Fourier Motzkin Elimination

Logistics

–HW10 due Friday April 27th

Today

- Using Fourier-Motzkin elimination for code generation
- Using Fourier-Motzkin elimination for determining schedule constraints

Why Transformation Frameworks?

Currently

- Frameworks used *in compiler* to …
  - abstract loops, memory accesses, and data dependences in loop
  - specify the effect of a sequence of loop transformations on the loop, its memory accesses, and its data dependences
  - generate code from the transformed loop
- Loop transformations affect the *schedule* of the loop

Future

- How can framework technology be exposed in the programming model?

Frameworks

- Unimodular
- Polyhedral
- Presburger
- Sparse Polyhedral

Algorithms needed for automation

Operations on sets and relations

- Union iteration space sets
- Union relations that represent dependences
- Apply a relation to a set to model transforming a loop and to check transformation legality
- Compose two relations to model composing transformations

Scheduling

- Determine an efficient and legal schedule
- Determine which loops should be parallel

Storage Mapping

- If not using UOV, then need to do this in coordination with the scheduling

Code Generation

- Given a schedule and which loops to parallelize and/or tile, generate efficient code
- Code generation for parameterized tiles

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

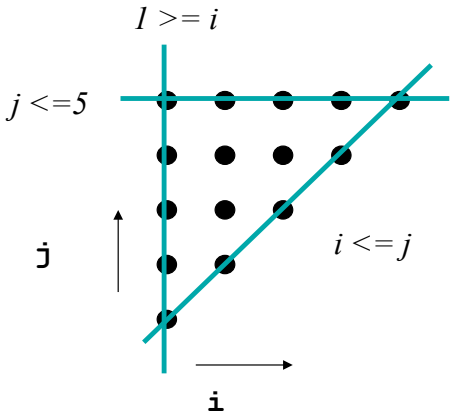
Fourier-Motzkin Elimination: The Idea

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) => P'

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

Output:

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

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 $i_k, (c_2 i_k \leq U)$

$$P = P - \{c_2 i_k \leq U\}$$

$$P' = P' \cup \{c_2 L \leq c_1 U\}$$

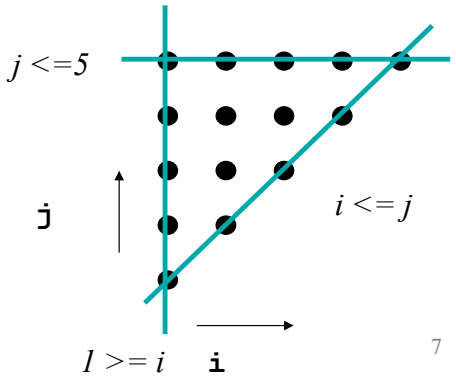
Distinguishing Upper and Lower Bounds

Simple Algorithm

- given that the polyhedron is represented as follows:

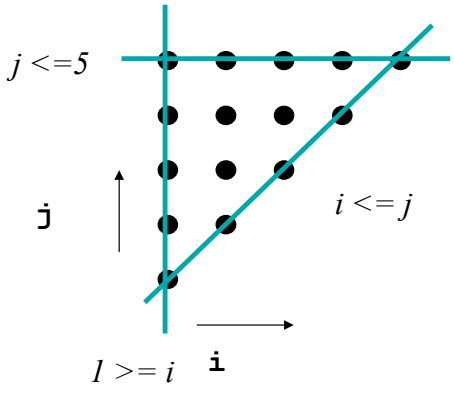
$$P = \{(i_1, i_2, ..., i_d) \mid Q\vec{i} \geq (\vec{q} + B\vec{p})\}$$

- 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



Triangular Iteration Space Example

(i, j) for target iteration space



(j, i) for target iteration space

General Algorithm for Generating Loop Bounds

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

where the i vector is the desired loop order

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

$U_{i_1}, U_{i_2}, ..., L_{i_d}$ such that $U_{i_k} = g(i_1, ...i_{k-1})$

Algorithm:

$P_n = P$

for k = d to 1 by -1

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

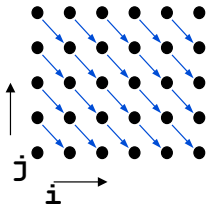
U_{i_k} = all upper bounds for i_k in P_k

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

Loop Skewing and Permutation

Original code

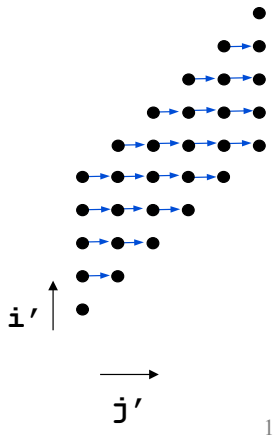
```
do i = 1,6
  do j = 1,5
     A(i,j) = A(i-1,j+1)+1
  enddo
enddo
```



Distance vector: (1, -1)

Skewing followed by Permutation:

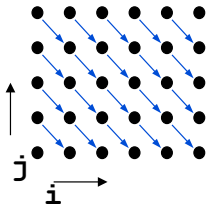
$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i' \\ j' \end{bmatrix}$$



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:

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

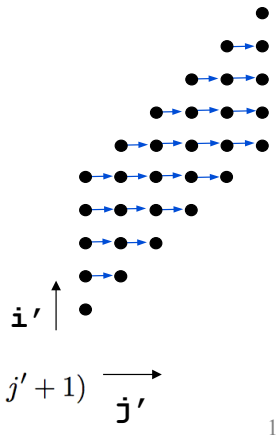
New Array Accesses:

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

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

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

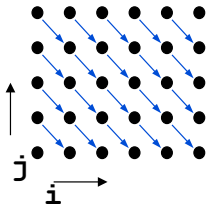
$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = A(j'-1, i'-j'+1)$$



Transforming the Loop Bounds

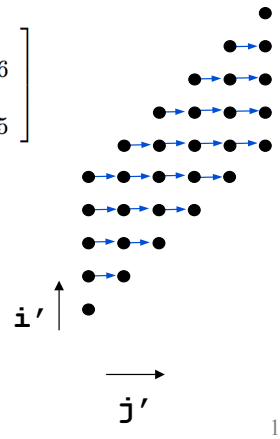
Original code

```
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
```

Fourier Motzkin for Scheduling

Recall

- We need to project out the lambdas
- Now we know how to do that automatically

Using the Farkas lemma

Assume the following dependence polyhedron

$$D_{R\rightarrow S} = \{[\vec{i} \rightarrow \vec{j} \mid A \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix} \geq \vec{0} \text{ and } B \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix} = \vec{0}\}$$

Assume a schedule function of the form

$$\theta_R(\vec{i}) = \vec{v}^T\vec{i} + \vec{b}$$
$$\theta_S(\vec{j}) = \vec{w}^T\vec{j} + \vec{c}$$

We need $\Delta_{R,S} = \theta_S(\vec{i}) - \theta_R(\vec{j}) - 1 >= 0$

The process of determining set of legal schedules

(1) Change all of the equality constraints in $D_{R \to S}$ to inequality constraints.

$$D_{R \to S} = \{[\vec{i} \to \vec{j} \mid A' \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix} \geq \vec{0}\}$$

(2) Use the Farkas lemma to create a set of constraints for the schedule.

$$\theta_S(\vec{j}) - \theta_R(\vec{i}) - 1 = \lambda_0 + \vec{\lambda}^T (A \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{p} \\ 1 \end{bmatrix})$$

$$\lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}$$

$$\theta_R(\vec{i}) = \vec{v}^T \vec{i} + \vec{b}$$

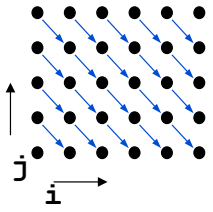
$$\theta_S(\vec{j}) = \vec{w}^T \vec{j} + \vec{c}$$

(3) Solve for v, w, b, and c vector constraints by projecting out lambdas.

Example of using the Farkas lemma

Original code

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



(1) Dependence polyhedron

(2) Farkas lemma to set up constraints

(3) Project out lambdas to determine set of legal schedules

Next Time

Lecture

- Code generation for parameterized tiling

Schedule

- HW10 due Friday April 27th

4/19/12

```
for (i = 0; i <= 5; i++) {
    A[i] = A[i-1] + 42;
}
```

$$D = \{ [i] \to [i'] \mid i = i' - 2 \quad 0 \le i \le 5 \quad 0 \le i' \le 5 \}$$

$$\theta(x) = a x + b$$

$$\theta(i') - \theta(i) - 1 \geq 0$$

$$\theta(i') - \theta(i) - 1 = \lambda_0 + \lambda_1 (i - i' + 2) + \lambda_2 (-i + i' - 2) + \lambda_3 (i) + \lambda_4 (5 - i) + \lambda_5 (i') + \lambda_6 (5 - i')$$

$$a i' + b - a i - b - 1 \leq \geq$$

$$\lambda_y \geq 0$$

$\{ [a, b] : (\text{exists } i', i, \lambda_0 \ldots :$

$$a i' + b - a i - b - 1 - (\lambda \text{ stuff}) \leq -a i' - b + a i + b + 1 + (\lambda \text{ stuff})$$