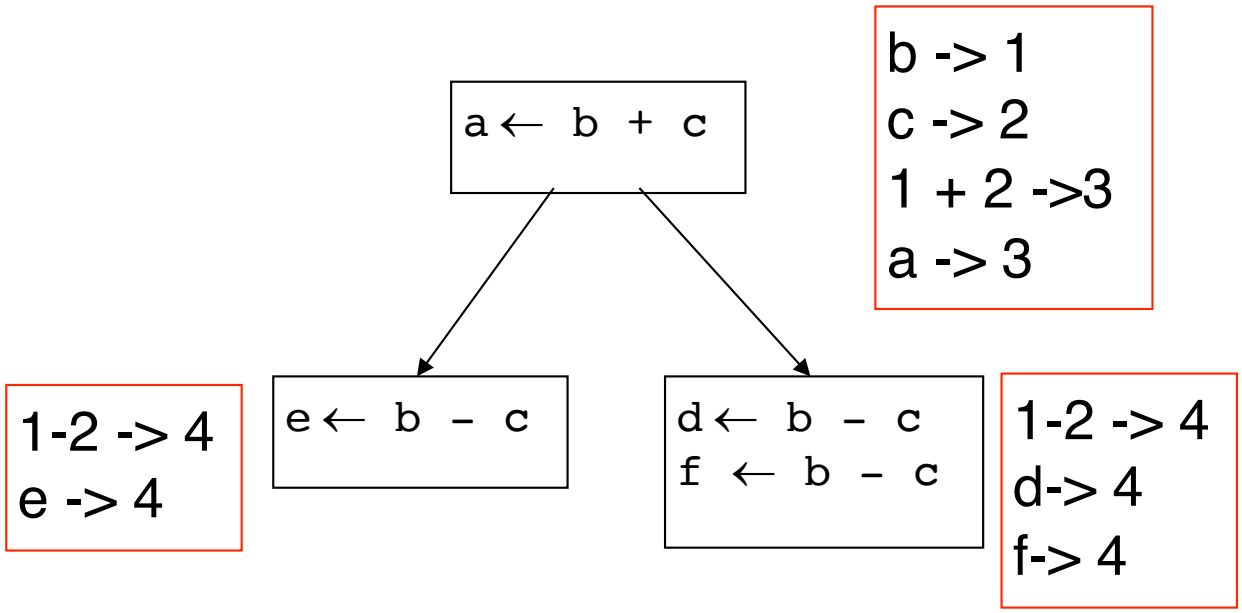
- Reuse the value numbering results of some common blocks for efficiency
- Which necessitates the undoing of a block's effect
  - After {A,C,D}, it must recreate the state of {A,C} before processing E.
  - Options:

1. Record the state of the tables at each block boundary, and restore the state when needed
2. Walking backward and undo the effect. Need record the "lost" information.
3. Scoped hash tables (Lowest cost)
   keep the table produced at the current block

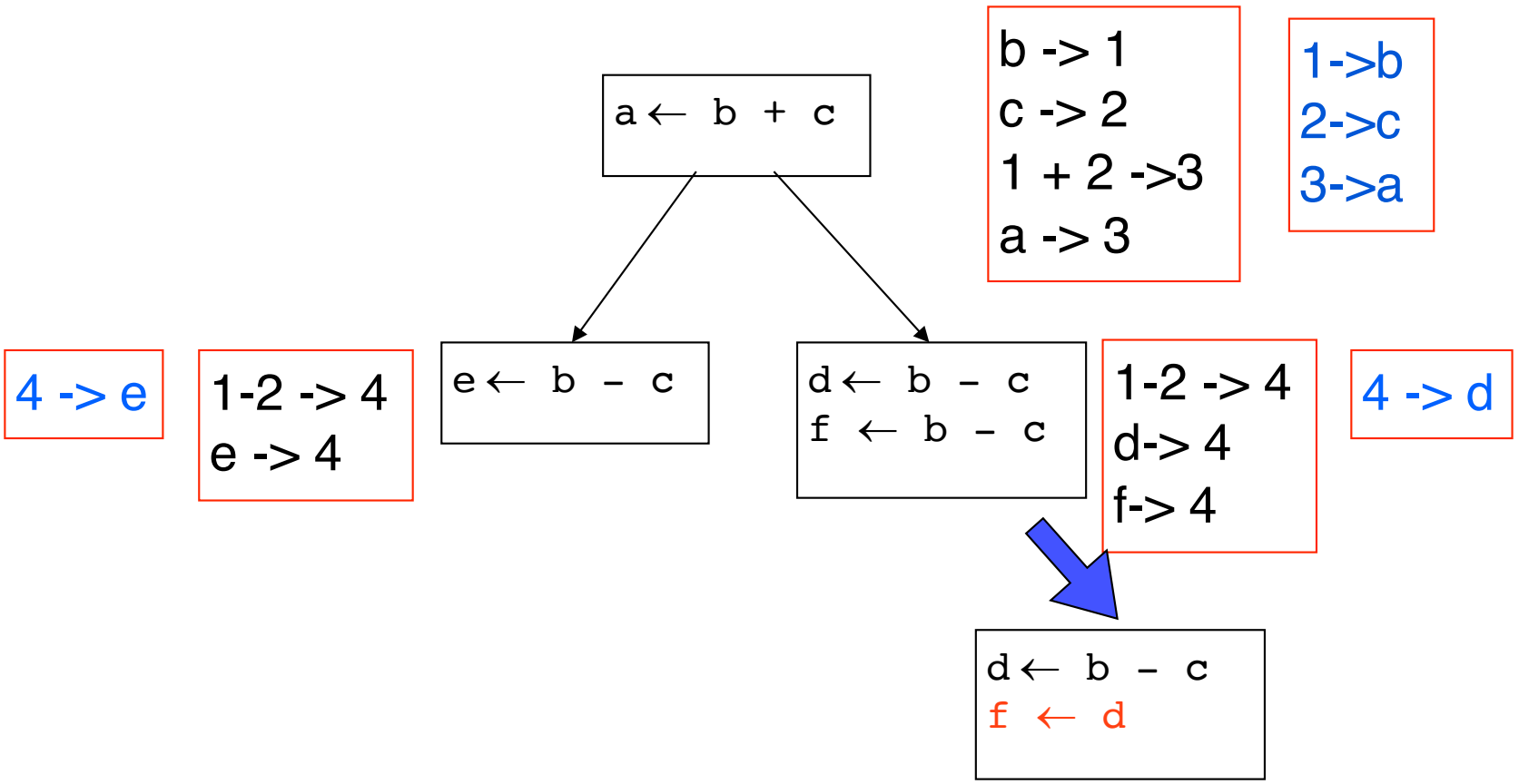
Scoped Value Table



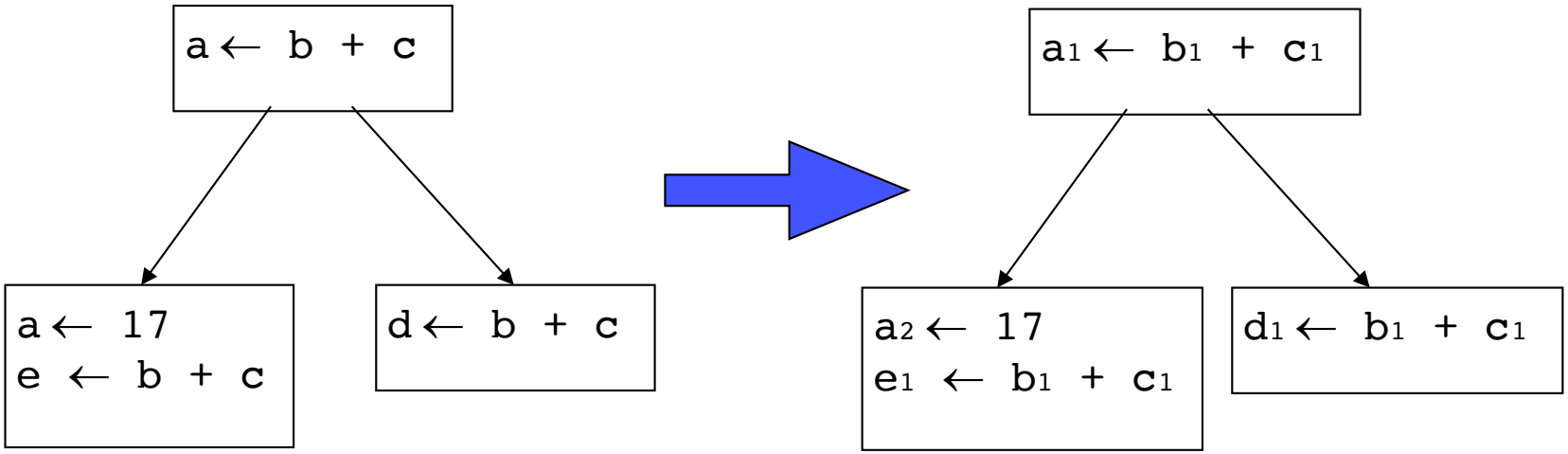
Scoped Value Table



Rewritten

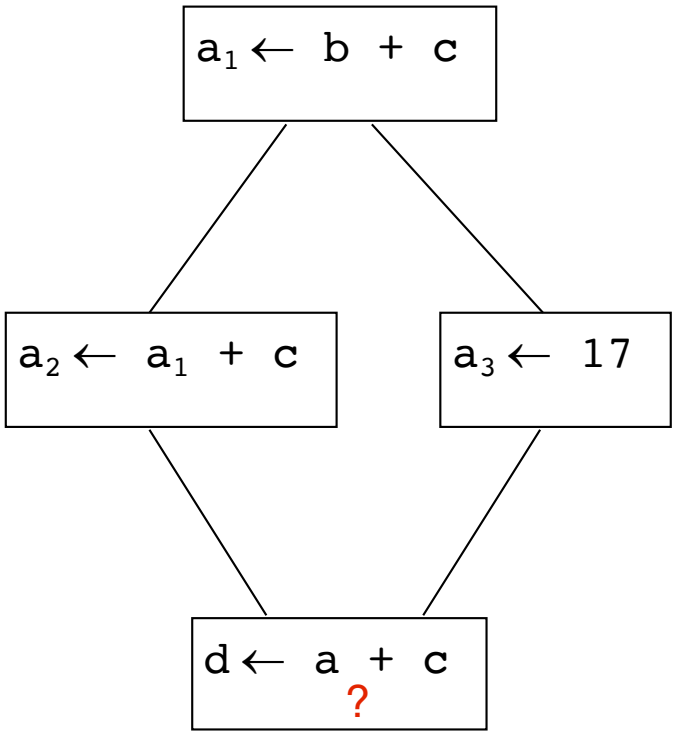


Rewritten



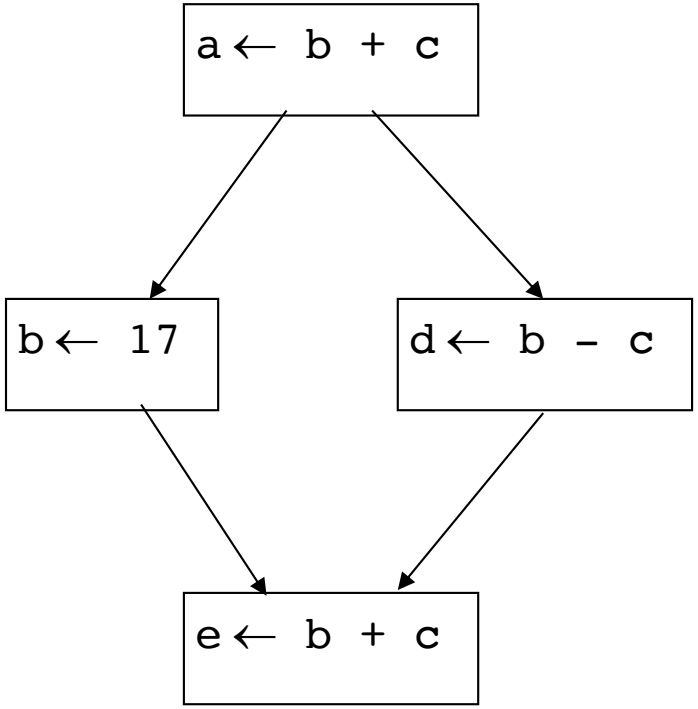
Renaming is still needed. But does it work in all scenarios?

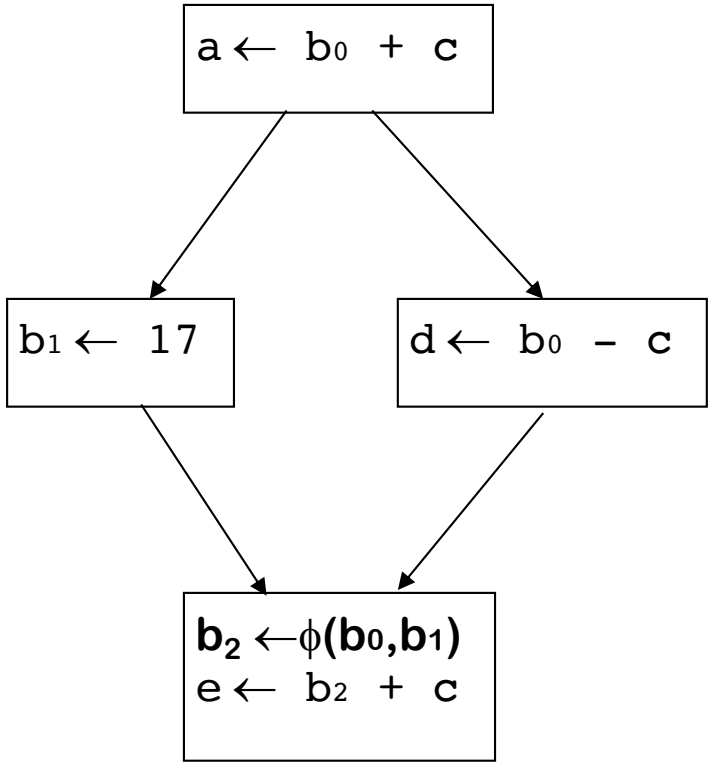
Extra Complexity



Key: SSA Resolves Name Conflicts

SSA Resolves Name Conflicts





SSA (Single Static Assignment) Name Space

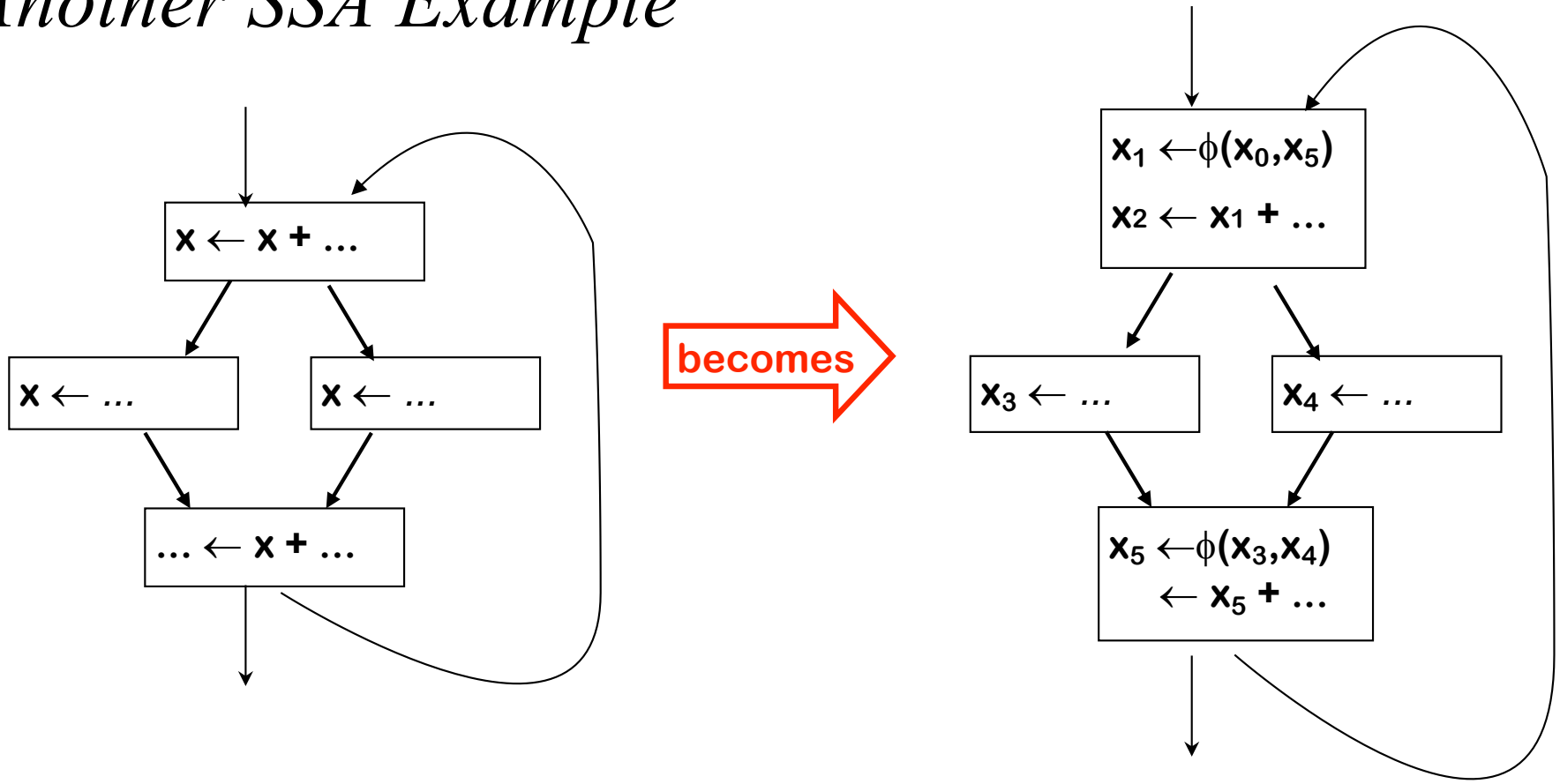
Two principles

- Each name is defined by exactly one operation
- Each operand refers to exactly one definition

To reconcile these principles with real code

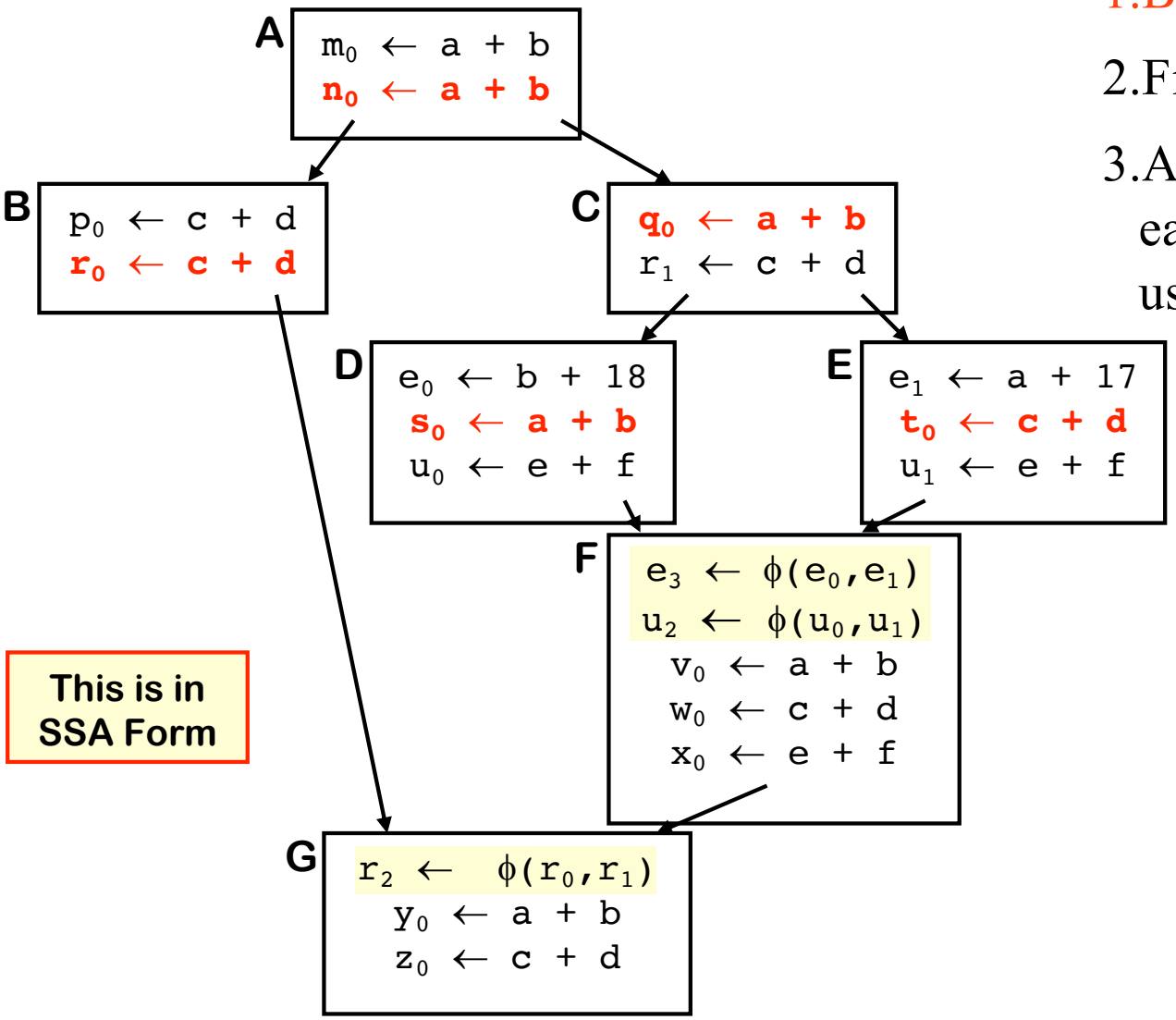
- Insert ϕ-functions at merge points to reconcile name space

Another SSA Example



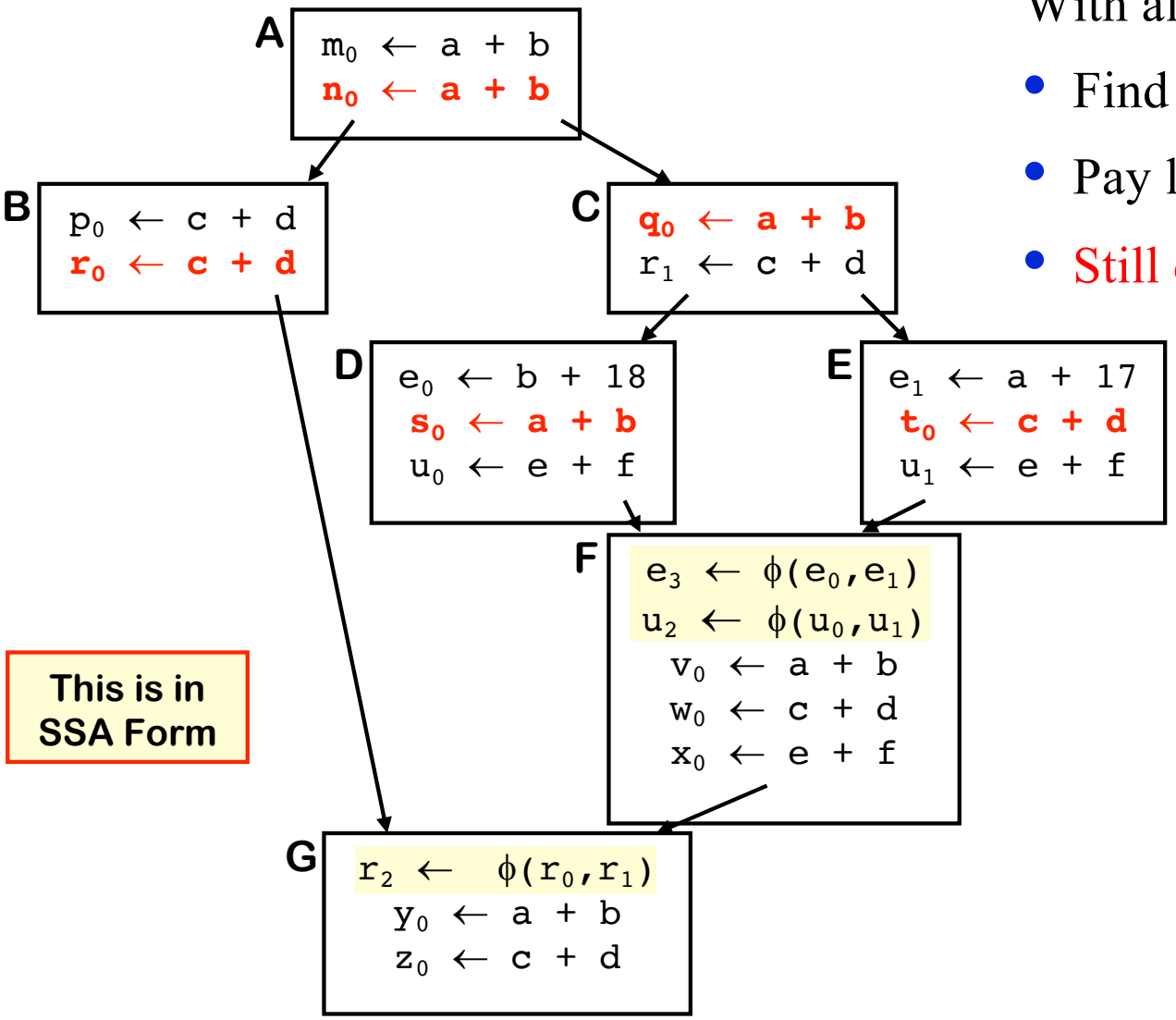
Detail: CT-2ndEd: Section 5.4.2;
CT-1stEd: Section 5.5.

Superlocal Value Numbering



1. Build SSA form
2. Find EBBs
3. Apply value numbering to each path in each EBB using scoped hash tables

Superlocal Value Numbering



With all the bells & whistles

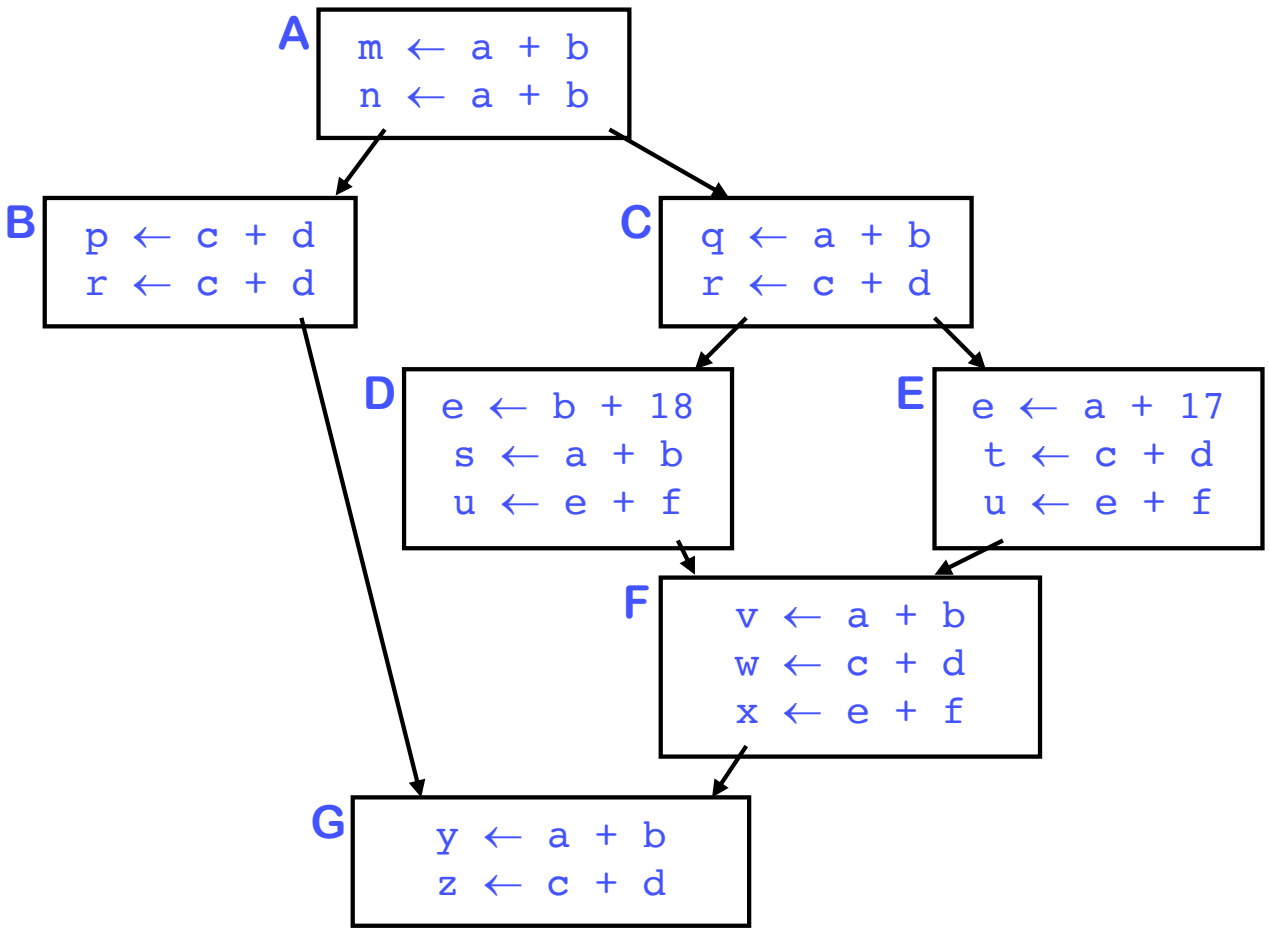
- Find more redundancy
- Pay little additional cost
- Still does nothing for F & G

Dominator-Based Value Numbering

Regional (Dominator-based) Methods

- Dominators of b: all blocks that dominate b
  - if every path from the entry of the graph to b goes through a, then a is one of b's dominator.
  - The full set of dominators for b is denoted by DOM(b).
- Strict Dominators:
  - If a dominators b and $a \neq b$, then we say a strictly dominates b.
- Immediate Dominator:
  - The immediate dominator of b is the strict dominator of b that is closest to b. It is denoted IDOM(b).

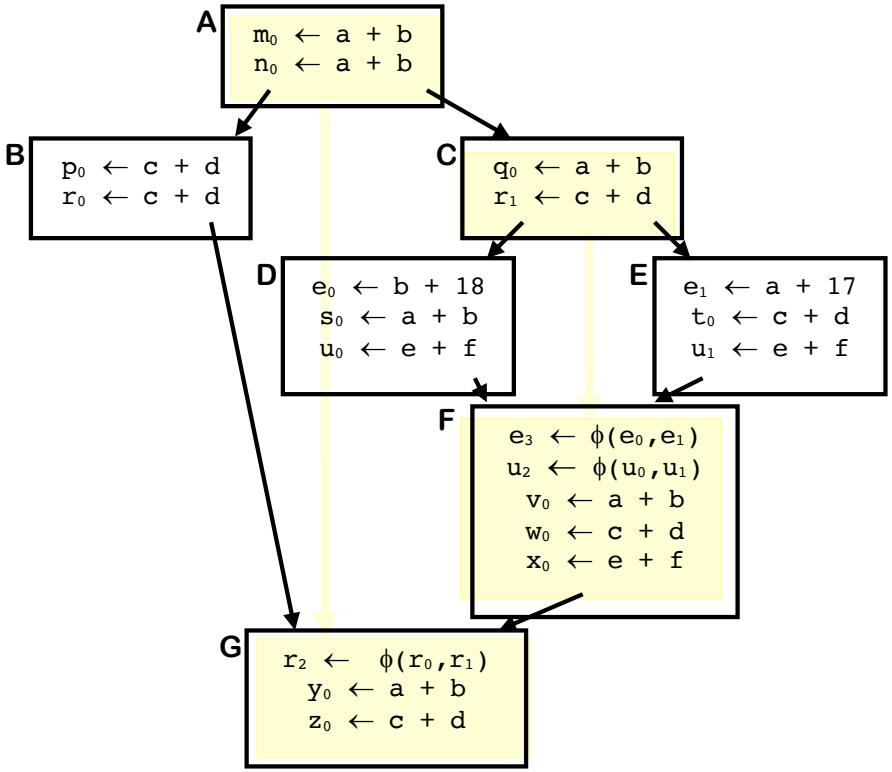
Example



BLOCK	A	B	C	D	E	F	G
DOM							
IDOM							

Dominator-Based Value Numbering

- Basic strategy: use table from IDom(x) to start value numbering x
  - Use C for F and A for G
  - Imposes a Dom-based application order



Summary

- Two methods in a scope beyond a basic block
  - Superlocal value numbering (SVN)
    - Value numbering across basic blocks
  - Dominator-based value numbering (DVN)
    - Uses dominance information to handle join points in CFG
- They can be used together
  - First Build SSA
  - Do SVN
  - Do DVN with the value tables built in SVN reused

Build SSA form is the prerequisite for both!

Examples with redundancy can not be eliminated?