Loop Transformations

- Goals
- Data Dependencies
- Iteration Spaces
- The Transformation Matrix, π and S.
- Example Transformations
- Deriving S

Goal of Loop Transforms

- Automatically re-structure code for
  - Parallelism
  - Locality
- Driven by dependency analysis

Defining Dependencies

- Flow Dependence W → R δ^f } true
- Anti-Dependence R → W δ^a } false
- Output Dependence W → W δ^o } false
- Input Dependence R → R δ^i

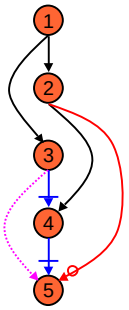
```
S1) a=0;
S2) b=a;
S3) c=a+d+e;
S4) d=b;
S5) b=5+e;
```

Not generally defined

Example Dependencies

```
S1) a=0;
S2) b=a;
S3) c=a+d+e;
S4) d=b;
S5) b=5+e;
```

S1 δ^f S2	due to a
S1 δ^f S3	due to a
S2 δ^f S4	due to b
S3 δ^a S4	due to d
S4 δ^a S5	due to b
S2 δ^o S5	due to b
S3 δ^i S5	due to a



Data Dependence in Loops

- Dependence can flow across iterations of the loop.
- Dependence information is annotated with iteration information.
- If dependence is across iterations it is loop carried otherwise loop independent.

```
for (i=0; i<n; i++) {
   A[i] = B[i];
   B[i+1] = A[i];
}
```


Data Dependence in Loops

- Dependence can flow across iterations of the loop.
- Dependence information is annotated with iteration information.
- If dependence is across iterations it is loop carried otherwise loop independent.

```
for (i=0; i<n; i++) {
   A[i] = B[i];
   B[i+1] = A[i];
}
```

δ^f loop carried

δ^f loop independent

Unroll Loop to Find Dependencies

```
for (i=0; i<n; i++) {
  A[i] = B[i];
  B[i+1] = A[i];
}
```

δ^f loop carried

δ^f loop independent

```
A[0] = B[0];   } i=0
B[1] = A[0];
A[1] = B[1];   } i=1
B[2] = A[1];
A[2] = B[2];   } i=2
B[3] = A[2];
   ⋮
```

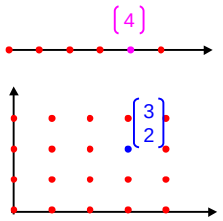
Distance/Direction of the dependence is also important.

Iteration Space

Every iteration generates a point in an n-dimensional space, where n is the depth of the loop nest.

```
for (i=0; i<n; i++) {
   •••
}
```

```
for (i=0; i<n; i++)
   for (j=0; j<4; j++) {
      •••
}
```



Distance Vector

```
for (i=0; i<n; i++) {
   A[i] = B[i];
   B[i+1] = A[i];
}
```

```
A[0] = B[0];   } i=0
B[1] = A[0];
A[1] = B[1];   } i=1
B[2] = A[1];
A[2] = B[2];   } i=2
B[3] = A[2];
   ⋮
```

Distance vector is the difference between the target and source iterations.

$$\mathbf{d} = \mathbf{I}_t - \mathbf{I}_s$$

Exactly the distance of the dependence, i.e.,

$$\mathbf{I}_s + \mathbf{d} = \mathbf{I}_t$$

Example of Distance Vectors

```
for (i=0; i<n; i++)
  for (j=0; j<m; j++){
     A[i,j] =   ;
        = A[i,j];
     B[i,j+1] =   ;
        = B[i,j];
     C[i+1,j] =   ;
        = C[i,j+1] ;
  }
```

$A_{0,2}$= $=A_{0,2}$ $B_{0,3}$= $=B_{0,2}$ $C_{1,2}$= $=C_{0,3}$	$A_{1,2}$= $=A_{1,2}$ $B_{1,3}$= $=B_{1,2}$ $C_{2,2}$= $=C_{1,3}$	$A_{2,2}$= $=A_{2,2}$ $B_{2,3}$= $=B_{2,2}$ $C_{3,2}$= $=C_{2,3}$
$A_{0,1}$= $=A_{0,1}$ $B_{0,2}$= $=B_{0,1}$ $C_{1,1}$= $=C_{0,2}$	$A_{1,1}$= $=A_{1,1}$ $B_{1,2}$= $=B_{1,1}$ $C_{2,1}$= $=C_{1,2}$	$A_{2,1}$= $=A_{2,1}$ $B_{2,2}$= $=B_{2,1}$ $C_{3,1}$= $=C_{2,2}$
$A_{0,0}$= $=A_{0,0}$ $B_{0,1}$= $=B_{0,0}$ $C_{1,0}$= $=C_{0,1}$	$A_{1,0}$= $=A_{1,0}$ $B_{1,1}$= $=B_{1,0}$ $C_{2,0}$= $=C_{1,1}$	$A_{2,0}$= $=A_{2,0}$ $B_{2,1}$= $=B_{2,0}$ $C_{3,0}$= $=C_{2,1}$

j

i

Example of Distance Vectors

```
for (i=0; i<n; i++)
  for (j=0; j<m; j++){
     A[i,j] =   ;
        = A[i,j];
     B[i,j+1] =   ;
        = B[i,j];
     C[i+1,j] =   ;
        = C[i,j+1] ;
  }
```

$A_{0,2}$= $=A_{0,2}$ $B_{0,3}$= $=B_{0,2}$ $C_{1,2}$= $=C_{0,3}$	$A_{1,2}$= $=A_{1,2}$ $B_{1,3}$= $=B_{1,2}$ $C_{2,2}$= $=C_{1,3}$	$A_{2,2}$= $=A_{2,2}$ $B_{2,3}$= $=B_{2,2}$ $C_{3,2}$= $=C_{2,3}$
$A_{0,1}$= $=A_{0,1}$ $B_{0,2}$= $=B_{0,1}$ $C_{1,1}$= $=C_{0,2}$	$A_{1,1}$= $=A_{1,1}$ $B_{1,2}$= $=B_{1,1}$ $C_{2,1}$= $=C_{1,2}$	$A_{2,1}$= $=A_{2,1}$ $B_{2,2}$= $=B_{2,1}$ $C_{3,1}$= $=C_{2,2}$
$A_{0,0}$= $=A_{0,0}$ $B_{0,1}$= $=B_{0,0}$ $C_{1,0}$= $=C_{0,1}$	$A_{1,0}$= $=A_{1,0}$ $B_{1,1}$= $=B_{1,0}$ $C_{2,0}$= $=C_{1,1}$	$A_{2,0}$= $=A_{2,0}$ $B_{2,1}$= $=B_{2,0}$ $C_{3,0}$= $=C_{2,1}$

j

i

A yields: $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ B yields: $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ C yields: $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$

FIR Distance Vectors

```
for (i=0; i<n; i++)
  for (j=0; j<m; j++){
    Y[i]=Y[i]+X[i+j]*W[j];
```

Y yields: δ^a $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Y yields: δ^f $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

X yields: δ^i $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$

W yields: δ^j $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Y_0= $=Y_0$ $=X_3$ $=W_3$	Y_1= $=Y_1$ $=X_4$ $=W_3$	Y_2= $=Y_2$ $=X_5$ $=W_3$	Y_3= $=Y_3$ $=X_6$ $=W_3$
Y_0= $=Y_0$ $=X_2$ $=W_2$	Y_1= $=Y_1$ $=X_3$ $=W_2$	Y_2= $=Y_2$ $=X_4$ $=W_2$	Y_3= $=Y_3$ $=X_5$ $=W_2$
Y_0= $=Y_0$ $=X_1$ $=W_1$	Y_1= $=Y_1$ $=X_2$ $=W_1$	Y_2= $=Y_2$ $=X_3$ $=W_1$	Y_3= $=Y_3$ $=X_4$ $=W_1$
Y_0= $=Y_0$ $=X_0$ $=W_0$	Y_1= $=Y_1$ $=X_1$ $=W_0$	Y_2= $=Y_2$ $=X_2$ $=W_0$	Y_3= $=Y_3$ $=X_3$ $=W_0$

j

i

Relabel/Pipeline Variables

Remove anti-dependencies and input dependencies by relabeling, or, pipelining variables.

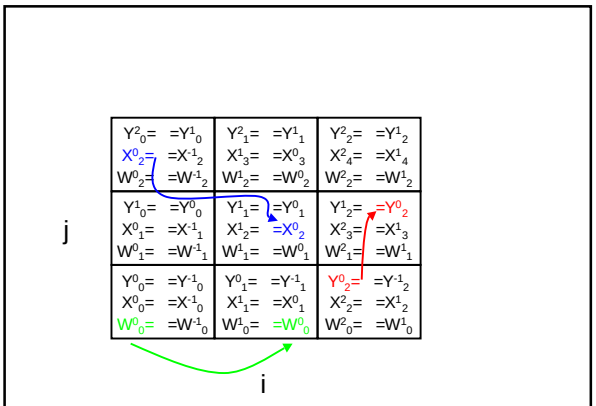
```
for (i=0; i<n; i++)
 for (j=0; j<m; j++){
  W^i[j]=W^i-1[j];
  X^i[i+j]=X^i-1[i+j]
  Y^j[i]=Y^j-1[i]+X^i[i+j]*W^i[j];
}
```

Creates new flow dependencies.
Removes anti/input dependencies

Fir Dependencies

```
for (i=0; i<n; i++)
 for (j=0; j<m; j++){
  W^i[j]=W^i-1[j];
  X^i[i+j]=X^i-1[i+j]
  Y^j[i]=Y^j-1[i]+X^i[i+j]*W^i[j];
}
```

$$D = \begin{matrix} Y & W & X \end{matrix} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$



Transforming to time and space

- Using Data Dependencies, find T
  - T is bijection and monotonic
  - Since T is a bijection, the volume of the iteration space remains constant
  - Since T is monotonic, it respects data dependencies.
- T defines a mapping of the iteration space into a time component, π, and a space component, S.

$$T = \begin{pmatrix} \pi \\ S \end{pmatrix}$$

Meaning of π

$$T = \begin{pmatrix} \pi \\ S \end{pmatrix}$$

- First k rows of T are π. Maps k loops to time.
- if $\pi\mathbf{I}_1 = \pi\mathbf{I}_2$ then $\mathbf{I}_1$ and $\mathbf{I}_2$ execute at the same time.
- Thus, $\pi\mathbf{d} > 0$. This says that it takes $\pi\mathbf{d}$ time units to move data item to where it is needed.
- Total Running Time: max(π**I**)-min(π**I**)
- Since π has k rows, running time is $O(N^k)$.
- Design goal: keep k small.

Meaning of S

$$T = \begin{pmatrix} \pi \\ S \end{pmatrix}$$

- Last n-k rows are S.
- Maps n-k rows in space.
- Since $\pi\mathbf{d} > 0$ we can pick anything for S that makes T a bijection. (I.e, All rows of T must be linearly independent.)

π for FIR

$$D = \begin{matrix} Y & W & X \end{matrix} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$

Say $\pi = [p_1\ p_2]$. **Since $\pi\mathbf{d} > 0$** we can see from the data dependencies that

$p_2 \neq 0$ (from Y)

$p_1 \neq 0$ (from W)

and $p_1 > p_2$ (from X)

To get fastest runtime pick p1 and p2 to be smallest integers possible. Thus, $\pi = [2\ 1]$

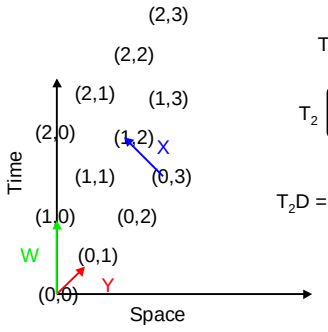
T for fir

$$T_1 = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \quad T_1 \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 2i+j \\ i \end{pmatrix} \quad T_1D = \begin{matrix} Y & W & X \end{matrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$T_2 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \quad T_2 \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 2i+j \\ j \end{pmatrix} \quad T_2D = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$

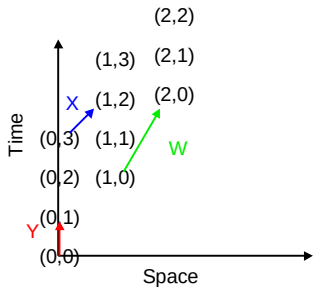
$$T_3 = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \quad T_2 \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 2i+j \\ i+j \end{pmatrix} \quad T_3D = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

T_2



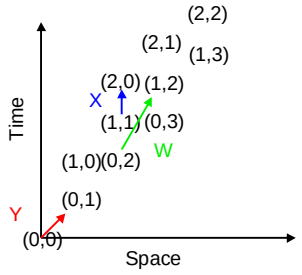
$$T_2 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \quad T_2 \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 2i+j \\ j \end{pmatrix} \quad T_2D = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$

T_1



$$T_1 = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \quad T_1 \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 2i+j \\ i \end{pmatrix} \quad T_1D = \begin{matrix} Y & W & X \end{matrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

T_3



$$T_3 = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \quad T_2 \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 2i+j \\ i+j \end{pmatrix} \quad T_3D = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

T_3 is unimodular

How about Double Pipeline of X?

$$X: \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$Y: \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$W: \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Needs other transforms

```
for (i=0; i<n; i++)
  for (j=0; j<m; j++){
    Y[i]=Y[i]+X[i+j]*W[j];
}

M=m-1;
for (i=0; i<n; i++)
  for (j=0; j<m; j++){
    Y[i]=Y[i]+X[i+M-j]*W[M-j];
}
```

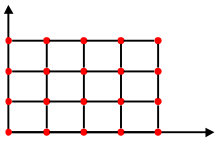
j

Y_0= =Y_0 =X_0 =W_0	Y_1= =Y_1 =X_1 =W_0	Y_2= =Y_2 =X_2 =W_0	Y_3= =Y_3 =X_3 =W_0
Y_0= =Y_0 =X_1 =W_1	Y_1= =Y_1 =X_2 =W_1	Y_2= =Y_2 =X_3 =W_1	Y_3= =Y_3 =X_4 =W_1
Y_0= =Y_0 =X_2 =W_2	Y_1= =Y_1 =X_3 =W_2	Y_2= =Y_2 =X_4 =W_2	Y_3= =Y_3 =X_5 =W_2
Y_0= =Y_0 =X_3 =W_3	Y_1= =Y_1 =X_4 =W_3	Y_2= =Y_2 =X_5 =W_3	Y_3= =Y_3 =X_6 =W_3

i

Describing a Fabric

- If we know the PE/interconnect structure we can determine S.
- Processors described by point in $\mathbf{J}^n$.
- Interconnect described by **P**. Each vector in P is a connection between j and j+p



$J^2=\{(j_1,j_2): 0\le j_1\le 4,\ 0\le j_2\le 3\}$

$$P = \begin{pmatrix} 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

Constraints on S

- SD = PK
- S is space Transformation
- D is dependence matrix
- P is interconnect description
- K is utilization matrix
  - $K_{ji} \ge 0$ Either utilize of don't
  - $-\sum_j k_{ji} \le \pi \mathbf{d}_i$ Make sure data arrives in time.

Find S for FIR

$$\underset{S}{\begin{pmatrix} & \end{pmatrix}}\underset{D}{\begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}} = \underset{P}{\begin{pmatrix} 0 & 1 & -1 \end{pmatrix}}\underset{K}{\begin{pmatrix} & \\ & \\ & \end{pmatrix}}$$

$\pi D = \begin{matrix} 1 & 1 & 2 \end{matrix}$