Dependence (Objectives)

- To understand the concept of data dependences as applied to array variables
- To understand dependence in relation to loops
- To understand distance and direction vectors and how they describe a dependence

Dependence Definition

- Models memory access behavior of array references
- Ensures program correctness for transformations
- Quantify memory behavior of loops
- Given two references $R1$ and R2, $R2$ **depends** on $R1$ (or there is a dependence from $R1$ to $R2$) if
  - both references access the same memory location (and at least one of them stores to it)
  - there is a feasible run-time execution path from R1 to $R2$

Dependence Types

- *True Dependence*: the first reference stores into a location that is later read by the second reference ($R1\ \delta\ R2$)

```
A(I) = ...        R1
... = A(I)        R2
```

- *Antidependence*: the first reference reads from a location into which the second reference later stores ($R1\ \delta^{-1}\ R2$)

```
... = A(I)        R1
A(I) = ...        R2
```

Dependence Types

- *Ouput Dependence*: the both references store into a location ($R1 \ \delta^o R2$)

```
A(I) = ...        R1
A(I) = ...        R2
```

- *Input dependence* (not a real dependence, used for analysis only): the both references reads from a location ($R1 \ \delta^i R2$)

```
... = A(I)        R1
... = A(I)        R2
```

Dependence and Loops

- *Iteration vector*: a vector of values for the loop control (induction) variables
  - one entry per variable
  - indexed from outermost to innermost
- The set of all iteration vectors for a loop is called the *iteration space*

```
DO I = 1, 10
    DO J = 3, 5
        DO K = 6, 9
           A(I,J,K) = ...
```

when I = 1, J = 4, and K = 7 the iteration vector is <1,4,7>

Distance Vectors

- Using iteration vectors we can describe the distance and direction between two references
  - how far apart?
  - legality
- **Distance Vector**: If two iteration vectors **i** and **j** represent the execution of two references that are contained in *n* common loops, then the *distance vector* (or distance between the references) is defined as a vector of length *n* such that
  - $d(i,j)_k = j_k - i_k,\ 1 <= k <= n$

Example Distance Vector

```
DO I = 1, 100
   DO J = 1, 100
      DO K = 1, 100
         A(I,J,K) =
```

- The distance vector between the reference of A(1,1,1) on iteration <1,1,1> and the reference of A(3,5,4) on iteration <3,5,4> is <2,4,3>

Dependence Distance Vector

- Suppose that there is a dependence from reference $R1$ on iteration **i** of a loop nest to reference $R2$ on iteration **j** of the loop nest, then the *dependence distance vector* is *d(i,j)*

```
DO I = 1, 100
   DO J = 1, 100
      A(I,J) = A(I-3, J-1)
```

A(2,2) is accessed by A(I,J) on iteration <2,2>
A(2,2) is accessed by A(I-3,J-1) on iteration <5,3>
The distance vector is <3,1>

Direction Vector

- Direction Vector: if two iteration vectors **i** and **j** represent the execution of two reference that are contained in *n* common loops, then the direction vector is defined as a vector of length *n* such that

$$D(i,j)_k = \begin{cases} "<" & if \quad d(i,j)_k > 0 \\ "=" & if \quad d(i,j)_k = 0 \\ ">" & if \quad d(i,j)_k < 0 \end{cases}$$

Example

```
DO I = 1, 100
    DO J = 1, 100
        A(I,J) = A(I-3, J-1)
```

A(2,2) is accessed by A(I,J) on iteration <2,2>.
A(2,2) is also accessed by A(I-3,J-1) on iteration <5,3>.
The direction vector is (<,<)

Why Direction Vectors?

- If a dependence cannot be characterized with a single distance vector, we can summarize the dependences with a direction vector.

```
DO I = 1, N
    A[I] =A[2*I]

ENDDO
```

- This dependence has no single distance vector. It has distances of 1, 2, ... Therefore, it is described with a direction vector (<)

Legal Vectors

- There cannot be a ``>'' in the outermost entry of a direction vector (negative value in a distance vector) by definition. This implies a dependence in the opposite direction.

```
DO I = 1, N
     A[I] =A[I+1]
                    ( > )      Illegal vector!
ENDDO
```

Summarized Vectors

- More than one direction vector can be summarized as ≤, ≥, ≠, *

```
DO I = 1, N
      A(2*I-1) = …
                      = A(I)
ENDDO            ( ≤ )
```

Loop Independent Dependence

- A dependence from *R*1 to *R*2 is *loop independent* iff there exists two iteration vectors **i** and **j** such that the following two conditions hold:
  - reference *R*1 refers to memory location *M* on iteration **i**; *R*2 refers to *M* on iteration **j**; and $d(i,j)$ = <0,...,0>
  - There is a control flow path from *R*1 to *R*2 within one iteration

```
DO I=1,N
   A(I) = ...

     ...  = A(I)
ENDDO
```

Loop Carried Dependence

- A dependence from *R*1 to *R*2 is *loop carried* iff there exists two iteration vectors **i** and **j** such that the following two conditions hold:
  - reference *R*1 refers to memory location *M* on iteration **i**; *R*2 referes to *M* on iteration **j**; and
    - $d(i,j)_k > 0$ for some k
    - $d(i,j)_l = 0$ for all $l < k$ (k is the outermost non-zero entry)
  - There is a control flow path from *R*1 to *R*2
- The *level* of a loop-carried dependence is the index of the outermost non-``=" in $D(i,j)$ (0 in $d(i,j)$)

Loop Carried Dependence: example

- In the following example *J* is the carrier of the dependence and it is said to be carried at level 2.

```
DO I = 1,N
    DO J = 1,N
        DO K = 1, N
            A(I,J,K+1) =...A(I,J-1,K)
```

<0, 1, 1>

Problem

- Determine all of the dependences in the following loop. Give the distance and direction vectors for each dependence.

```
DO J = 2, N
    DO I = 2, N
        A(I,J) = 0.175*(A(I-1,J)
                        + A(I+1,J)
                        + A(I,J-1)
                        + A(I,J+1))
                     + 0.3*A(I,J)
    ENDDO
ENDDO
```