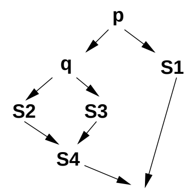
Control Dependences

```
if p then S1
elseif q then S2
          else S3
     endif
     S4
endif
```



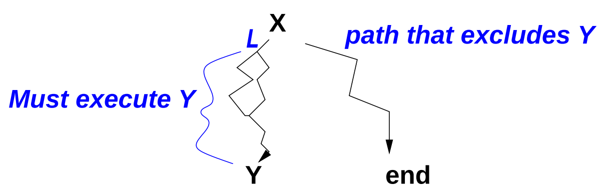
Sequential Program ↔ Fixed Order

Goal: Remove Unnecessary Order

Useful for parallelism

Control Dependence Intuition

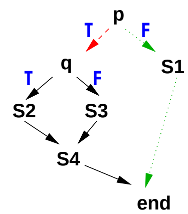
Def: **Y is *control dependent* on X with label L iff**



Control Dependence example

```
if p then S1
elseif q then S2
         else S3
     endif
     S4
endif
```

must execute q, S4



exclude q, S4

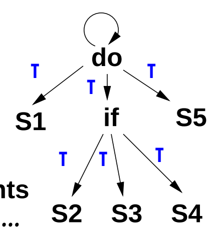
q, S4 have same control dependence

on p with label T

Other control dependences?

Potential Parallelism in Procedures

```
do 10 i = 1,N
    S1
    if p then
        S2
        S3
        S4
     endif
    S5
enddo
```



parallel

parallel

Between Statements

Call, Do,...

Inside Iterations

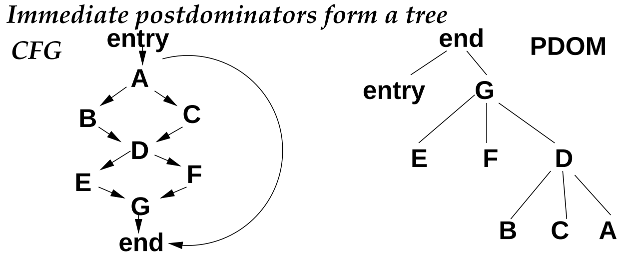
Nested

Postdominator Relation

Def: **X** ***postdominates*** **Y iff**
X is on every path in CFG from Y to end

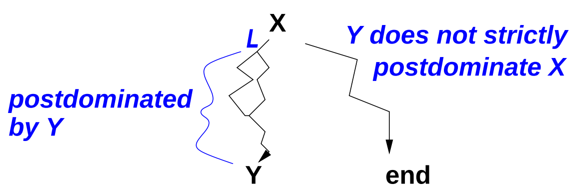
strictly postdom. **iff X ≠ Y and X postdom. Y**

Immediate postdominators form a tree



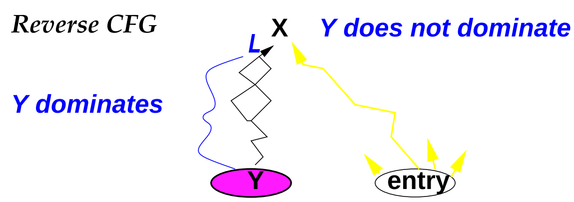
Control Dependence Definition

Def: **Y is** ***control dependent*** **on X with label L iff**



Control Dependence & Dominators

Def: **Y is *control dependent* on X with label L iff X in DF(Y) in Reverse CFG**



Control Dep. & Dominance Frontiers

Y is in CD(X) in CFG G ⟷

X in DF(Y) in Reverse CFG

Good Algorithm for CD

↓

Good Algorithm for DF

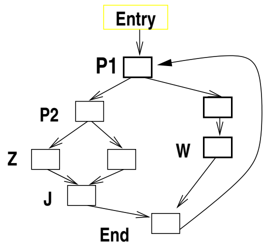
↓

SSA acceptance

efficient

well-defined

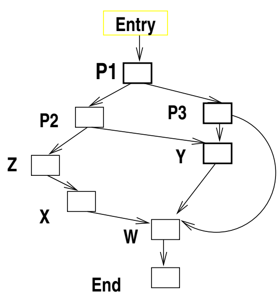
Control Dependence Example



Reverse CFG	*CFG*
P2 in DF(Z)	Z in CD(P2)
P1 in DF(P2)	P2 in CD(P1)
P1 in DF(W)	W in CD(P1)
P1 in DF(P1)	P1 in CD(P1)
P1 in DF(J)	J in CD(P1)

Control Dependence Example

Reverse CFG *CFG*



Data Dependence

Def: **S2 is *data dependent* on S1 w.r.t. variable**

X iff there is a path of nonzero length in the CFG

from S1 to S2, with no intervening def. of X,

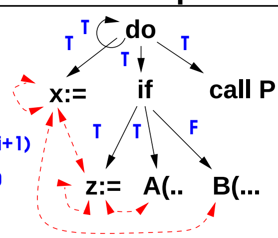
and either

S1:	**X:=**	**:=X**	**X:=**
S2:	**:=X**	**X:=**	**X:=**
	flow	*anti*	*output*

(storage-related)

Program Dependence Graph Example

```
do while (0 < i < n)
    x = FOO (y)
    if p then
        z := x + y
        A(2*i) := z + A(2*i+1)
    else B(i) := x + 5.0
    call P(i)
enddo
```



Not all dependences shown (e.g. var. i)

Which are flow, anti, output?

Data Dependence

Gives constraints on parallelism that must be satisfied

Must be honored to have correct program

Any order that does not violate these dependences is correct!

Program Dependence Graph =

Control Dependence Graph +

Data Dependences

Program Dependence Graph (PDG)

Facilitates performing most traditional optimizations

Constant folding, scalar propagation, common subexpression elimination, code motion, reduction in strength

Requires only single walk over PDG

Exposes more possibilities for re-order

Incremental changes

update data dependence when c.d. changes

Data Dependence Analysis

for linear subscript expressions

Dependence Equations

```
Ex.   do i = 1,10
        ...A(3*i+1)...
        ...A(5*i+2)...
      enddo
```

$3X + 1 = 5Y + 2$

$1 \leq X,Y \leq 10$

Decision Algorithms

Any integer solution	***linear***
Bounded rational solution	***linear***
Bounded integer solution	***exponential***

Data Dependence Analysis

```
Ex 1. do i = 1,10
        A(2*i)
        A(2*i+1)
      enddo
```

Independent

```
Ex 2. do i = 1,10
        A(i)
        A(i-1)
```

Dependent with distance 1 (to next iteration)

```
Ex 3. do i = 1,10
        A(i)
        A(2 *i)
```

Dependent with direction < (to future iteration)

Data Dependence Analysis

Ex 4.
```
do j = 1,100
   do i = 1,100
      A(i, j)
      A(i-1,j)
   enddo
enddo
```

Dependence vector **(0 , 1)**
j i

loop-carried dependence on i loop

GCD Test

Th. If gcd(a1, a2, ..., an) $\nmid$ c, then there is no integer solution to the equation

a1 * i1 + a2 * i2 + + an * in = c

Ex. A(2 * i) A (2 * i + 1)

2 * i1 = 2 * i2 + 1

2 * i1 - 2 * i2 = 1

gcd(2 , -2) = 2, and 2 $\nmid$ 1

so the theorem guarantees no integer solutions

Independence

Ex. A(i) A(i-1)