Module 16: Data Flow Analysis in Presence of Procedure Calls

Lecture 32: Iteration

The Lecture Contains:

- Iteration Space
- Iteration Vector
- Normalized Iteration Vector
- Dependence Distance
- Direction Vector
- Loop Carried Dependence Relations
- Dependence Level
- Iteration Vector - Triangular Space
- Non Tightly Nested Loops
- Code Sinking
- Data Dependence With Conditionals
- Conditionals in Loops

Previous Next

Module 16: Data Flow Analysis in Presence of Procedure Calls

Lecture 32: Iteration

Iteration Space

- Concept associated with the loops
- Contains one point for each iteration of the loop
- If a statement in one iteration dependes on a statement in another iteration then dependence is represented by an edge from source to target (called iteration space dependence graph)

```
for i = 2 to 9 do
x[i] = . . .
= . . . x[i-1] . . .
endfor
```

- Space requirement too large
- Compiler can not always determine number of iterations

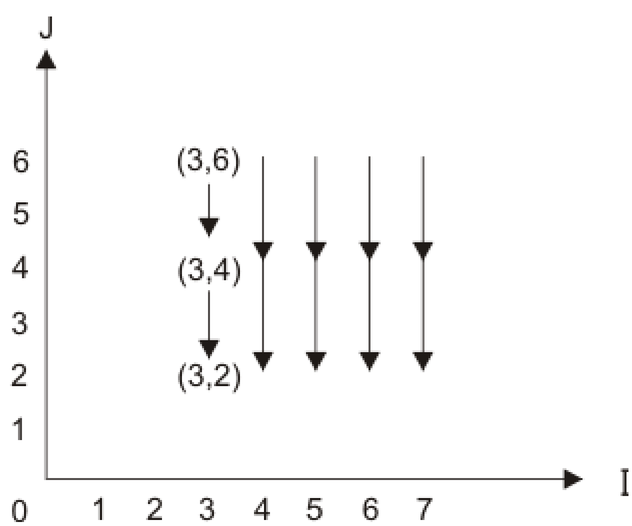
Iteration Vector

Each iteration is assigned a vector

iv = (I_1, I_2, . . ., In)

Where I_k is the value of loop index variable of kth nested loop at that iteration.

```
for i = 3 to 7 do
for j = 6 to 2 step -2 do
a[i,j] = a[i,,j+2] + 1
endfor
endfor
```



Previous Next

Normalized Iteration Vector

- Iterations are labeled 0, 1, 2, . . .
- Advantage
  - Later iterations always have larger vector than earlier iterations if $i_1 < i_2$ then i_1 is always executed before i_2
  - next iteration is always one more than the current iteration $i_k^n = (i_k^{iv} - l_k)/S_k$

Dependence Distance

Vector difference between iteration vector of the source and the target iterations. $d = i^T - i^S$

Direction Vector

- Frequently used but less precise
- It is ordering vector relating the source and the target iteration vectors
- For some optimizations direction information is sufficient
- Some times distance is not fixed but the direction is fixed

```
for i = 1 to 10 do
A[2*I] = B[I] + 1
C[I] = A[I]
endfor
```

- Distance varies from 1 to 5, therefore $S_2 \, \delta_*^f \, S_3$
- Direction is constant, $S_2 \, \delta_<^f \, S_3$

Loop Carried Dependence Relations

```
for i = 1 to n do
for j = 1 to n do
A[i,j] =
= A[i,j]
B[i,j+1] =
= B[i,j]
C[i+1,j] =
= C[i,j+1]
endfor
endfor
```

Previous Next

- For statements involving A[i,,j]
  distance vector (0,0)
  direction vector (=,=)
  dependence is loop independent
- For statements involving B[i,,j]
  distance vector (0,1)
  direction vector (=,<)
  dependence is carried by the inner loop
- For statements involving C[i,,j]
  distance vector (1,-1)
  direction vector (<,>)
  dependence is carried by the outer loop

Dependence Level

Loop nest level that carries the data dependence relation

Therefore, for C[i,,j] level is 1
for B[i,,j] level is 2
for A[i,,j] level is ∞

Iteration Vector - Triangular Space

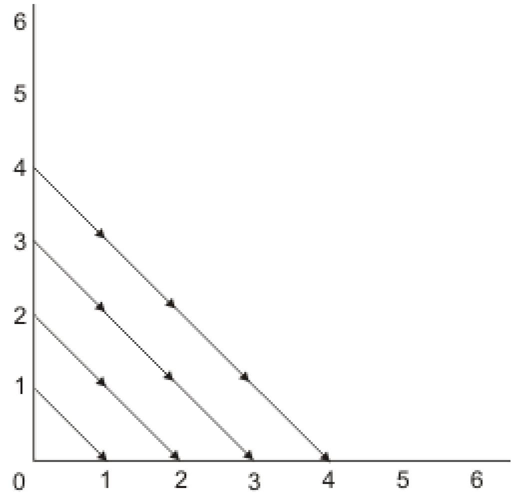
Advantages of normalized iteration vector:

- Later iterations have larger iteration vector
- Adjacent iterations differ by only one
  However, neither of these require first iteration to have vector (0,0, · · ·, 0).

Consider following program:

```
for i = 1 to 7 do
for j = i to 7 do
A[i+1,,j] = A[i,,j]+1
endfor
endfor
```

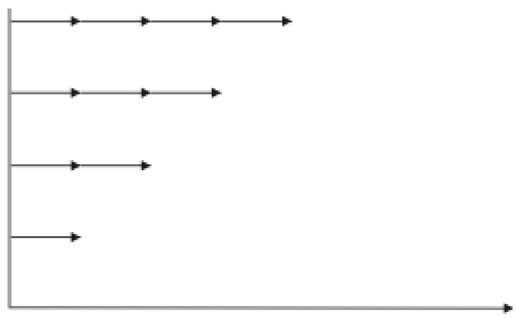
Previous Next



Normalized Iteration Space

Dependence distance vector is (1, -1)

A more natural way to depict such a loop is to assign iteration vectors that give triangular space as:



Dependence Distance is (0,1)

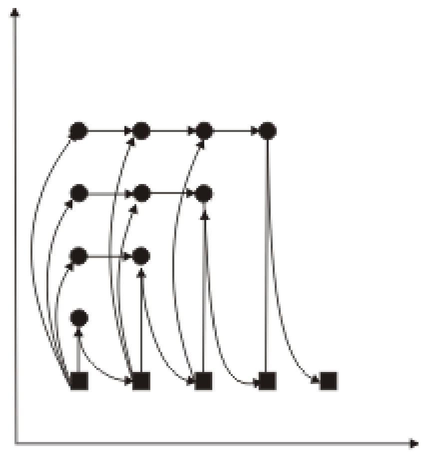
Semi normalized iteration vector: when distance between successive iterations is 1 and the lower limit is not zero

Previous Next

Non Tightly Nested Loops

A nested loop where outer loop contains additional statements outside inner loop. For example

```
for i = 1 to n do
B[i] = B[i] / A[i,i]
for j = i+1 to n do
B[j] = B[j]-A[i,,j]*B[i]
endfor
endfor
```



for n = 5

$$S_4 \delta^f_{(1,0)} S_4$$

$$S_2\, \delta^f_0\, S_4$$

$$S_4 \delta^f_1 S_2$$

Code Sinking

Sink outer loop body into the inner loop by adding conditionals:

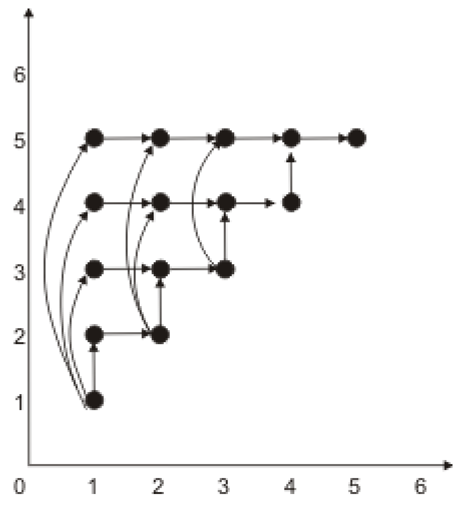
```
for i = 1 to n do
for j = i to n do
if (j=i) B[i] = B[i] / A[i,i]
if (j>i) B[j] = B[j]-A[i,,j]*B[i]
endfor
endfor
```

Previous Next

Module 16: Data Flow Analysis in Presence of Procedure Calls

Lecture 32: Iteration

Code Sinking



for n = 5

$S_4 \delta^f_{(1,0)} S_4$

$S_2 \delta^f_{(=,<)} S_4$

$S_4 \delta^f_{(1,1)} S_2$

Data Dependence With Conditionals

- There must be a path from definition to use
- In case of conditionals, path cannot be determined at compile time
- Any path may be taken at runtime
- Data dependence cannot occur between statements in alternate paths

Previous Next

Module 16: Data Flow Analysis in Presence of Procedure Calls

Lecture 32: Iteration

1. X = 1
2. Y = 2
3. if Y < T then
4. X = 2
5. else
6. Y = X
7. endif
8. Z = X + Y

Find out data dependence for X.

- Definition of X in 1 and 4
- Use of X at 6 and 8

Therefore

$S_1\ \delta^f\ S_6 \quad S_1\ \delta^f\ S_8$
$S_4\ \delta^f\ S_8 \quad S_1\ \delta^o\ S_4$

Conditionals in Loops

- Statements that can’t be executed on the same iteration cannot be involved in a loop-independent dependence
- Conditionals do not affect loop carried dependence

1. for i = 2 to 9 do
2. if a(i) > 0 then
3. a(i) = b(i-1) + 1
4. else
5. b(i) = a(i) * 2
6. endif
7. endfor

Note: $S_2\ \overline{\delta}\ S_3 \quad S_5\ \delta f\ S_3$

Previous Next