Loop Transformations, Dependences, and Parallelization

Announcements

- HW3 is due Wednesday February 15th

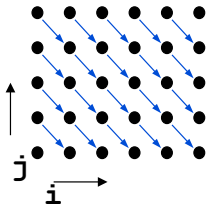
Today

- HW3 intro
- Unimodular framework rehash with edits
- Skewing Smith-Waterman (the fix is in!), composing transformations
- Unimodular transformation framework
  - Affine transformations? (NOPE)
  - Why unimodular matrix?
  - Usage in other research and extensions
  - Loop reversal
- Testing transformation legality
- Converting C++ code to iteration space representation

Iteration Space Representation

Original code

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



Represent the iteration space

–As an intersection of inequalities

–The iteration space is the integer tuples within the intersection

Bounds:

$$\begin{aligned} 1 &<= i \\ i &<= 6 \\ 1 &<= j \\ j &<= 5 \end{aligned}$$

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \leq \begin{bmatrix} -1 \\ 6 \\ -1 \\ 5 \end{bmatrix}$$

Lexicographical Order as Schedule

Iteration point

– Integer tuple with dimensionality d $(i_0, i_1, \ldots, i_d)$

Lexicographical Order

– First order the iteration points by i_0, then i_1, … and finally i_d.

$$(i_0, i_1, ..., i_{d-1}) \prec (i_0, i_1, ..., i_{d-1}) \equiv$$
$$(i_0 < j_0) \vee (i_0 = j_0 \wedge i_1 < j_1) \vee ...(i_0 = j_0 \wedge i_1 = j_1 \wedge ...i_{d-1} = j_{d-1})$$

Frameworks for Loop Transformations

Loop Transformations as functions

$$\vec{i'} = f(\vec{i})$$

Unimodular Loop Transformations [Banerjee 90],[Wolf & Lam 91]

– can represent loop permutation, loop reversal, and loop skewing

– unimodular linear mapping (determinant of matrix is + or - 1)

$$\vec{i'} = T\vec{i}$$

– T is a matrix, i and i' are iteration vectors

– example

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

– limitations

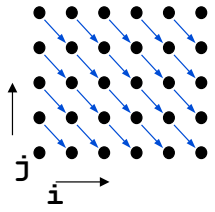
– only perfectly nested loops

– all statements are transformed the same

Loop Skewing

Original code

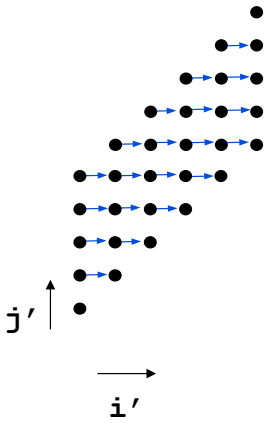
```
do i = 1,6
  do j = 1,5
     A(i,j) = A(i-1,j+1)+1
  enddo
enddo
```



Distance vector: (1, -1)

Skewing:

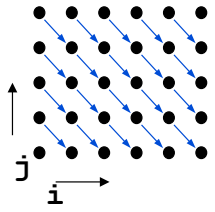
$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i \\ i+j \end{bmatrix}$$



Transforming the Dependences and Array Accesses

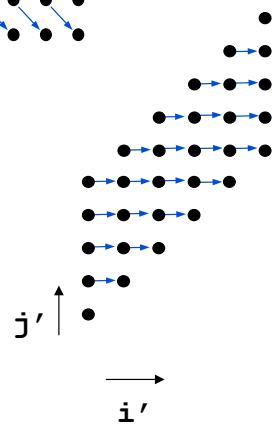
Original code

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



Dependence vector:

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



New Array Accesses:

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

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

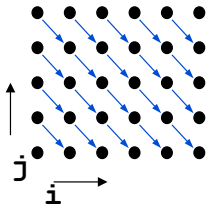
$$A\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = A(i-1, j+1)$$

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

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
```



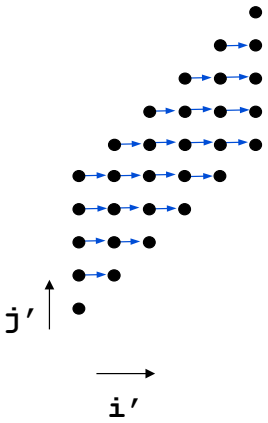
Bounds:

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \leq \begin{bmatrix} -1 \\ 6 \\ -1 \\ 5 \end{bmatrix}$$

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

Transformed code

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



Revisiting (smithWaterman.c) and really parallelizing it!

```
for (i=1;i<=a[0];i++) {
  for (j=1;j<=b[0];j++) {
    diag    = h[i-1][j-1] + sim[a[i]][b[j]];
    down    = h[i-1][j] + DELTA;
    right   = h[i][j-1] + DELTA;
    …
```

Let x=i+j and y=i.

```
for (x=2; x<=a[0]+b[0]; x++) {
   #omp parallel for private(y) shared(a,b)
   for (y=max(1, x-b[0]); y<=min(a[0],x-1); y++) {
     diag    = h[y-1][x-y-1] + sim[a[x]][b[x-y]];
     down    = h[y-1][x-y] + DELTA;
     right   = h[y][x-y-1] + DELTA;
     …
```

Unimodular Framework

Does it have an affine component?

– Actually no. That moves us to the polyhedral model.

Why a unimodular matrix?

– Has an inverse and the inverse is unimodular.

– Preserves the volume of a polytope.

Transformations we can automate

–Loop permutation, skewing, and reversal

–Any combination of the above

Loop Reversal

Idea

– Change the direction of loop iteration

(*i.e.,* From low-to-high indices to high-to-low indices or vice versa)

Benefits

– Could improve cache performance

– Enables other transformations

Example

```
do i = 6,1,-1
  A(i) = B(i) + C(i)
enddo
```

```
do i = 1,6
   A(i) = B(i) + C(i)
enddo
```

Loop Reversal and Distance Vectors

Impact

- Reversal of loop *i* negates the i^{th} entry of all distance vectors associated with the loop
- What about direction vectors?

When is reversal legal?

- When the loop being reversed does not carry a dependence
  (*i.e.*, When the transformed distance vectors remain legal)

Example

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

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

Dependence: Flow
Distance Vector: (1,1)
Transformed
Distance Vector: (1,-1) legal

Transformation Legality

Recall …

- A dependence vector is legal if it is lexicographically non-negative.
- Applying the transformation function to each dependence vector produces a dependence vector for the new iteration space.

When is a transformation legal assuming a lexicographical schedule?

What about parallelism?

Converting C loops to iteration space representation

Analyses needed

- Loop analysis
  - Loop bounds from AST or control-flow graph
  - Induction variable detection
- Pointer analysis
  - Do pointers point at same or overlapping memory?
  - Note that in C can cast a pointer to an integer and back and can do pointer arithmetic.
  - In general requires whole program analysis.
- Dependence analysis

Is this even possible?

- Current tools make the optimistic pointer assumption
- We need programming models that simplify or remove the need for such analyses

Concepts

smithWaterman.c example

–Projecting out a variable from bounds (prelim to Fourier Motzkin elimination)

Unimodular transformation framework

- represents loop permutation, loop reversal, and loop skewing
- provides mathematical framework for ...
  - testing transformation legality,
  - transforming array accesses and loop bounds,
  - and combining transformations

Compiler analyses needed in C to obtain an iteration space representation

References

[Banerjee90] Uptal Banerjee, “Unimodular transformations of double loops,” In Advances in Languages and Compilers for Parallel Computing, 1990.

[Wolf & Lam 91] Wolf and Lam, “A Data Locality Optimizing Algorithm,” In Programming Languages Design and Implementation, 1991.

Next Time

Reading

- No reading assignments next week

Homework

- HW3 is due Wednesday 2/15/12

Lecture

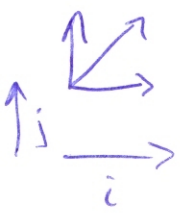
- Polyhedral framework

Smith waterman

dependences

(1,1)
(1,0)
(0,1)
(i,j)

} both loops carry at least on dep



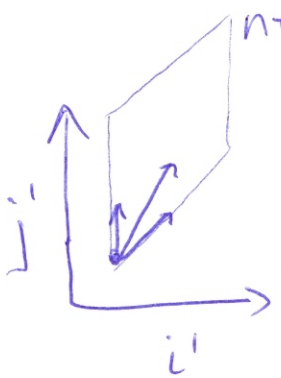
---

let $j' = i + j$ and $i' = i$

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

---

new dependences (i', j')



$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$ i' carries

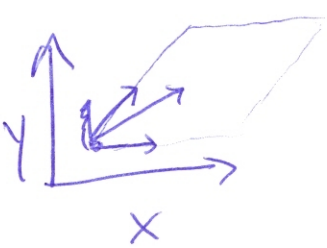
$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$ i' carries

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

---

permute as well as skew

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \quad \begin{matrix} x = i + j \\ y = i \end{matrix}$$



$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$ x carries

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$ x "

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$ x "

what about the bounds?

$1 \le i \le a[0]$ $\quad -i \le -1$

$1 \le j \le b[0]$ $\quad -j \le -1$

in matrix form

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \le \begin{bmatrix} -1 \\ a[0] \\ -1 \\ b[0] \end{bmatrix}$$

replace $\begin{bmatrix} i \\ j \end{bmatrix}$ with $T^{-1}\begin{bmatrix} x \\ y \end{bmatrix}$

$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then $T^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

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

new bounds

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} -1 \\ a[0] \\ -1 \\ b[0] \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 0 & 1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

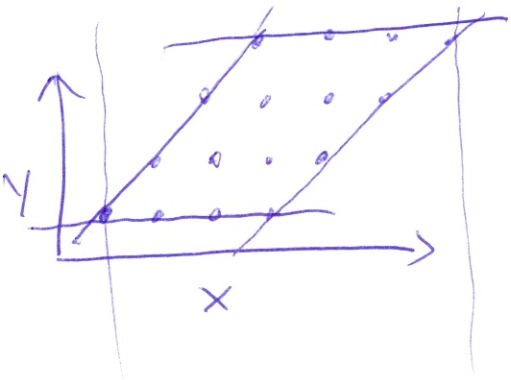
$1 \le y \le a[0]$

$1 \le x - y \le b[$

x is outer loop!

$1 \le y \le a[0]$

$1 \le x - y \le b[0]$



Need x bounds w/out y

lower bounds of y

$1 \le y$

$x - b[0] \le y$

upper bounds for y

$y \le a[0]$

$y \le x - 1$

$1 \le x \le a[0] \Rightarrow 1 \le a[0]$

$1 \le y \le x - 1 \Rightarrow 2 \le x$

$x - b[0] \le a[0] \Rightarrow x \le a[0] + b$

$x - b[0] \le x - 1 \Rightarrow 1 \le b[0]$