An Overview to

Polyhedral Model

Fangzhou Jiao

Polyhedral Model

- A framework for performing loop transformation
- Loop representation: using polytopes to achieve fine-grain representation of program
- Loop transformation: transforming loop by doing affine transformation on polytopes
- Dependency test: several mathematical methods for validating transformation on loop polytopes
- Code generation: generate transformed code from loop polytopes

Benefits?

- Fine-grained representation of program
- Dependency Graph:
  - Each node corresponds with one statement in source program
  - Syntax based

Benefits?

- Fine-grained representation of program
- Dependency Graph:
  - Each node corresponds with one statement in source program
  - Syntax based
- Polyhedral Model:
  - Each point in polytope corresponds with one instance of statement
  - Finer grained analysis and transformation is possible

Program Abstraction Level

- Statement

```
For (I=1;I<=10;I++)
  A[I] = A[I-1] + 1
```

- Operation (Instance of statement)

```
A[4] = A[3] + 1
```

Iteration Domain

- Iteration Vector
  - A n-level loop nest can be represented as a n-entry vector, each component corresponding to each level loop iterator

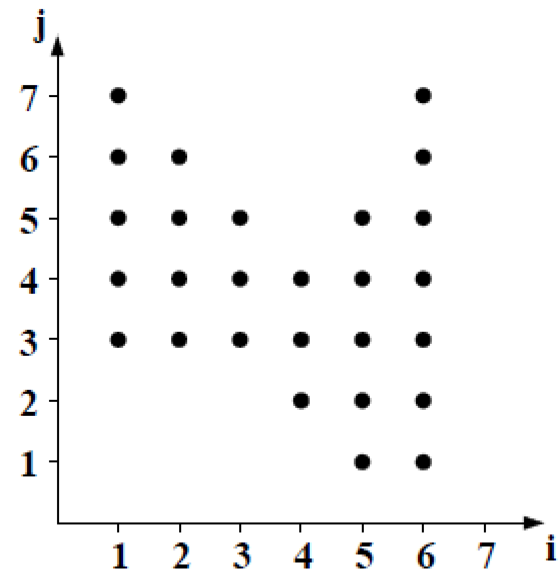
```
For (x1=L1;x1<U1;x1++)
  …
  For (x2=L2;x2<U2;x2++)
    …
    …
    For (xn=Ln;xn<Un;xn++)
      …
```

$$\vec{x} = \begin{pmatrix} x1 \\ x2 \\ . \\ . \\ . \\ xn \end{pmatrix}$$

Iteration Domain

- Iteration Domain: Set of all possible iteration vectors for a given statement

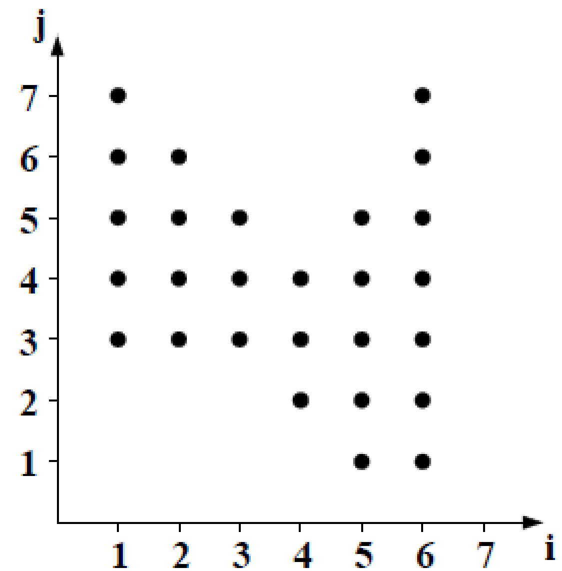
```
For (i=1;i<=6;i++)
  For (j=min(max(6-1,1),3);
       j<=max(8-i,2*i-5);
       j++)
    a[i][j]=a[i-1][j];
```



Iteration Domain

- Iteration Domain: Set of all possible iteration vectors for a given statement

```
For (i=1;i<=6;i++)
  For (j=min(max(6-1,1),3);
        j<=max(8-i,2*i-5);
        j++)
    a[i][j]=a[i-1][j];
```



Notice: This iteration domain is not valid for polyhedral model!

Iteration Domain

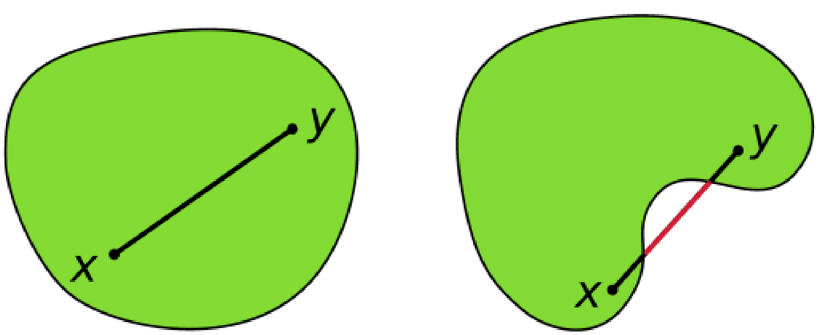
- Iteration domain can be a polytope since it is the set of n-dimension vectors
- For polyhedral model, the iteration domain must be a convex set.

Iteration Domain

- Iteration domain can be a polytope since it is the set of n-dimension vectors
- For polyhedral model, the iteration domain must be a convex set.
- Convex Set:
  - In simple words: For a line segment between any two point in set S, each point on this segment should be in S.

Iteration Domain

- Iteration domain can be a polytope since it is the set of n-dimension vectors
- For polyhedral model, the iteration domain must be a convex set.
- Convex Set:
  - In simple words: For a line segment between any two point in set S, each point on this segment should be in S.



Iteration Domain

- Iteration domain can be a polytope since it is the set of n-dimension vectors
- For polyhedral model, the iteration domain must be a convex set.
- Convex Set:
  - In simple words: For a line segment between any two point in set S, each point on this segment should be in S.
- $\mathbb{Z}$-Polyhedron
  - In most situation loop counters are integers
  - So we use a polyhedron of integer points to represent loop iteration domain

Modeling Iteration Domains

- Dimension of Iteration Domain: Decided by loop nesting levels
- Bounds of Iteration Domain: Decided by loop bounds
  - Using inequalities

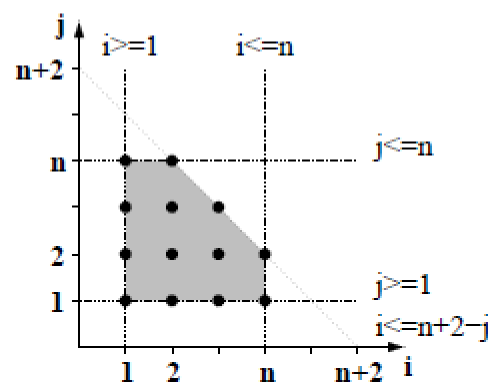
```
For (i=1;i<=n;i++)
 For (j=1;j<=n;j++)
  if (i<=n+2-j)
   b[j]=b[j]+a[i];
```

Modeling Iteration Domains

- Dimension of Iteration Domain: Decided by loop nesting levels
- Bounds of Iteration Domain: Decided by loop bounds
  - Using inequalities

```
For (i=1;i<=n;i++)
 For (j=1;j<=n;j++)
  if (i<=n+2-j)
   b[j]=b[j]+a[i];
```

$$1 \leq i \leq n\,, 1 \leq j \leq n$$
$$i \leq n+2-j$$



Modeling Iteration Domains

- Representing iteration bounds by affine function:

$$1 \le i \le n \;:\; \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ n \end{pmatrix} \ge 0$$

$$1 \le j \le n \;:\; \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ n \end{pmatrix} \ge 0$$

$$\mathrm{i} \le n + 2 - j \;:\; \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + (n + 2) \ge 0$$

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ n \\ -1 \\ n \\ n + 2 \end{pmatrix} \ge \vec{0}$$

Examples: Iteration Domain

```
For (i=0;i<=N;i++)
 For (j=0;j<=i;j++)
  if (i>=M) a[j]=0;
```

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{bmatrix} 0 \\ N \\ 0 \\ 0 \\ -M \end{bmatrix} \geq \vec{0}$$

Examples: Iteration Domain

```
For (i=0;i<=N;i+=2)
 For (j=0;j<=N;j++)
  if (i%3==1 && j%2==0) A[i]=0
```

- Can this loop be represented in polyhedral model?
- The if statement can cause "cavities" in polyhedral.

Examples: Iteration Domain

```
For (i=0;i<=N;i+=2)
 For (j=0;j<=N;j++)
  if (i%3==1 && j%2==0) A[i]=0
```

- Can this loop be represented in polyhedral model?
- The if statement can cause “cavities” in polyhedral.
- Use loop normalization

Loop Normalization

- **Algorithm:**

1. Replace loop boundaries and steps:

   DO I=L,U,S -> DO i=1,(U-L+S)/S,1

2. Replace each reference to original loop variable I with:

   i*S-S+L

3. Reset the loop variable value to ensure the after loop reference to loop variable can get correct value:

   I = i*S-S+L

Example

```
For (i=0;i<=N;i+=2)
 For (j=0;j<=N;j++)
  if (i%3==1 && j%2==0) A[i]=0
```

Example

```
For (i=0;i<=N;i+=2)
 For (j=0;j<=N;j++)
  if (i%3==1 && j%2==0) A[i]=0

For (i=4;i<=N;i+=6)
 For (j=0;j<=N;j+=2)
  A[i]=0
```

Example

```
For (i=0;i<=N;i+=2)
 For (j=0;j<=N;j++)
  if (i%3==1 && j%2==0) A[i]=0


For (i=4;i<=N;i+=6)
 For (j=0;j<=N;j+=2)
  A[i]=0


For (ii=1;ii<=(N+2)/6;ii++)
 For (jj=1;jj<=(N+2)/2;jj++)
  i=ii*6-6+4
  j=jj*2-2
  A[i]=0
```

Example

```
For (ii=1;ii<=(N+2)/6;ii++)
 For (jj=1;jj<=(N+2)/2;jj++)
  i=ii*6-6+4
  j=jj*2-2
  A[i]=0
```

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{bmatrix} 1 \\ (N+2)/6 \\ 1 \\ (N+2)/2 \end{bmatrix} \geq \vec{0}$$

Review: Dependency

- There exists a data dependency from statement s_1 to s_2 if and only if:
  - s_1 and s_2 access to same memory location and at least one of them stores into it
  - A feasible execution path exists from s_1 to s_2

- These rules can be extended to polyhedral model.

Review: Dependency

- There exists a data dependency from statement s_1 to s_2 if and only if:
  - s_1 and s_2 access to same memory location and at least one of them stores into it
  - A feasible execution path exists from s_1 to s_2

- These rules can be extended to polyhedral model.
- Dependency Polyhedral:
  - Array reference function: indicating reference to same memory
  - Iteration domain
  - Precedence order: indicating execution path

Dependency Polyhedral

- Array reference function:
  - For statement s and r accessing same array:
  - $F_s\overrightarrow{x_s} + \overrightarrow{a_s} = F_r\overrightarrow{x_r} + \overrightarrow{a_r}$

Dependency Polyhedral

- **Array reference function:**
  - For statement s and r accessing same array:
  - $F_s\overrightarrow{x_s} + \overrightarrow{a_s} = F_r\overrightarrow{x_r} + \overrightarrow{a_r}$

- **Precedence order function:**
  - Statement s is textually before statement r:
  - $P_s\overrightarrow{x_s} - P_r\overrightarrow{x_r} + \vec{b} \geq \vec{0}$

Construction Dependency Polyhedral

- The dependence polyhedron for $R\delta S$ at a given level i and for a given pair of references to statement r and s is described as Cartesian product of:

$$\begin{bmatrix} F_s & -F_r \\ D_s & 0 \\ 0 & D_r \\ P_s & -P_r \end{bmatrix} \begin{pmatrix} \overrightarrow{x_s} \\ \overrightarrow{x_r} \end{pmatrix} + \begin{pmatrix} a_s - a_r \\ c_s \\ c_r \\ b \end{pmatrix} \begin{matrix} = & 0 \\ \geq & \vec{0} \end{matrix}$$

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]
```

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

Iteration Domain:

$$\mathcal{D}S_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} \geq \vec{0}$$

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

Array Reference Function:

$$F_A(\overrightarrow{x_{s1}}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix}$$

$$F_{A'}(\overrightarrow{x_{s1}}) = \left[\begin{array}{ccc|c|} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{array}\right] \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix}$$

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

Precedence Order:

For statement `S1` in two consecutive loop, `i-i'=1 ,j-j'=1`

$$P_{s1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix}$$

To satisfy $P_s\overrightarrow{x_s} - P_r\overrightarrow{x_r} + \vec{b} \geq \vec{0}$, $\vec{b}$ should be $[-1 \quad -1]$.

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

$$\left[\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right] \begin{pmatrix} i \\ j \\ i' \\ j' \end{pmatrix} + \left(\begin{array}{c} 1 \\ 1 \\ \hline 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ -1 \\ -1 \end{array}\right) \begin{array}{c} = \\ \geq \end{array} \vec{0}$$

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

F: Array Reference Function

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \\ i' \\ j' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \hline 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ -1 \\ -1 \end{pmatrix} \begin{matrix} = \\ \geq \end{matrix} \vec{0}$$

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

A: Iteration Domain

$$\left[\begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right] \begin{pmatrix} i \\ j \\ i' \\ j' \end{pmatrix} + \left(\begin{array}{c} 1 \\ 1 \\ \hline 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ -1 \\ -1 \end{array}\right) \begin{array}{c} = \\ \geq \end{array} \vec{0}$$

Examples for Dependency Polyhedron

```
For (i=0;i<=N;i++)
 For (j=0;j<=N;j++)
  A[i][j]=A[i+1][j+1]   (S1)
```

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \\ i' \\ j' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \hline 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ 0 \\ n \\ -1 \\ -1 \end{pmatrix} \begin{matrix} = \\ \geq \end{matrix} \vec{0}$$

P: Precedence Order

Examples for Dependency Polyhedron

- Matrix format using in polyhedral compiling library:

Given $\mathcal{D}_{R,S}$: $\left[\begin{array}{ccccc} 1 & -1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \end{array}\right] \cdot \begin{pmatrix} i_R \\ i_S \\ j_S \\ n \\ 1 \end{pmatrix} \begin{array}{c} =0 \\ \hline \geq \vec{0} \end{array}$

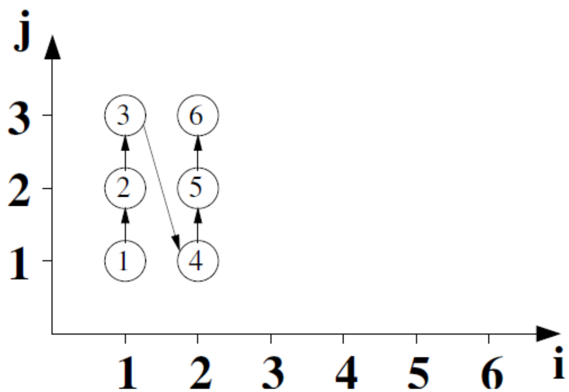
It is written: $\left[\begin{array}{cccccc} 0 & 1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 & 0 \end{array}\right] \cdot \begin{pmatrix} i_R \\ i_S \\ j_S \\ n \\ 1 \end{pmatrix}$

On the first column, 0 stands for $= 0$, 1 for ≥ 0

Transformation using Polytopes: Loop Interchange

Before:

```
For (i=1;i<=2;i++)
  For (j=1;j<=3;j++)
```



$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

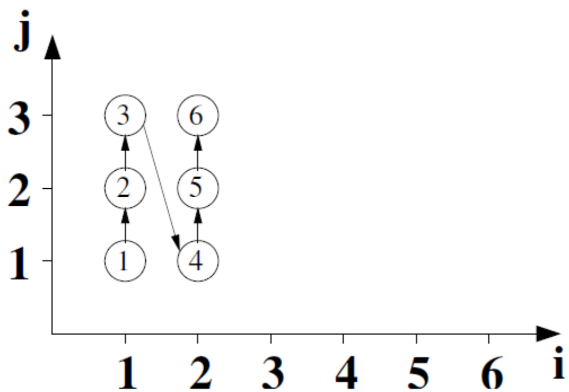
Transformation using Polytopes: Loop Interchange

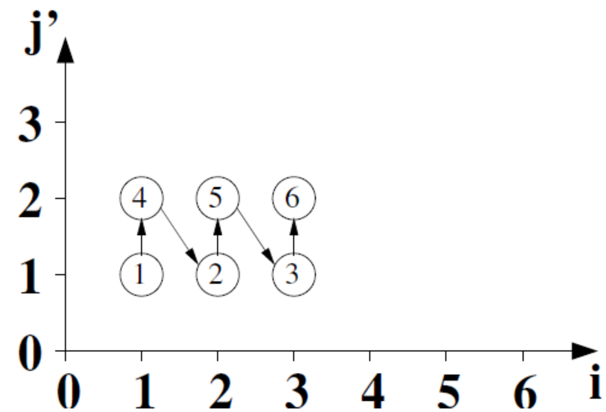
Before:

```
For (i=1;i<=2;i++)
  For (j=1;j<=3;j++)
```

After:

```
For (j=1;j<=3;j++)
  For (i=1;i<=2;i++)
```





$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

$$\begin{bmatrix} 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

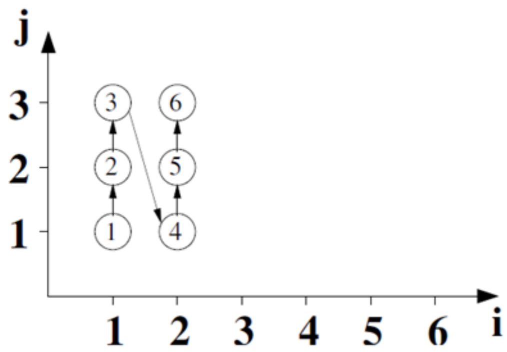
- Transformation Function

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

Transformation using Polytopes: Loop Reversal

Before:

```
For (i=1;i<=2;i++)
  For (j=1;j<=3;j++)
```

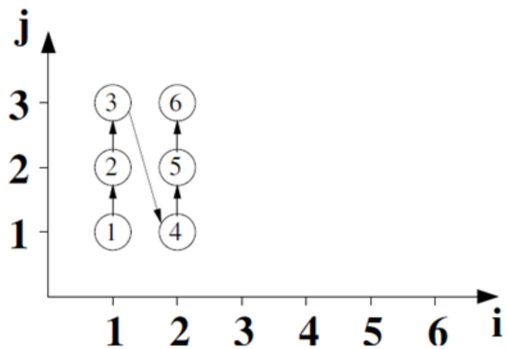


$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

Transformation using Polytopes: Loop Reversal

Before:

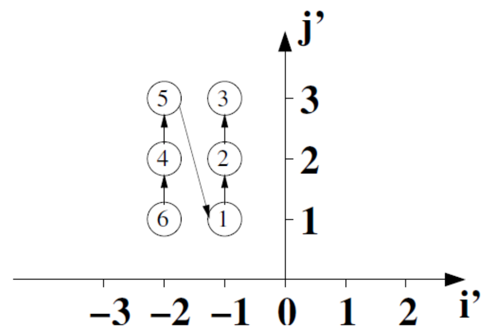
```
For (i=1;i<=2;i++)
  For (j=1;j<=3;j++)
```



$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

After:

```
For (i=-1;i>=-2;i--)
  For (j=1;j<=3;j++)
```



$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

- Transformation Function:

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

Polyhedral Model: Pros and Cons

- Pros:
  - Finer grained representation, analysis, optimization
  - Especially appropriate for loop transformation

Example of Loop Tiling

- (From the C-to-CUDA paper, this tiling intends to improve locality)

```
for (i=0;i<N;i++) {
  P: x[i]=0;
  for (j=0;j<N;j++)
    Q: x[i]+=a[j][i]*y[j];
}
```

(a) Original code

```
for ( it=0; it<=floord(N-1,32);it++) {
  for ( jt=0; jt<=floord(N-1,32);jt++) {
    if ( jt == 0) {
        for ( i=max(32*it,0);
    i<=min(32*it+31,N-1); i++) {
          P: x[i]=0;
          Q: x[i]=x[i]+a[0][i]*y[0];
      }
    }
    for ( i=max(32*it,0);
     i<=min(32*it+31,N-1); i++) {
      for ( j=max(32*jt,1);
       j<=min(32*jt+31,N-1);j++) {
        Q: x[i]=x[i]+a[j][i]*y[j];
      }
    }
  }
}
```

(b) Tiled code

$$D_Q^{orig} \cdot \begin{pmatrix} i \\ j \\ N \\ 1 \end{pmatrix} \geq 0 \qquad D_Q^{tiled} \cdot \begin{pmatrix} it \\ jt \\ i \\ j \\ N \\ 1 \end{pmatrix} \geq 0$$

(c) Original and tiled iteration space

Polyhedral Model: Pros and Cons

- Cons:
  - Efficiency: Compile-time efficiency, since integer programming is NP-complete
  - Building polyhedrons in compile time is also memory consuming

References

- C. Bastoul. *Improving Data Locality in Static Control Programs*. PhD thesis, University Paris 6, Pierre et Marie Curie, France, Dec. 2004. (Recommended, easier to read)
- C. Bastoul. *Code generation in the polyhedral model is easier than you think.* In PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques, pages 7-16, Sept. 2004.
- http://www.cse.ohio-state.edu/~pouchet/#lectures (Strongly Recommended!)
- Paul Feautrier, *Dataflow Analysis of Scalar Array References*, International Journal of Parallel Programming, Vol 20, No.1, 1991
- K Trifunovic, A Cohen, D Edelsohn, L Feng, etc. *GRAPHITE Two Years After: First Lessons Learned From Real-World Polyhedral Compilation.* 2nd International Workshop on GCC Research Opportunities. 2010
- M Baskaran, J Ramanujam, P Sadayappan. *Automatic C-to-CUDA code generation for affine programs*. International Conference on Compiler Construction, 2010

Thanks!