



Dependence Analysis

Delivered by
Dr. Jitendra V. Tembhurne

Contents

- Introduction of Dependence analysis
- Different forms of parallelism
- Loop Parallelism
- Parallelism and Data Dependence
- Single Loop
- Dependence Concepts
 - Loop independence
 - Loop carried
- Data dependence
- Dependence in loop
- Types of dependence flow, anti, output and input
- Example on dependence
- Data Dependence graph

Introduction of Dependence Analysis

- Program restructuring
- Dependence constraints
- Dependence problem
 - given two program variables, decide if they have instances that reference the same memory location during execution of the program
- Aim – gather useful information
- Example – loop with non unit strides

Different forms of parallelism

Statement Parallelism

```
cobegin  
  a := b + c  
  d := e + f  
coend
```

Operation Parallelism

```
a = (b+c) * (d+e)
```

Function Parallelism

```
f (a) = if (a <= 1) then return 0  
       else return f(a-1) + f(a-2)  
       endif
```

Loop Parallelism

```
Do i = 1 , n  
  a(i) = b(i)*2  
EndDo
```

Loop Parallelism / Array Parallelism

Original loop

```
Do i = 1, n  
    a(i) = b(i)*2  
EndDo
```

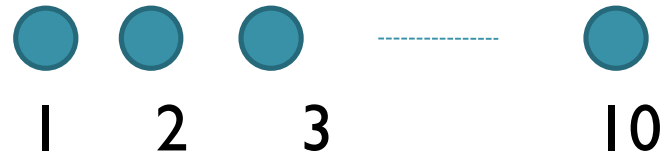
Parallel loop

```
Doall i = 1, n  
    a(i) = b(i)*2  
EndDo
```

- All iterations of the iterator i can be performed independently
- Independence implies Parallelism
- We will concentrate on loop parallelism $O(n)$ potential parallelism. Statement and Operation - $O(1)$.

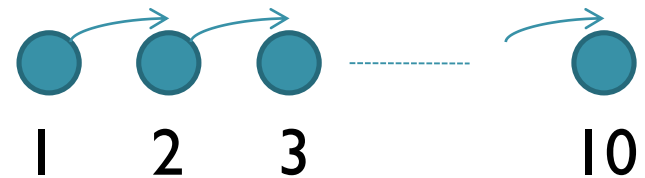
Parallelism and Data Dependence

```
Do i = 1, 100  
  a(i) = b(i) + c(i)  
EndDo
```



- Completely parallel each iteration is totally independent

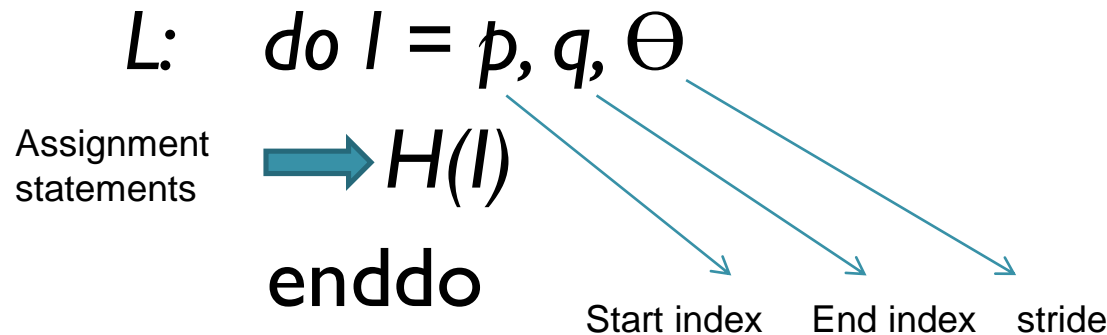
```
Do i = 2, 100  
  a(i) = a(i-1)  
    = function_call(i)  
EndDo
```



- Completely serial each iteration depends on the previous iteration

Single Loop

- **Index and Iteration Spaces**



- Index values (or index points) of L are the values of the index variable l .
- Index space of the loop is the set of all its index points.

Theorem: An integer l is an index value of the loop L iff


- $(l - p)/\theta$ is an integer, and
- $0 < (l - p)/\theta < (q - p)/\theta$.

The l takes integral values for which $(l - p)/\theta$ is integer between 0 and $(q - p)/\theta$.

$\hat{l} = (l - p)/\theta$. Hence, $l = p + \hat{l} \theta$

Where \hat{l} the iteration variable of the loop L .

Theorem: An integer \hat{l} is an iteration value of the loop L iff $0 \leq \hat{l} \leq (q - p)/\theta$



Theorem: The iteration space of L is the set of integers $\{0, 1, \dots, Q\}$

Where $Q = \text{floor}[(q - p)/\theta]$.

Single Loop

- **Normalized loop**

\dot{L} : $do \hat{l} = 0, Q, 1$

Assignment
statements

$\Rightarrow H(p + \hat{l}\theta)$

enddo

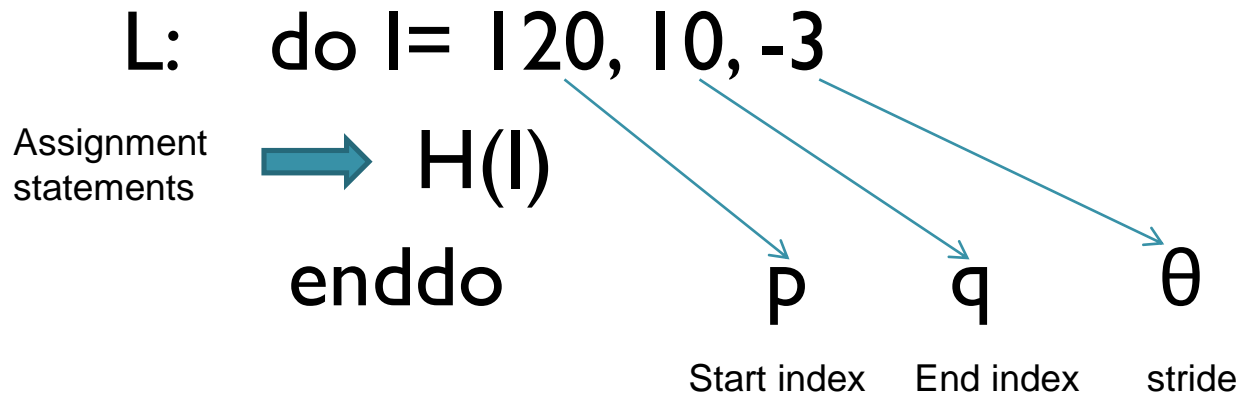
Start index

End index

stride

- Iteration of \dot{L} are $H(p)$, $H(p + \theta)$, $H(p + 2\theta)$,, $H(p + Q\theta)$.

Example



Where,

$$\begin{aligned} \text{Index variable } l = 120 - 3 \hat{l} &\rightarrow l = p + \theta \hat{l} \\ 120 \geq 120 - 3 \hat{l} \geq 10 &\rightarrow 0 \leq \hat{l} \leq 36 \end{aligned}$$

Index space = $\{ 120, 117, 114, \dots, 12 \}$

Iteration values = $\{ 0, 1, 2, 3, \dots, 36 \}$

Example

- Find the index and iteration spaces of the loop L, when
 - $p = 17, q = 39, \Theta = 5$
 - $p = -15, q = 20, \Theta = 2$

Dependence concept

An instance $T(j)$ of a statement T depends on an instance $S(i)$ of a statement S , if there exists a memory location M such that;

1. Both $S(i)$ and $T(j)$ reference (read or write) M ;
2. $S(i)$ is executed before $T(j)$ in the sequential execution of the program;
3. During sequential execution, the location M is not written in the time period from the end of execution of $S(i)$ to the beginning of execution of $T(j)$.

The statements S and T need not be distinct, but Condition 2 requires that the instances $S(i)$ and $T(j)$ be distinct.

Let \hat{i} and \hat{j} denote the iteration values corresponding to i and j , respectively.

Loop Carried dependence

Definition:

- Statement S has a *loop-carried dependence* on T iff there exist two iteration vectors \hat{i} and \hat{j} such that
 - 1) S refers to memory location M on iteration \hat{i} , T refers to M on iteration \hat{j} , and $\hat{i} < \hat{j}$ and $S < T$

Example (I)

```
DO I = 1, 3
S   A(I+1) = F(I)
T   F(I+1) = A(I)
ENDDO
```

$S \delta_{(<)} T$ and $T \delta_{(<)} S$

- The loop-carried dependence $S \delta_{(<)} T$ is forward
- The loop-carried dependence $T \delta_{(<)} S$ is backward
- All loop-carried dependences are of level 1, because $D(\hat{i}, \hat{j}) = 1 (<)$ for every dependence

Loop-Independent Dependence

Definition:

- Statement S has a *loop-independent dependence* on T iff there exist two iteration vectors \hat{i} and \hat{j} such that
 - 1) S refers to memory location M on iteration \hat{i} , T refers to M on iteration \hat{j} , and $\hat{i} = \hat{j}$ and $S < T$ since $S(i)$ is to be executed before $T(i)$

Data Dependence

- The relationship between reads and writes to memory has critical impact on parallelism.
Four types of data dependence;

Flow(true)	Anti	Output	Input
$a =$ $= a$	$= a$ $a =$	$a =$ $a =$	$= a$ $= a$

- Only flow dependences are true dependences. Anti and output can be removed by remaining
- Dataflow analysis can be used to defines data dependences on a per block level for scalars.

Types of Data Dependence

Flow(True) dependence

A statement T is **flow dependent** on S (written $S \delta^f T$) if and only if S modifies a resource that T reads and S precedes T in execution. The following is an example of a flow dependence (**RAW: Read After Write**):

Example: L: do I = 4, 100, 2
 X(I) = I + 2
 Y(I) = X(I) + 2
 enddo

Types of Data Dependence

Anti-dependence

A statement T is **anti-dependent** on S (written $S \delta^a T$) if and only if S modifies a resource that T reads and S precedes T in execution. The following is an example of a anti-dependence (**WAR:Write After Read**):

Example: L: do $I = 4, 100, 2$
 $X(I) = \underline{Y(I)} + 3$
 $\underline{Y(I)} = X(I) + X(I - 1)$
 enddo

Types of Data Dependence

Output dependence

A statement T is **output dependent** on S (written $S \delta^o T$) if and only if S modifies a resource that T reads and S precedes T in execution. The following is an example of a output dependence (**WAR: Write After Write**):

Example: L: do I = 4, 100, 2
 X(I) = Y(I) + 3
 X(I) = X(I) + X(I - 1)
 enddo

Types of Data Dependence

Input dependence

A statement T is **input dependent** on S (written $S \delta^i T$) if and only if S modifies a resource that T reads and S precedes T in execution. The following is an example of a input dependence (**RAR: Read After Read**):

Example: L: do I = 4, 100, 2
 $Z(I) = \underline{X(I)} + 3$
 $Y(I) = \underline{X(I)} + X(I - 1)$
 enddo

Example(1) on Dependence

L: **do** $l = 4, 100, 2$
S: $X(l) = X(l) + l$
T: $Y(2l-4) = X(l) + X(l + l) + X(l + 2) + X(l - 2)$
U: $Y(l) = Z(l)$ ↑ ↑ ↑
 enddo **F** **A** **F**

The index values of the loop are 4, 6, 8, ..., 100, and the corresponding iteration values are 0, 1, 2, ..., 48. Can be calculated from

$$\hat{l} = (l - p) / \theta \text{ where,}$$

$l \rightarrow$ index value

$p \rightarrow$ start value

$\theta \rightarrow$ Stride

$\hat{l} \rightarrow$ iteration value

Iterations of loop

The first three iterations of L along with the last one are shown below:

S(4): $\mathbf{X(4)} = \mathbf{X(4)} + \mathbf{I}$

T(4): $\mathbf{Y(4)} = \mathbf{X(4)} + \mathbf{X(5)} + \mathbf{X(6)} + \mathbf{X(2)}$

U(4): $\mathbf{Y(4)} = \mathbf{Z(4)}$

S(6): $\mathbf{X(6)} = \mathbf{X(6)} + \mathbf{I}$

T(6): $\mathbf{Y(8)} = \mathbf{X(6)} + \mathbf{X(7)} + \mathbf{X(8)} + \mathbf{X(4)}$

U(6): $\mathbf{Y(6)} = \mathbf{Z(6)}$

S(8): $\mathbf{X(8)} = \mathbf{X(8)} + \mathbf{I}$

T(8): $\mathbf{Y(12)} = \mathbf{X(8)} + \mathbf{X(9)} + \mathbf{X(10)} + \mathbf{X(6)}$

U(8): $\mathbf{Y(8)} = \mathbf{z(8)}$

.

.

.

S(100): $\mathbf{X(100)} = \mathbf{X(100)} + \mathbf{I}$

T(100): $\mathbf{Y(196)} = \mathbf{X(100)} + \mathbf{X(101)} + \mathbf{X(102)} + \mathbf{X(98)}$

U(100): $\mathbf{Y(100)} = \mathbf{Z(100)}$

Data Dependence graph of loop

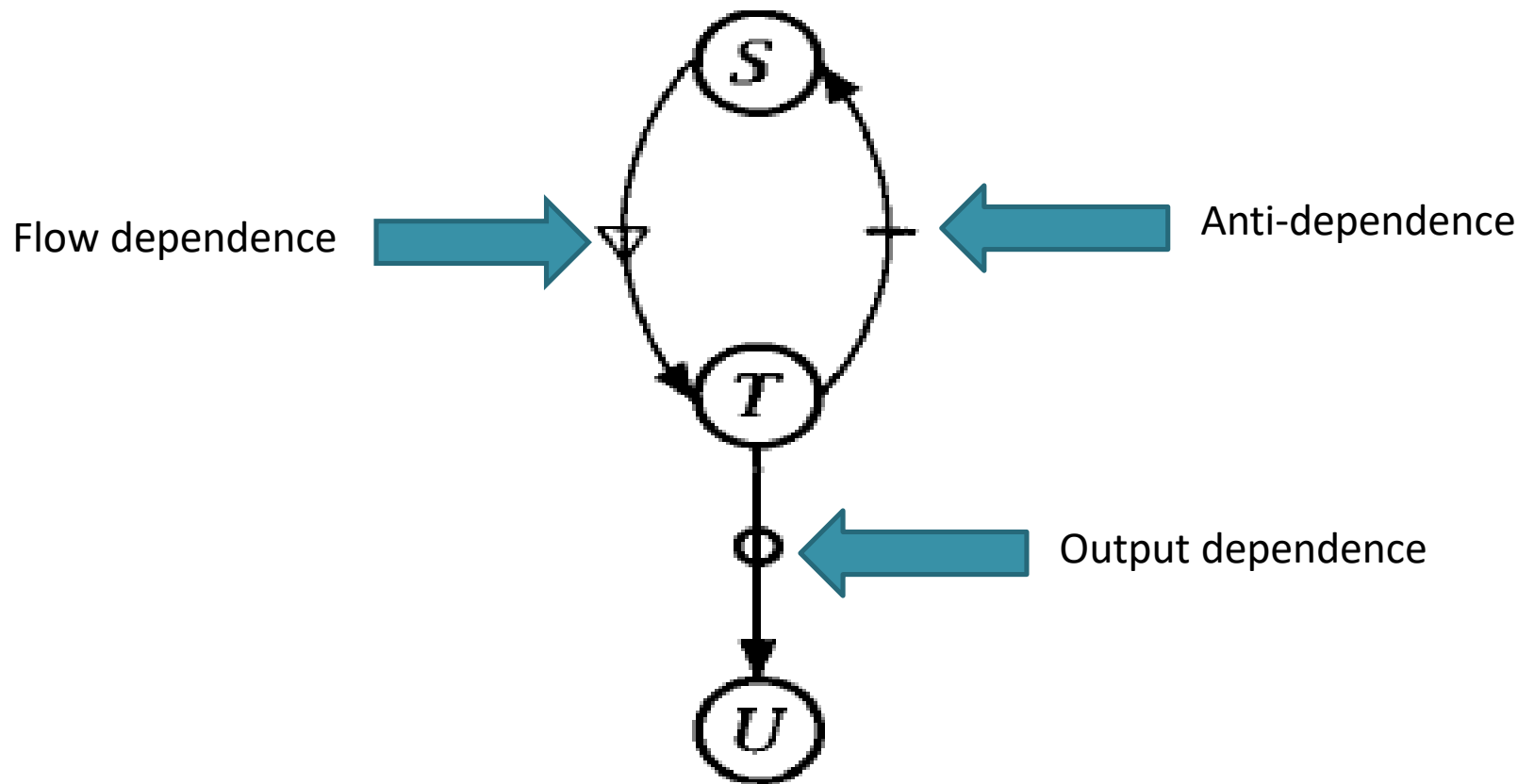


Fig: Statement dependence graph

Example (2)

do $l = 1, 100$

S: $A(l) = B(l+2) + 1$

T: $B(l) = A(l-1) - 1$

enddo

S(1): $A(1) = B(3) + 1$

T(1): $B(1) = A(0) - 1$

S(2): $A(2) = B(4) + 1$

T(2): $B(2) = A(1) - 1$

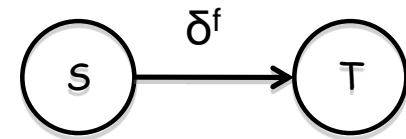
S(3): $A(3) = B(5) + 1$

T(3): $B(3) = A(2) - 1$

.....

S(100): $A(100) = B(102) + 1$

T(100): $B(100) = A(99) - 1$



Due to $A()$

Set of **iteration pairs** associated with this dependence:

$\{(i,j): j=i+1, 1 \leq i \leq 99\}$

Dependence distance:

$j-i = 1$ constant in this case.

Dependence Problem

Example: Do $i = 1, N$
 $a(f(i)) =$
 $= a(g(i))$
 EndDo

1. The flow dependence of T on S consists of all instance pair $(S(i), T(j))$ such that T(j) is flow dependent on S(i).
2. The anti-dependence of T on S consists of all instance pair $(S(i), T(j))$ such that T(j) is anti-dependent on S(i).

Dependence in loop

3. The output dependence of T on S consists of all instance pair $(S(i), T(j))$ such that $T(j)$ is output dependent on $S(i)$.
4. The input dependence of T on S consists of all instance pair $(S(i), T(j))$ such that $T(j)$ is input dependent on $S(i)$.

Data Dependence

In general we need to know if two usages of an array access the same memory location and what type of dependence helpful as this can be done relatively cheaply for simple programs

- General dependence is intractable at $a(f(i)) = a(g(i))$ for arbitrary f, g