Control Dependence
(Section 7.3)

Phil Schielke

April 13, 1995

Overview

Define control dependence

See how we can use control dependence information to generate code.

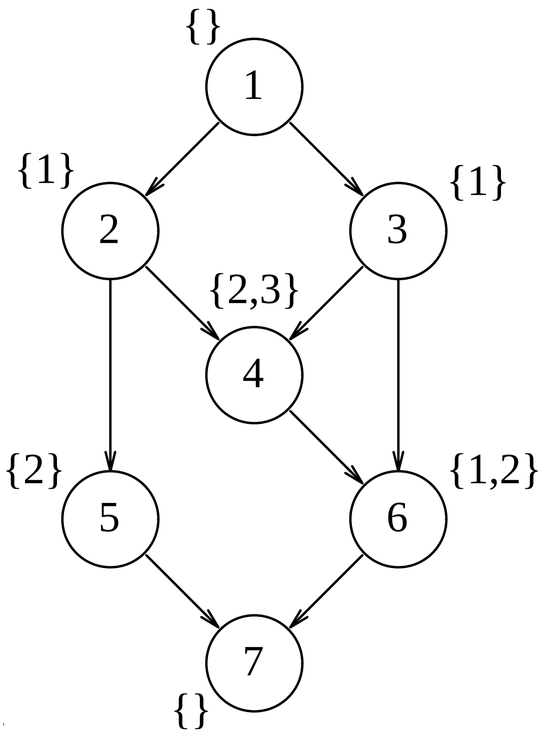
Motivation

Why not just use IF-conversion?

Too many conditional statements

What if parallelization is unsuccessful?

Control Dependence



Dominance

Definition: Let X and Y be nodes in the control flow graph of a program. If X appears on every path from Entry to Y, then X dominates Y.

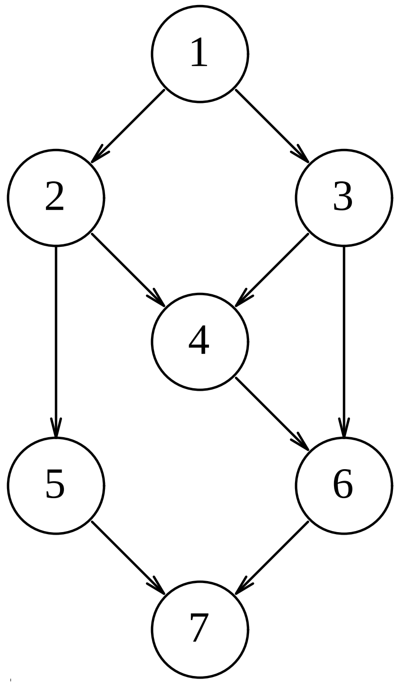
Definition: If X dominates Y and $X \neq Y$, then X strictly dominates Y.

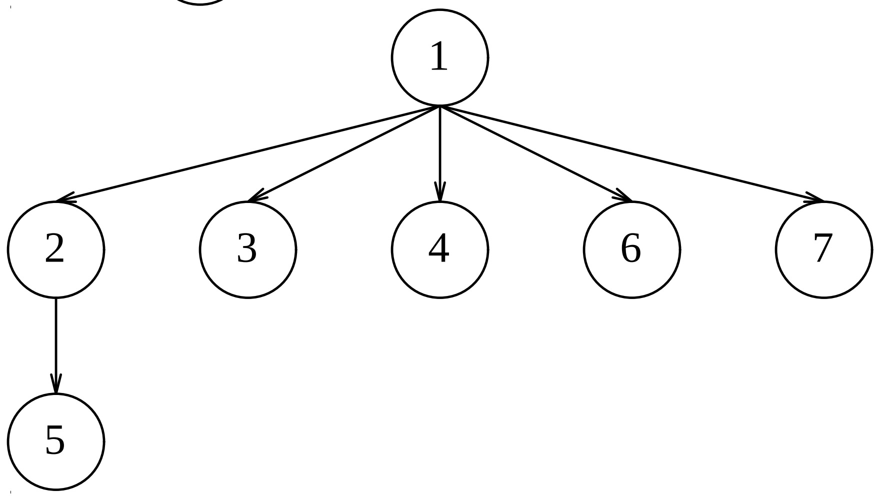
Definition: The immediate dominator of Y (denoted *idom*(Y)) is the closest strict dominator of Y on any path from Entry to Y.

Post-Dominance

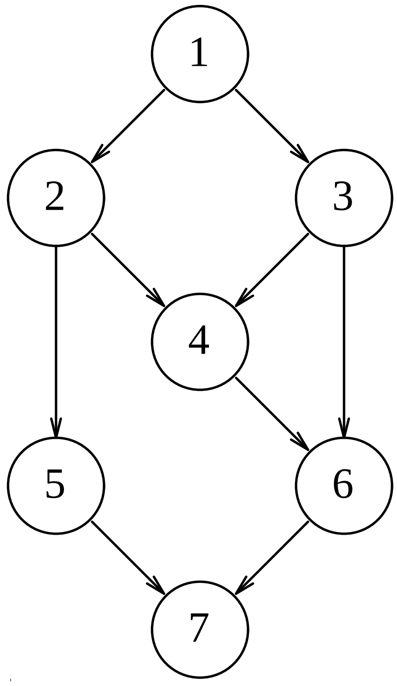
Definition: Let X and Y be nodes in the control flow graph of a program. If X appears on every path from Y to the exit node, then X postdominates Y.

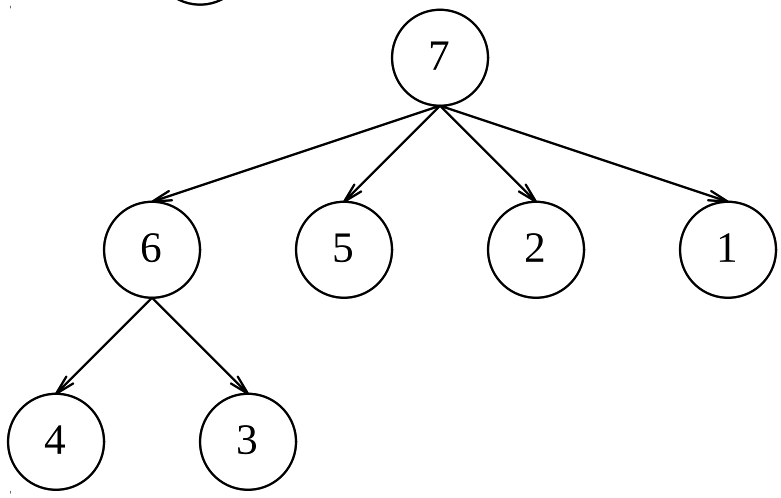
Dominator Tree





Post-Dominator Tree





Control Dependence

Definition: A statement Y is said to be control dependent on another statment X if

(1) there exists a non-trivial path from X to Y such that every statement $Z \neq X$ in the path is postdominated by Y and

(2) X is not postdominated by Y.

Control Dependence and Transformations

Main concern is loop distribution.

```
    DO I = 1, N
S1     IF (A(I) < B(I)) GOTO 20
S2        B(I) = B(I) + C(I)
20     CONTINUE
    ENDDO

    DO I = 1, N
S1     IF (A(I) < B(I)) GOTO 20
    ENDDO

    DO I = 1, N
S2     B(I) = B(I) + C(I)
    ENDDO

20  CONTINUE
```

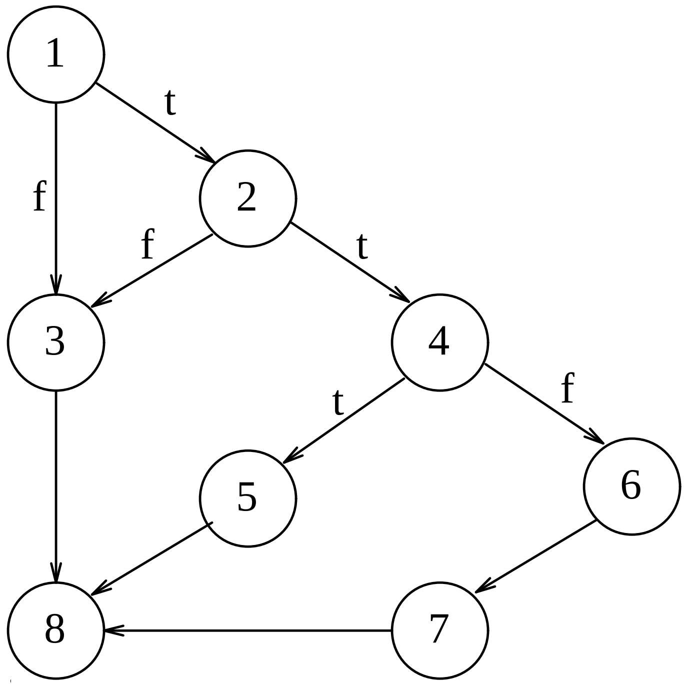
Control Dependence and Transformations

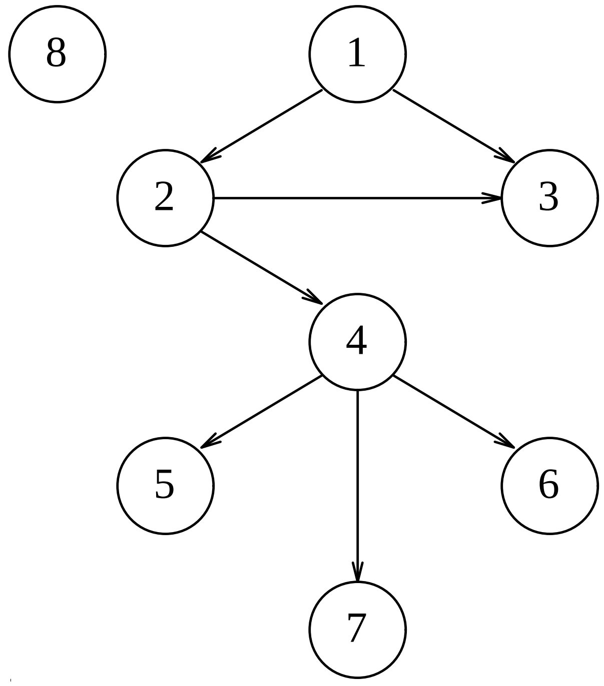
```
    DO I = 1, N
S1     e(I) = A(I) < B(I)
    ENDDO

    DO I = 1, N
S2     IF (e(I) = FALSE) B(I) = B(I) + C(I)
    ENDDO
```

Execution Variables

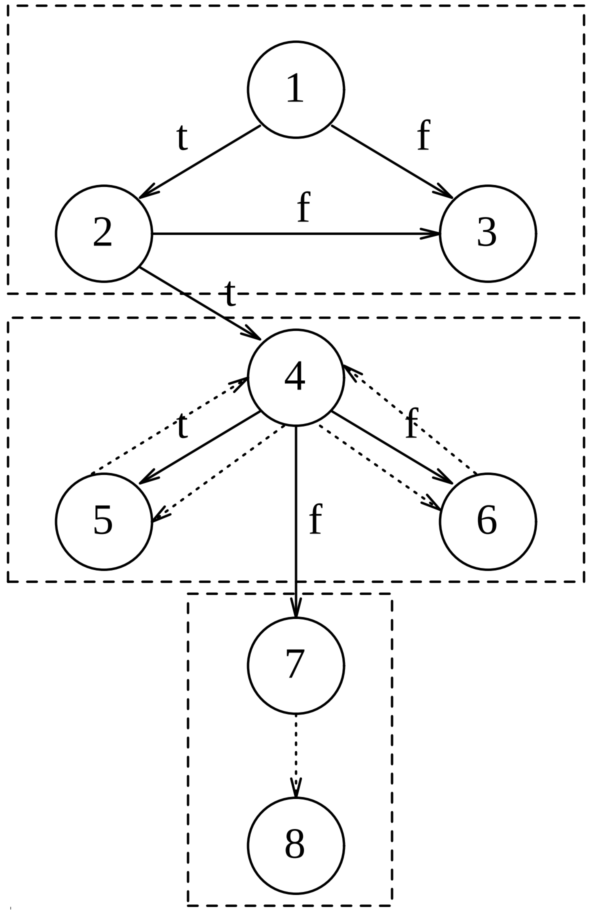
```
    DO I = 1, N
1      IF (A(I) <> 0) THEN
2         IF (B(I)/A(I) > 1) GOTO 4
       ENDIF
3      A(I) = B(I)
       GOTO 8
4      IF (A(I) > T) THEN
5         T = (B(I) - A(I)) + T
       ELSE
6         T = (T + B(I)) - A(I)
7         B(I) = A(I)
       ENDIF
8         C(I) = B(I) + C(I)
    ENDDO
```

Example





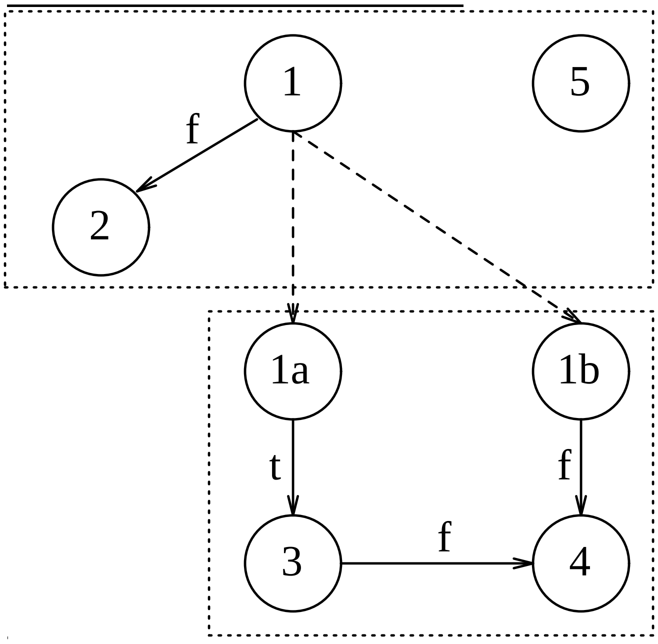
Example



Example

```
    DO I = 1, N
       E2(I) = top
1      IF (A(I) <> 0) THEN
2         E2(I) = (B(I)/A(I) > 1)
       ENDIF
3      IF(E2(I) <> true) A(I) = B(I)
       GOTO 8
       IF (E2(I) = true) THEN
4       IF (A(I) > T) THEN
...
```

Code Generation



Second loop:

```
  DO I = 1, N
     IF(E1(I) = FALSE) GOTO 4
     IF(E1(I) = TRUE) THEN
       IF (NOT p4) THEN
4:        S4
       ENDIF
     ENDIF
  ENDDO
```

Structured Code Generation

Theorem 7.3: The body of a DO-loop that contains no gotos or other DO-loops will give rise to a control dependence graph in which no statement is control dependent on more than one other statement in the loop body.

Generating code for structured DO-loops is fairly straightforward.

Example

```
    DO I = 1, N
S1     IF (p1) THEN
          S2
       ELSE
          S3
       ENDIF
    ENDDO
```

After *DistrbuteCDG(G,Gcd,*$\{\{$*S1,S2*$\}$,$\{$*S3*$\}\}$*)*

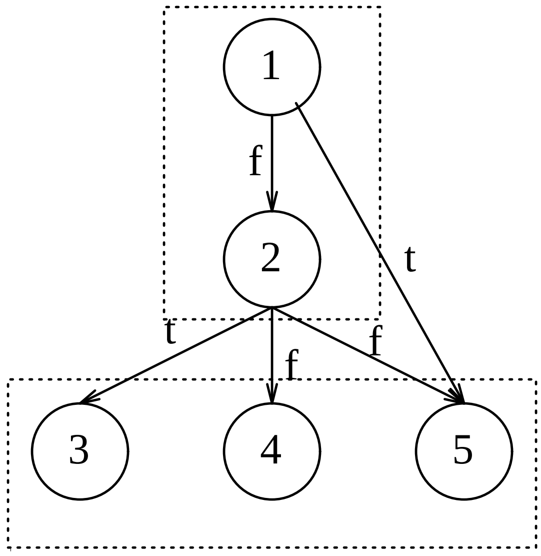
```
    DO I = 1, N
S0     E1(I) = top
S1     E1(I) = p1
S1a    IF (E1(I) = true)
          S2
S4     IF (E1(I) = false)
          S3
    ENDDO
```

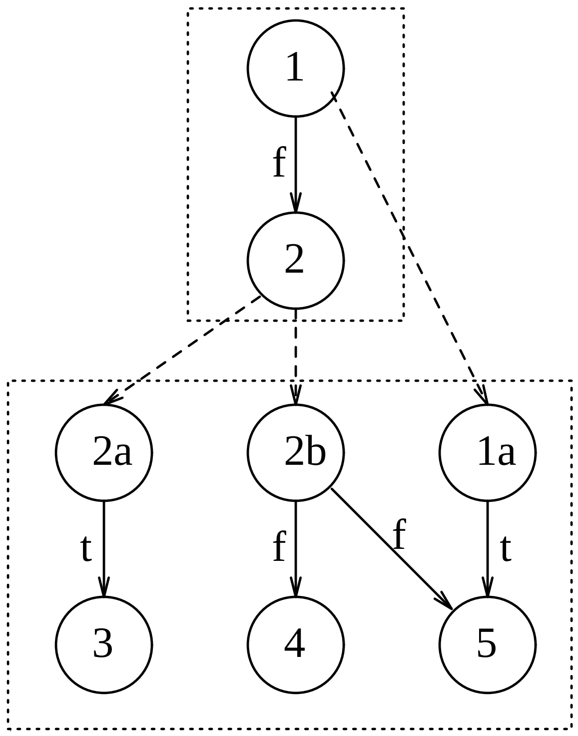
Example

Now generate the code (with loop distribution)

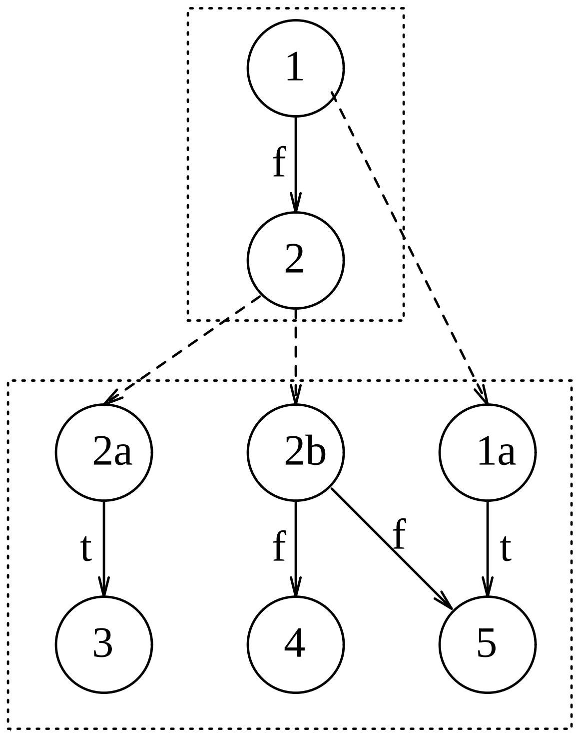
```
    DO I = 1, N
S0    E1(I) = top
S1    E1(I) = p1
S1a   IF (E1(I) = true)
        S2
      ENDIF
    ENDDO

    DO I = 1, N
S4    IF (E(I) = false)
        S3
      ENDIF
    ENDDO
```

```
     DO I = 1, N
S1     IF (p1) GOTO 5
S2     IF (p2) THEN
         S3
       ELSE
         S4
5        S5
       ENDIF
     ENDDO
```






```
    DO I = 1, N
       E1(I) = top
       E2(I) = top
S1     E1(I) = p1
       IF (E1(I) = false) THEN
S2        E2(I) = p2
       ENDIF
    ENDDO
```




```
     DO I = 1, N
       IF (E1(I) = true) GOTO 5
       IF (E2(I) = true) THEN
         S3
       ELSE
         S4
5        S5
       ENDIF
     ENDDO
```