

## Code Generation Using Fourier Motzkin

### Previously

- Data dependences and loops
- Loop transformations
- Unimodular transformation framework
- Kelly and Pugh transformation framework (affine transformations per statement)

### Today

- Code generation
  - use Fourier Motzkin to calculate new loop bounds
  - review array access transformations
  - review loop bounds transformations

## Code Generation

### Goals

- express outermost loop bounds in terms of symbolic constants and constants
- express inner loop bounds in terms of any enclosing loop variables, symbolic constants, and constants

### Approach

- Project out inner loop iteration variables to determine loop bounds for outer loops
- Fourier Motzkin elimination is the algorithm that projects a variable out of a polyhedron

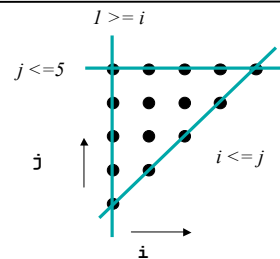
## Fourier-Motzkin Elimination: The Idea

### Polyhedron

- convex intersection of a set of inequalities
- model for iteration spaces

### Problem

- given a polyhedron how do we generate loop bounds that scan all of its points?
- example: two possible loop orders
  - (i, j)
  - (j, i)



## Fourier-Motzkin Elimination: The Algorithm

### FM( $P, i_k$ ) $\Rightarrow P'$

Input:  $P = \{(i_1, i_2, \dots, i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$   
 $i_k$  such that  $1 \leq k \leq d$

Output:  $P' = \{(i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_d) \mid Q'\vec{i}' \geq (\vec{q}' + B'\vec{p}')\}$

### Algorithm:

for each lower bound of  $i_k, (L \leq c_1 i_k)$   
 $P = P - \{L \leq c_1 i_k\}$   
 for each upper bound of  $i_k, (c_2 i_k \leq U)$   
 $P = P - \{c_2 i_k \leq U\}$   
 $P' = P' \cup \{c_2 L \leq c_1 U\}$

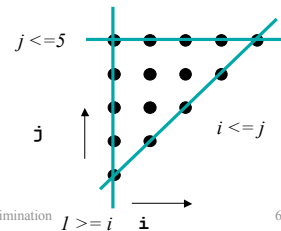
## Distinguishing Upper and Lower Bounds

### Simple Algorithm

- given that the polyhedron is represented as follows:

$$P = \{(i_1, i_2, \dots, i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$$

- any constraint with a positive coefficient for  $i_k$  is a lower bound
- any constraint with a negative coefficient for  $i_k$  is an upper bound



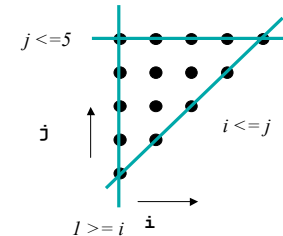
CS553 Lecture

Fourier-Motzkin Elimination

6

## Triangular Iteration Space Example

(i, j) for target iteration space



(j, i) for target iteration space

CS553 Lecture

Fourier-Motzkin Elimination

7

## General Algorithm for Generating Loop Bounds

Input:  $P = \{(i_1, i_2, \dots, i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$   
 where the  $i$  vector is the desired loop order

Output:  $L_{i_1}, L_{i_2}, \dots, L_{i_d}$  such that  $L_{i_k} = f(i_1, \dots, i_{k-1})$   
 $U_{i_1}, U_{i_2}, \dots, U_{i_d}$  such that  $U_{i_k} = g(i_1, \dots, i_{k-1})$

Algorithm:

$$P_n = P$$

for  $k = d$  to 1 by -1

$L_{i_k} =$  all lower bounds for  $i_k$  in  $P_k$

$U_{i_k} =$  all upper bounds for  $i_k$  in  $P_k$

$$P_{k-1} = FM(P_k, i_k)$$

CS553 Lecture

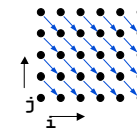
Fourier-Motzkin Elimination

8

## Loop Skewing and Permutation

Original code

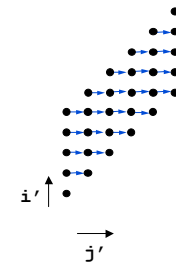
```
do i = 1, 6
  do j = 1, 5
    A(i, j) = A(i-1, j+1) + 1
  enddo
enddo
```



Distance vector: (1, -1)

Skewing followed by Permutation:

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



CS553 Lecture

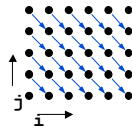
Fourier-Motzkin Elimination

9

## Transforming the Dependences and Array Accesses

### Original code

```
do i = 1,6
  do j = 1,5
    A(i,j) = A(i-1,j+1)+1
  enddo
enddo
```



### Dependence vector:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

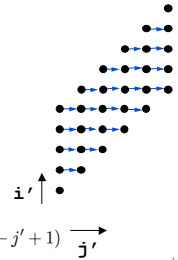
### New Array Accesses:

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = A(i, j)$$

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = A(j', i' - j')$$

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = A(i-1, j+1)$$

$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = A(j'-1, i'-j'+1)$$



CS553 Lecture

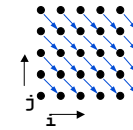
Fourier-Motzkin Elimination

10

## Transforming the Loop Bounds

### Original code

```
do i = 1,6
  do j = 1,5
    A(i,j) = A(i-1,j+1)+1
  enddo
enddo
```

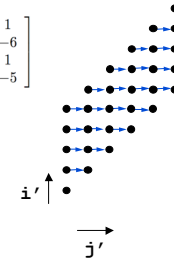


### Bounds:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \geq \begin{bmatrix} 1 \\ -6 \\ 1 \\ -5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} \geq \begin{bmatrix} 1 \\ -6 \\ 1 \\ -5 \end{bmatrix}$$

### Transformed code (use general loop bound alg)

```
do i' = 2,11
  do j' = max(i'-5,1), min(6,i'-1)
    A(j',i'-j') = A(j'-1,i'-j'+1)+1
  enddo
enddo
```



CS553 Lecture

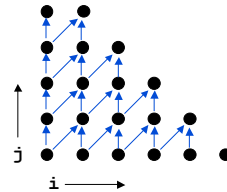
Fourier-Motzkin Elimination

11

## Wavefront Parallelism Example

### Example

```
do i = 1,6
  do j = 1,min(5,7-i)
    A(i,j) = A(i-1,j-1)
    + A(i,j-1)
  enddo
enddo
```



Iteration Space

### Goal

- Determine a unimodular transformation that enables indicating that the inner loop is fully parallel. (with an OpenMP directive for example)

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

```
do i' = 1,5
  do j' = 1, 7-i' (parallel)
    A(j',i') = A(j'-1,i'-1)
    + A(j',i'-1)
  enddo
enddo
```

CS553 Lecture

Fourier-Motzkin Elimination

12

## Concepts

### Fourier-Motzkin Elimination

- algorithm
- using for code generation

### Loop bounds

- how to determine upper and lower bounds for a variable when bounds are in matrix format

### Examples

- triangular matrix
- skew and permute example
- wavefront example

CS553 Lecture

Fourier-Motzkin Elimination

13

## Next Time

---

### Lecture

- Parallelization with no synchronization or communication

### Suggested Exercises

- 11.3.2, 11.3.3, 11.3.4