

CSC2/455 Software Analysis and Improvement

Dominators and SSA Form

Sreepathi Pai

URCS

February 6, 2019

Outline

Review

Dominator Analysis (DOM)

SSA Form

Postscript

Outline

Review

Dominator Analysis (DOM)

SSA Form

Postscript

Data flow analysis framework

- ▶ Live variable analysis
 - ▶ “Is there a read of this variable along any path?”
- ▶ Reaching Definitions
 - ▶ “Which definitions reach this use?”
- ▶ Available expressions
 - ▶ “Is this expression calculated previously and the results still usable?”
- ▶ Very Busy Expressions
 - ▶ “Are there expressions that can be precalculated?”
- ▶ Iterative data flow analysis
 - ▶ GEN, KILL, Transfer functions, Initialization

Outline

Review

Dominator Analysis (DOM)

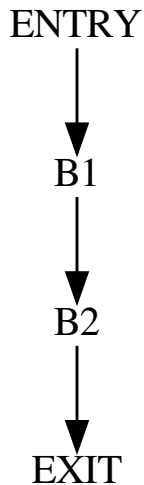
SSA Form

Postscript

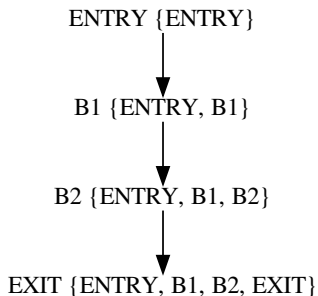
Dominators

- ▶ A node n in the CFG dominates a node m iff:
 - ▶ n is on all paths from entry to m
 - ▶ by definition, a node n always dominates itself
- ▶ Note this is a property of the structure of the CFG
 - ▶ I.e. has nothing to do with code in basic blocks
- ▶ Can we use data flow analysis to identify the dominators of a node?

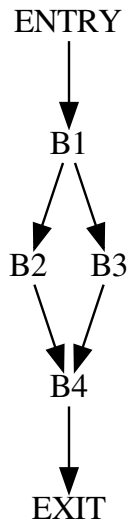
Example 1: Node with single predecessor



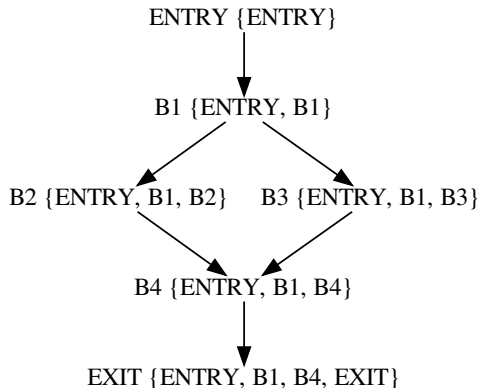
Example 1: Node with single predecessor (Answer)



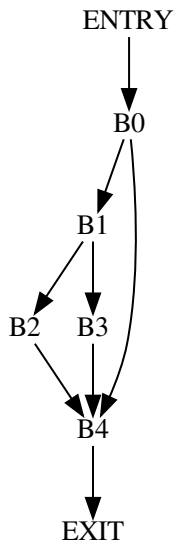
Example 2: Node with multiple predecessors



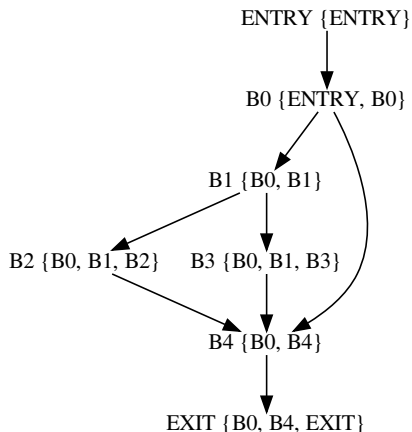
Example 2: Node with multiple predecessors (Answer)



Example 3: Slightly more involved example



Example 3: Slightly more involved example (Answer)



Data flow analysis Equation

$$\text{DOM}(n) = \{n\} \cup \left(\bigcap_{m \in \text{pred}(n)} \text{DOM}(m)\right)$$

► Initialization

- (for $n \neq \text{ENTRY}$): $\text{DOM}(n) = N$ (where N is the set of all nodes)
- (for $n = \text{ENTRY}$): $\text{DOM}(n) = \text{ENTRY}$

Outline

Review

Dominator Analysis (DOM)

SSA Form

Postscript

Static Single Assignment (SSA) Form

- ▶ Intermediate Representation
 - ▶ Similar to 3 address code
- ▶ Each variable only written once
 - ▶ Static [in source] Single [once] assignment
- ▶ SSA form can be generated from 3 address code
 - ▶ Introduce ϕ functions
 - ▶ Rename variables

Example 1: Straight-line code

```
y = x + 1;  
x = 2;  
y = x + y + 2;
```

gets transformed to:

```
y_0 = x_0 + 1  
x_1 = 2;  
y_1 = x_1 + y_0 + 2;
```

From this example, when should we rename variables?

Example 2: Branches

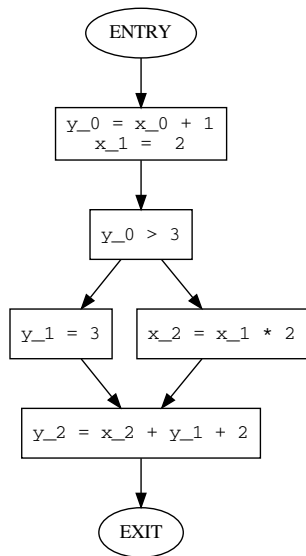
```
y = x + 1;  
x = 2;  
  
if(y > 3)  
    y = 3;  
else  
    x = x * 2;  
  
y = x + y + 2;
```

gets transformed to:

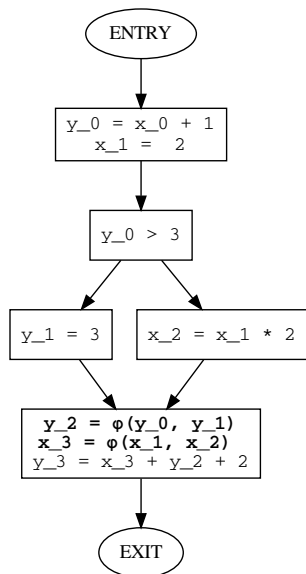
```
y_0 = x_0 + 1  
x_1 = 2;  
  
if(y_0 > 3)  
    y_1 = 3;  
else  
    x_2 = x_1 * 2;  
  
y_2 = x_2 + y_1 + 2;
```

Is this renaming correct?

Example 2: The CFG



Example 2: Fix using ϕ functions



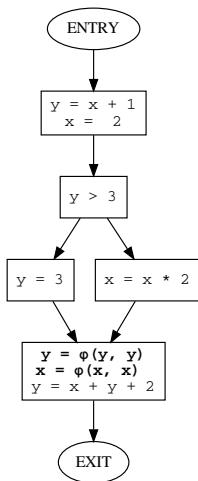
Simple Algorithm for constructing SSA form: 1

- ▶ Insert ϕ functions
 - ▶ In which nodes of CFG?
 - ▶ For which variables?
- ▶ Rename variables
 - ▶ To what?
 - ▶ Helps to think of LHS (definition) renames and RHS (use) renames

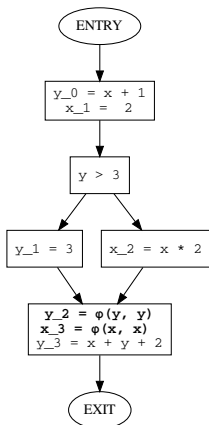
Simple Algorithm for constructing SSA form: 2

- ▶ Insert ϕ functions
 - ▶ In join nodes, before all other code
 - ▶ For *all* variables defined or used in procedure
 - ▶ Each ϕ function has one argument per incoming edge
 - ▶ Use $y = \phi(y, y)$ form for variable y
- ▶ Rename variables
 - ▶ To what?
 - ▶ Helps to think of LHS (definition) renames and RHS (use) renames

Simple Algorithm for constructing SSA form: 3



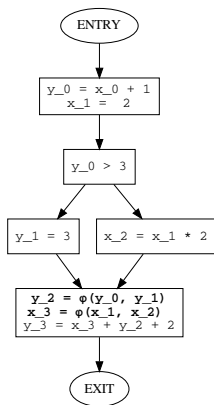
Simple Algorithm for constructing SSA form: Rename LHS



Simple Algorithm for constructing SSA form: Rename RHS

- ▶ Note that in SSA form, only one definition reaches a use (except the uses in ϕ)
- ▶ The arguments to ϕ are the definitions that reach it

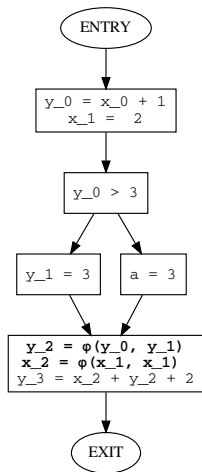
Simple Algorithm for constructing SSA form: Rename RHS



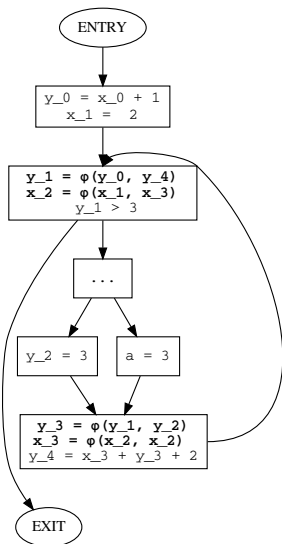
Simple Algorithm for constructing SSA form: Renaming

- ▶ In actual compilers, renaming LHS and RHS can be done by simply calculating reaching definitions
 - ▶ Remember we had to track each definition there too (recall $y \neq 0$)
- ▶ This construction is called the *maximal SSA form*
 - ▶ Simple to construct
 - ▶ Wasteful, can introduce too many ϕ functions (not in our example)

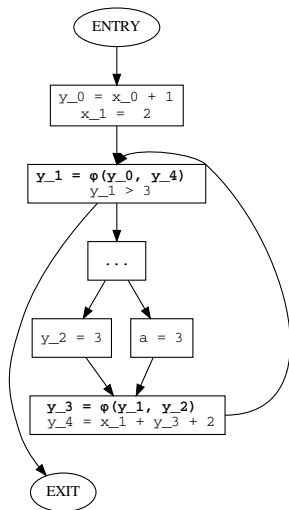
Example: Redundant ϕ functions



Example: Redundant ϕ functions (now with loops)



Example: Non-redundant ϕ functions



Outline

Review

Dominator Analysis (DOM)

SSA Form

Postscript

References

- ▶ Chapter 9 of Cooper and Turczon
 - ▶ Section 9.2.1
 - ▶ Section 9.3