

Induction Variables & Strength Reduction

Induction variable: *variable whose successive values form an arithmetic progression in loop*

Ex. `do i = 1,99`
 `a(i) := 2*i-1`
 `enddo`
i is inductive
*2*i-1 is inductive*

→

`t := -1`
`do i = 1,99`
 `t := t+2`
 `a(i) := t`
 `enddo`

Strength Reduction

*Replaces expensive ops (like *) with less expensive ops (like +)*

1

Loop Peeling

Remove 1 or more iterations from beginning/end of loop

Ex. `do i = 1,100`
 `a(i) := a(i) + a(1)`
 `enddo`
i > 1

→

`a(1) := a(1) + a(1)`
`do i = 2,100`
 `a(i) := a(i) + a(1)`
 `enddo`

Beneficial?

Always possible?

2

Index Set Splitting

Split index set 1,...,N into multiple sets

```
Ex. do i = 1,100
    a(i) := b(i) + c(i)
    if (i>50) then
        d(i) := a(i) + a(i-10)
    enddo

do i = 1,50
    a(i) := b(i) + c(i)
enddo
do i = ,51,100
    a(i) := b(i) + c(i)
    d(i) := a(i) + a(i-10)
enddo
```

Beneficial?

Always possible?

Global Value Numbering with SSA

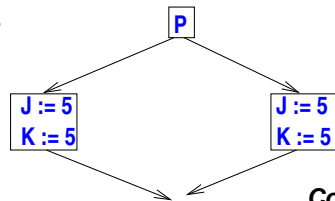
X, Y are *dynamically equivalent* at P if they have the same values whenever control reaches P on execution

Undecidable \longrightarrow Develop static notion *Congruence*

X congruent to Y \longrightarrow X dynamically equivalent to Y

Go beyond Basic Block

Ex.

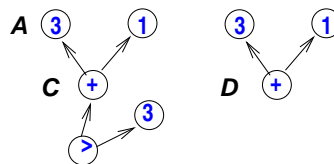


Conclude J,K congruent

4

Value Graph for Basic Block

A := 3
B := 3
C := A+1
D := B+1
if (C>3)...



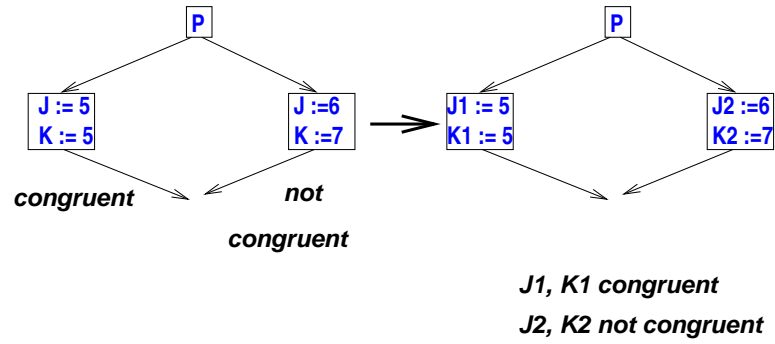
C and D are *congruent*:

have identical operators, and like operands are congruent

Like value-numbering

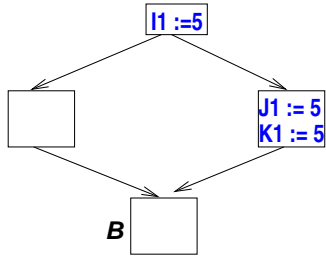
5

Why SSA?



6

What about control flow?



I1, J1, K1 congruent at B if assignments dominate B

7

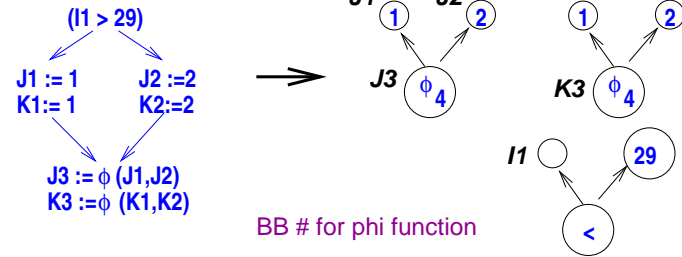
Value Graph for SSA

Nodes Constants, operators, phi-functions

Directed Edges From use to node where value generated

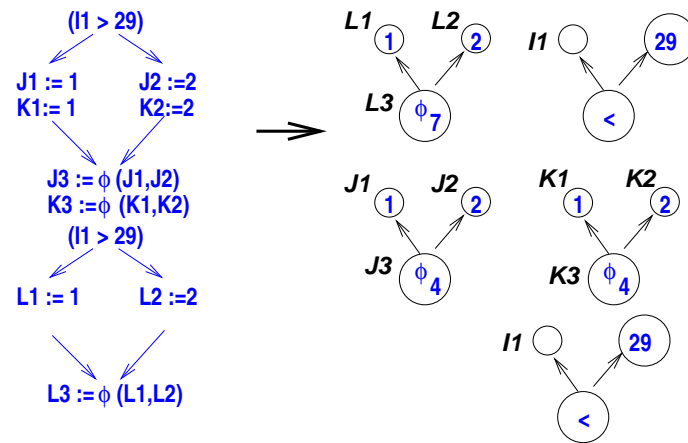
Labels Constant, operator, function symbols

Ex.



8

Value Graph for SSA (example)



9

Congruence

A is **congruent** to node B if

1. A is the same node as B, or
2. A and B are constant nodes, with the same constant value, or
3. A and B are operator nodes, with the same operator, and their like operands are congruent

Vars X and Y are **equivalent** at P if their nodes are congruent and defining assignments dominate P.

Ex.

J1, K1, L1 congruent

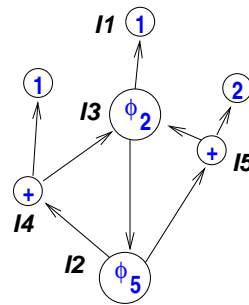
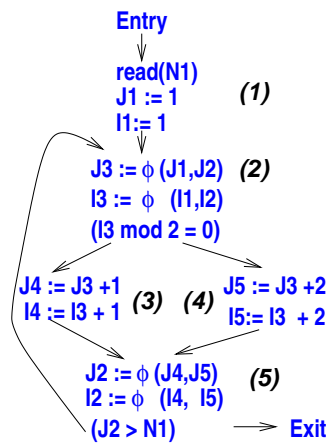
J2, K2, L2 congruent

J3, K3 congruent, but not with L3

Get equivalence classes of variables

10

Loop Example



*Value graph for J2 is identical
BUT cycles prohibit finding it!*

11

Algorithm Overview

1. Compute SSA.
2. Build value graph for SSA.
3. Optimistically assume all nodes with same label are congruent. Determine congruence of nodes by partitioning algorithm.
3. Check for equivalence.

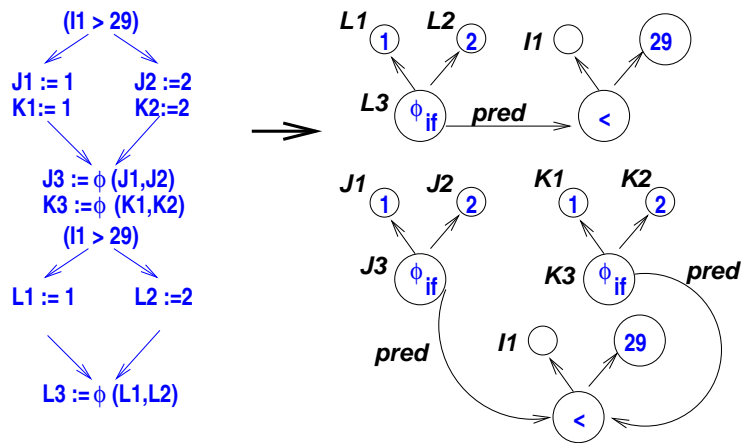
Partitioning: $(O(E \log E))$

1. Put all nodes with same label in same partition.

$i+1$: Two nodes are in same partition at step $i+1$, if at step i , they are in the same partition and the destination of their edges are in the same partition.

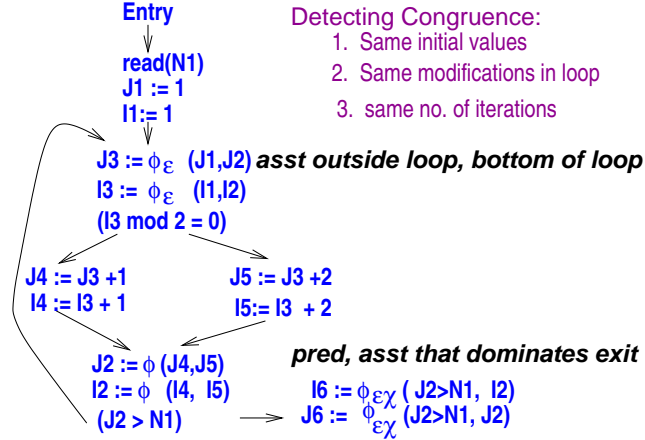
12

Taking Control Flow into account



13

Loop Example



14

Other Extensions

Incorporate arrays, pointers

Update, Access functions

Take commutativity into account

*Ex. $a*b$ same as $b*a$*

Combine with hash-based approach

(Cooper et. al.)

15