Control Dependences

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
Control Dependence example

If \( p \) then S1
else if \( q \) then S2
else S3
endif
endif

S1 q S2 S3 S4

q, S4 have same control dependence

\textit{on} \( p \) \textit{with label T}

Other control dependences?

Potential Parallelism in Procedures

\texttt{do 10 } \texttt{i = 1,N}
\texttt{S1}
\texttt{if } \texttt{p then}
\texttt{S2}
\texttt{S3}
\texttt{S4}
\texttt{endif}
\texttt{S5}
\texttt{enddo}

Between Statements
\textit{Call, Do,...}

Inside Iterations
Nested

parallel
Postdominator Relation

\textit{Def:} \( X \) postdominates \( Y \) iff 
\( X \) is on every path in CFG from \( Y \) to end

\textit{strictly postdom.} \( Y \) iff \( X = Y \) and \( X \) postdom. \( Y \)

\textbf{Immediate postdominators form a tree}

\textbf{CFG}

\begin{itemize}
  \item \text{entry} \rightarrow \text{A}
  \item \text{B} \rightarrow \text{D}
  \item \text{E} \rightarrow \text{D}
  \item \text{G} \rightarrow \text{end}
\end{itemize}

\textbf{PDOM}

\begin{itemize}
  \item \text{entry} \rightarrow \text{G}
  \item \text{E} \rightarrow \text{F}
  \item \text{D} \rightarrow \text{B}
  \item \text{C} \rightarrow \text{A}
\end{itemize}

Control Dependence Definition

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

\textit{postdominated by} \( Y \)

\textit{Y does not strictly postdominate} \( X \)

\textbf{end}

\textbf{end}
Control Dependence & Dominators

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

Y dominates

X

Y does not dominate

entry

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

<table>
<thead>
<tr>
<th>Reverse CFG</th>
<th>CFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ting Z in CD(P2)</td>
<td></td>
</tr>
<tr>
<td>ting P2 in DF(Z)</td>
<td>Z in CD(P2)</td>
</tr>
<tr>
<td>ting P1 in DF(P2)</td>
<td>P2 in CD(P1)</td>
</tr>
<tr>
<td>ting W in CD(P1)</td>
<td></td>
</tr>
<tr>
<td>ting P1 in DF(P1)</td>
<td>P1 in CD(P1)</td>
</tr>
<tr>
<td>ting 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=
S2: ⊆ X X= ⊆ X

flow | output
anti | (storage-related)
Program Dependence Graph Example

do while (0 < i < n)
  y = FOO (i)
  if p then
    z = x + y
    A(2^i) := z + A(2^{i+1})
  else B() := x + 5.0
  call P(i)
endo

do y := call P
  if T then
    T
  else
    F
  z := A(...) B(...) 

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
Ex.  do i = 1,10
     ...A(3*i+1)...
     ...A(5*i+2)...
   enddo
  Dependence Equations
    3X +1 = 5Y + 2
    1 ≤ X,Y ≤ 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

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

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

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)

loop-carried dependence on i loop
GCD Test

Th. If \( \gcd(a_1, a_2, \ldots, a_n) \mid c \), then there is no integer solution to the equation

\[
a_1 \cdot i_1 + a_2 \cdot i_2 + \ldots + a_n \cdot i_n = c
\]

Ex. \( A(2 \cdot i) \) \( A(2 \cdot i + 1) \)

\[
2 \cdot i_1 = 2 \cdot i_2 + 1
\]

\[
2 \cdot i_1 - 2 \cdot i_2 = 1
\]

\( \gcd(2, -2) = 2 \), and 2 \( \mid 1 \)

so the theorem guarantees no integer solutions

Independence

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