Control Dependences

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

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

Must execute Y

path that excludes Y
Control Dependence example

if p then S1
else if q then S2
  else S3
endif
endif
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

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.** $X$ **postdominates** $Y$ 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

$Y$ does not strictly postdominate $X$

---
Control Dependence & Dominators

Def: Y is \textit{control dependent} on X with label L \textit{iff} X in DF(Y) in Reverse CFG

Reverse CFG

Y dominates \quad Y \textit{does not dominate}

Control Dep. & Dominance Frontiers

Y is in CD(X) in CFG G \quad X in DF(Y) in Reverse CFG

\begin{itemize}
  \item \textit{Good Algorithm for CD}
  \item \textit{Good Algorithm for DF}
  \item SSA acceptance
\end{itemize}

\begin{itemize}
  \item efficient
  \item well-defined
\end{itemize}
Control Dependence Example

<table>
<thead>
<tr>
<th>Reverse CFG</th>
<th>CFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 in DF(P2)</td>
<td>Z in CD(P1)</td>
</tr>
<tr>
<td>P1 in DF(Z)</td>
<td>Z in CD(P2)</td>
</tr>
<tr>
<td>P1 in DF(W)</td>
<td>W in CD(P1)</td>
</tr>
<tr>
<td>P1 in DF(P1)</td>
<td>P1 in CD(P1)</td>
</tr>
<tr>
<td>P1 in DF(J)</td>
<td>J in CD(P1)</td>
</tr>
</tbody>
</table>

Control Dependence Example

<table>
<thead>
<tr>
<th>Reverse CFG</th>
<th>CFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2 in DF(Z)</td>
<td>Z in CD(P2)</td>
</tr>
<tr>
<td>P1 in DF(P2)</td>
<td>P2 in CD(P1)</td>
</tr>
<tr>
<td>P1 in DF(W)</td>
<td>W in CD(P1)</td>
</tr>
<tr>
<td>P1 in DF(P1)</td>
<td>P1 in CD(P1)</td>
</tr>
<tr>
<td>P1 in DF(J)</td>
<td>J in CD(P1)</td>
</tr>
</tbody>
</table>
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: \( X := X \) & \( X := \) & \( X := \)

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. \( \text{do } i = 1,10 \)
\( \ldots A(3i+1) \ldots \)
\( \ldots A(5i+2) \ldots \)
\( \text{endo} \)

\[ 3X + 1 = 5Y + 2 \]
\[ 1 \leq X,Y \leq 10 \]

Decision Algorithms

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

Data Dependence Analysis

Ex. 1. \( \text{do } i = 1,10 \)
\( A(2i) \)
\( A(2i+1) \)
\( \text{endo} \)

Independent

Ex. 2. \( \text{do } i = 1,10 \)
\( A(i) \)
\( A(i-1) \)

Dependent with distance 1 (to next iteration)

Ex. 3. \( \text{do } i = 1,10 \)
\( A(i) \)
\( A(2i) \)

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)
(1, j)

loop-carried dependence on i loop

GCD Test

Th. If \( \text{gcd}(a_1, a_2, ..., a_n) \mid c \), then
there is no integer solution to the equation
\[ a_1 \cdot i_1 + a_2 \cdot i_2 + \cdots + a_n \cdot i_n = c \]

Ex. \( A(2 \cdot i) ) A(2 \cdot i + 1) \)
\[ 2 \cdot i_1 = 2 \cdot i_2 + 1 \]
\[ 2 \cdot i_1 - 2 \cdot i_2 = 1 \]
\( \text{gcd}(2, -2) = 2, \text{ and } 2 \mid 1 \)

so the theorem guarantees no integer solutions

Independence

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