



Program Analysis
<https://www.cse.iitb.ac.in/~karkare/cs618/>

Static Single Assignment (SSA) (continued)

Amey Karkare
 Dept of Computer Science and Engg
 IIT Kanpur
 Visiting IIT Bombay
karkare@cse.iitk.ac.in
karkare@cse.iitb.ac.in

Complexity of Construction

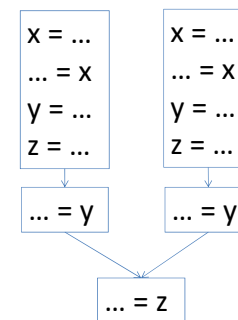
- $R = \max(N, E, A, M)$
- N: nodes, E: edges in flow graph
- A: number of assignments
- M: number of use of variables
- Computation of DF: $O(R^2)$
- Computation of SSA: $O(R^3)$
- In practice, worst case is rare.
- Practical complexity: $O(R)$

2

Linear Time Algo for ϕ -functions

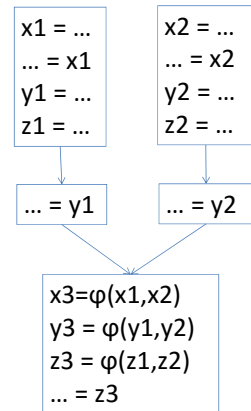
- By Sreedhar and Gao, in POPL'95
- Uses a new data structure called DJ-graph
- Linear time is achieved by careful ordering of nodes in the DJ-graph
 - DF for a node is computed only once and reused later if required.

3



Original Program

4



Minimal SSA form

5

Variants of SSA Form

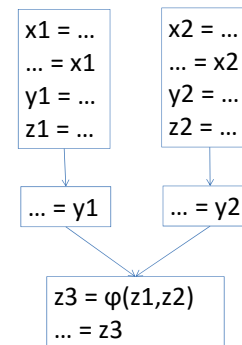
- Minimal SSA still contains extraneous ϕ -functions
 - Inserts some ϕ -functions where they are dead
 - Would like to avoid inserting them
- Pruned SSA
- Semi-Pruned SSA

6

Pruned SSA

- Only insert ϕ -functions where their value is live
 - Inserts fewer ϕ -functions
 - Costs more to do
 - Requires global Live variable analysis

7



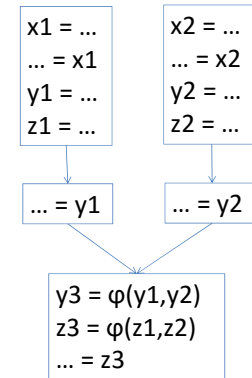
Pruned SSA form

8

Semi-pruned SSA

- *Semi-pruned SSA*: discard names used in only one block
 - Total number of ϕ -functions between minimal and pruned SSA
 - Needs only local Live information
 - Non-locals can be computed without iteration or elimination

9



Semi-pruned SSA form

10

Computing Non Locals

```

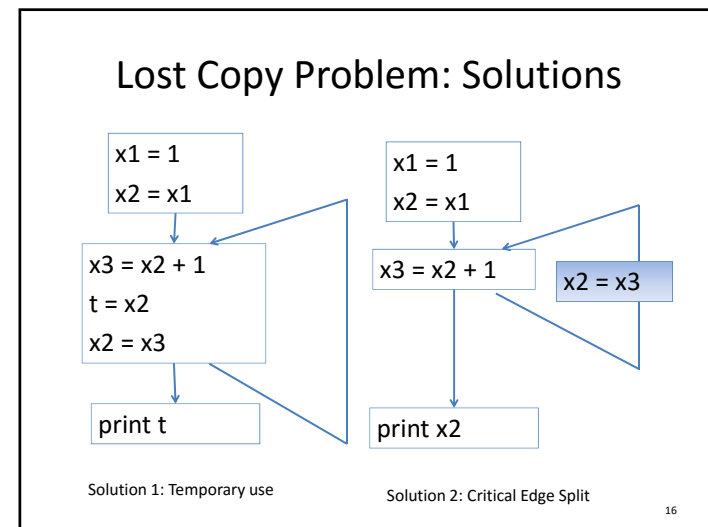
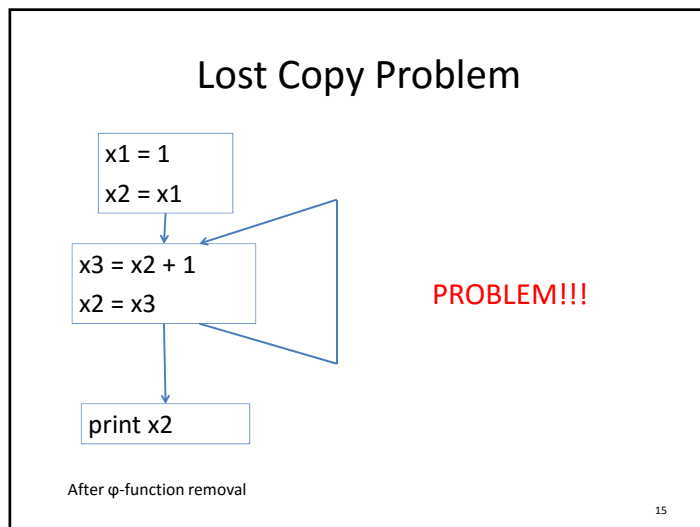
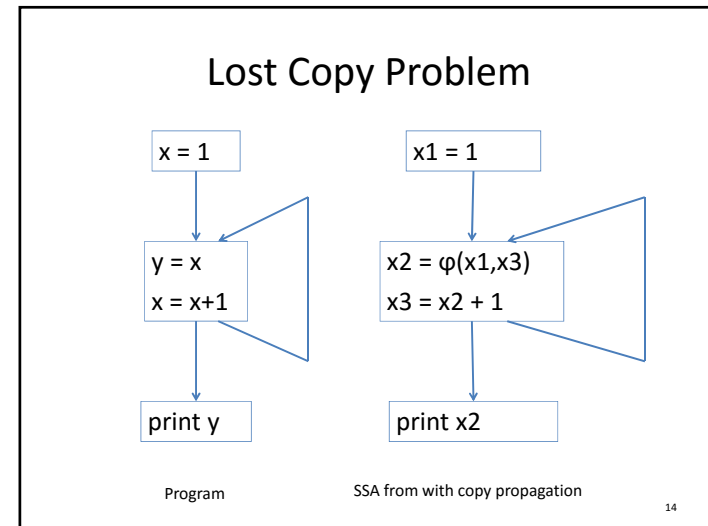
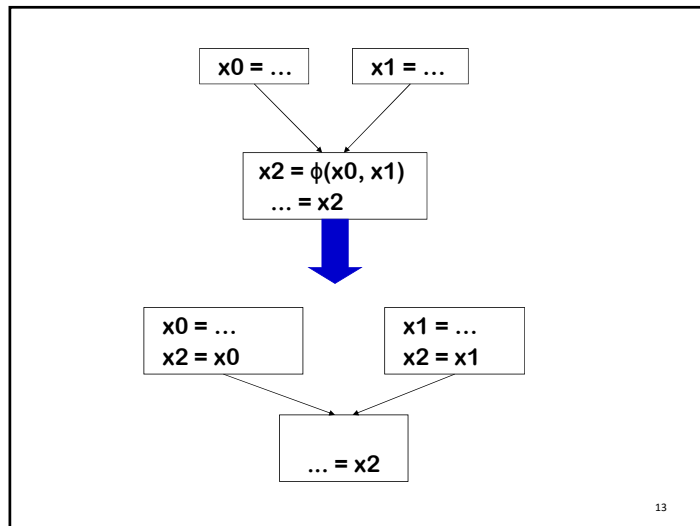
for each block B {
  defined = {}
  for each instruction v = x op y {
    if x not in defined
      non_locals = non_locals ∪ {x}
    if y not in defined
      non_locals = non_locals ∪ {y}
    defined = defined ∪ {v}
  }
}

```

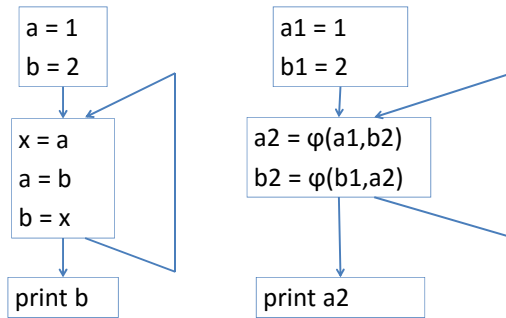
11

SSA to Executable

- At some point, we need executable code
 - Need to fix up the ϕ -function
- Basic idea
 - Insert copies in predecessors to mimic ϕ -function
 - Simple algorithm
 - Works in most cases, but **not always**
 - Adds lots of copies
 - Many of them will be optimized by later passes



Swap Problem

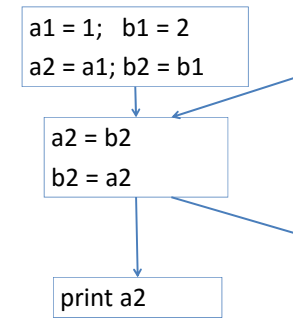


Program

SSA from with copy propagation

17

Swap Problem

**PROBLEM!!!**

Fix requires compiler to detect and break dependency from output of one ϕ -function to input of another ϕ -function. May require temporary if cyclic dependency exists.

After ϕ -function removal

18

SSA Form for Optimizations

- SSA form can improve and/or speed up many analyses and optimizations
 - (Conditional) Constant propagation
 - Dead code elimination
 - Value numbering
 - PRE
 - Loop Invariant Code Motion
 - Strength Reduction

19