The Polyhedral Model (Transformations)

Announcements
– HW4 is due Wednesday February 22th
– Project proposal is due NEXT Friday, extension so that example with tool is possible (see resources website for

Today
– Polyhedral definitions
– Direction/distance vectors and dependence relations
– Polyhedral model (two notations and ways of viewing shared schedule)
  – Multi-dimensional schedule and statement order
  – Mapping with constant elements for statement order (K&P)
  – Legality for both
  – Fusion and fission using K&P

Polyhedron
(source: http://www.cse.ohio-state.edu/~pouchet/lectures/888.11.lect1.html)

Affine functions
– A function $f : \mathbb{K}^m \rightarrow \mathbb{K}^n$ is affine if there exists a vector $\vec{b} \in \mathbb{K}^n$ and a matrix $A \in \mathbb{K}^{n \times m}$ such that $\forall \vec{x} \in \mathbb{K}^m, f(\vec{x}) = A\vec{x} + \vec{b}$

Affine half spaces
– An affine half-space of $\mathbb{K}^m$ (affine constraint) is defined as a set of points $\{ \vec{x} \in \mathbb{K}^m | \vec{a} \cdot \vec{x} \leq \vec{b} \}$

Polyhedron
– A set $S \subseteq \mathbb{K}^m$ is a polyhedron if there exists a system of finite inequalities $A\vec{x} \leq \vec{b}$ such that $P = \{ \vec{x} \in \mathbb{K}^m | A\vec{x} \leq \vec{b} \}$
  – Equivalently it is the intersection of finitely many half-spaces.

Intersection between polyhedral sets
– When you intersect two convex polyhedral sets the results is a convex polyhedral set.

$$ P = \{ \vec{x} \in \mathbb{K}^m | \vec{x} \in P_1 \land \vec{x} \in P_2 \} $$
Convexity for Reals

Reals

– Given \( S \) a subset of \( \mathbb{R}^n \). \( S \) is convex iff, \( \forall \mu, \lambda \in S \) and given \( c \in [0, 1] \)
\[
(1 - c)\mu + c\lambda \in S
\]
– IOW, drawing a line segment between any two points of \( S \), each point on this segment is also in \( S \).

Integers (Z – polyhedron)

– Lattice: A subset \( L \) in \( \mathbb{Z}^n \) is a lattice if it is generated by an integral combination of finitely many vectors:

– If the vectors have integral coordinates, \( L \) is an integer lattice.
– A Z-polyhedron is the intersection of a polyhedron and an affine integral full dimensional lattice.

(source: http://www.cse.ohio-state.edu/~pouchet/lectures/888.11.lect1.html)

Convexity for Integers

Lattice

– Lattice: A subset \( L \) in \( \mathbb{Z}^n \) is a lattice if it is generated by an integral combination of finitely many vectors: \( \vec{a}_1, \vec{a}_2, ..., \vec{a}_n (\vec{a}_i \in \mathbb{K}^n) \)

– If the \( \vec{a}_i \) vectors have integral coordinates, \( L \) is an integer lattice.

Z-polyhedron

– A Z-polyhedron is the intersection of a polyhedron and an affine integral full dimensional lattice.

– Iteration domains are in fact Z-polyhedra with a unit lattice

\[
Q = \{i, j||0 \leq i, j \leq 10\}
\]

\[
L = \{i, j||i, j \in \mathbb{Z}\}
\]

\[
Z = Q \cap L
\]

– Intersection of Z-polyhedra is not convex in general

(source: http://www.cse.ohio-state.edu/~pouchet/lectures/888.11.lect1.html)
Checking whether a set is empty

Many questions we need to automate check whether a polyhedral set or sets are empty or not.

– Is there a dependence at a certain loop level?
– Is a transformation legal?

Determining whether a set is feasible

– If constraints conflict then the set is empty, or infeasible.
– For real polyhedrons there are polynomial time algorithms.
– For Z-polyhedron there are worst-case exponential time algorithms, but these algorithms are reasonable for most problems related to loop transformations.

Dependence Testing in General

General code

\[
\begin{align*}
\text{do } & i_1 = l_1, h_1 \\
& \ldots \\
& \text{do } i_n = l_n, h_n \\
& \ldots \ A(f(i_1, \ldots, i_n)) \\
& \ldots \ A(g(i_1, \ldots, i_n)) \\
\text{enddo} \\
\text{enddo}
\end{align*}
\]

There exists a dependence between iterations \( I=(i_1, \ldots, i_n) \) and \( J=(j_1, \ldots, j_n) \) when at least one of the accesses is a write and

– \( f(I) = g(J) \)
– \( (l_1, \ldots, l_n) < I, J < (h_1, \ldots, h_n) \)
– \( I \ll J \) or \( J \ll I \), where \( \ll \) is lexicographically less
**Direction/Dependence Vectors**

```plaintext
do i = 1,5  
   do j = 1,6  
      A(i+1,j-1) = A(i-2,j+1) + exp(j,j)  
   enddo  
enddo
```

Set up the dependence problem

**Subtract the source from the target iteration**

**Dependence Relations**

```plaintext
do i = 1,5
   do j = 1,6
      A(i+1,j-1) = ...
      ... = ... A(i-2,j+1) + exp(j,j)
   enddo
endo
```

Possible notations

- Feautrier

- Kelly and Pugh
Recall Automatic Parallelization

Input program has a set of operations E with a strict order

Find a partial order on E that is deterministic and results in the same output as the original strict total order.

Overall process
- Translate the code to a model
- Select a transformation/schedule
  - Determination of partial order on E, data dependence analysis
  - Specify a transformation/schedule
  - Ensure that the loop transformation/schedule is legal
- Transform the model and generate the transformed code

Specifying Loop Transformations

Unimodular

\[ f(\vec{i}) = T\vec{i} \]

Polyhedral/Affine Transformations (also called a schedule or Change of Basis)

\[ f(\vec{i}) = T\vec{i} + \vec{j} \]

Kelly and Pugh (Presburger in general case)

\[ \{ \vec{i} \rightarrow \vec{x} | \vec{x} = T\vec{i} + \vec{j} \} \]
Loop Fusion

Idea
– Combine multiple loop nests into one

Example
\[
\begin{align*}
\text{do } i = 1, n & \quad \text{do } i = 1, n \\
A(i) &= A(i-1) & A(i) &= A(i-1) \\
\text{enddo} & \quad \text{endo} \\
\text{do } j = 1, n & \\
B(j) &= A(j)/2 & B(i) &= A(i)/2 \\
\text{enddo} & \quad \text{endo}
\end{align*}
\]

Pros
– May improve data locality
– Reduces loop overhead
– Enables **array contraction** (opposite of scalar expansion)
– May enable better instruction scheduling

Cons
– May hurt data locality
– May hurt icache performance

Legality of Loop Fusion

Basic Conditions
– Both loops must have same structure
  – Same loop depth
  – Same loop bounds
  – Same iteration directions
  \{Can we relax any of these restrictions?\}
– Dependences must be preserved
  \textit{e.g.}, Flow dependences must not become anti dependences

\[
\begin{align*}
\text{do } i = 1, n & \quad \text{do } i = 1, n \\
\text{body1} & \quad \text{body1} \\
\text{endo} & \quad \text{endo} \\
\text{do } i = 1, n & \\
\text{body2} & \\
\text{endo} & \quad \text{body2} \\
\end{align*}
\]

All cross-loop dependences flow from body1 to body2

\[
\begin{align*}
\text{do } i = 1, n & \quad \text{do } i = 1, n \\
\text{body1} & \quad \text{body1} \\
\text{endo} & \quad \text{endo} \\
\text{do } i = 1, n & \\
\text{body2} & \\
\text{endo} & \quad \text{body2} \\
\end{align*}
\]

Ensure that fusion does not introduce dependences from body2 to body1
**Loop Fusion Example**

What are the dependences?

```plaintext
do i = 1,n
  s1  A(i) = B(i) + 1
endo

s1 δf s2

s2 C(i) = A(i)/2
endo

s2 δf s3

s3 D(i) = 1/C(i+1)
endo
```

What are the dependences?

```plaintext
do i = 1,n
  s1  A(i) = B(i) + 1
endo

s1 δf s2

s2 C(i) = A(i)/2
endo

s2 δf s3

s3 D(i) = 1/C(i+1)
endo
```

Fusion changes the dependence between s2 and s3, so fusion is illegal

---

**Loop Fusion Example (cont)**

Loop reversal is legal for the original loops

- Does not change the direction of any dep in the original code
- Will reverse the direction in the fused loop: s3 δa s2 will become s2 δf s3

```plaintext
do i = n,1
  s1  A(i) = B(i) + 1
endo

s1 δf s2

s2 C(i) = A(i)/2
endo

s2 δf s3

s3 D(i) = 1/C(i+1)
endo
```

```plaintext
do i = n,1,-1
  s1  A(i) = B(i) + 1
endo

s1 δf s2

s2 C(i) = A(i)/2
endo

s2 δf s3

s3 D(i) = 1/C(i+1)
endo
```

After reversal and fusion all original dependences are preserved
Kelly and Pugh Transformation Framework

Specify iteration space as a set of integer tuples
\[ \{(i, j) \mid 1 \leq i, j \leq n\} \]

Specify data dependences as relations between integer tuples (i.e., data dependence relations)
\[
\{(i_1, j_1) \rightarrow (i_2, j_2) \mid (i_1 = i_2 - 1) \land (j_1 = j_2 - 1) \land (1 \leq i_1, j_1, i_2, j_2 \leq n) \land i_1 < i_2\} \\
\cap \\
\{(i_1, j_1) \rightarrow (i_2, j_2) \mid (i_1 = i_2 - 1) \land (j_1 = j_2 - 1) \land (1 \leq i_1, j_1, i_2, j_2 \leq n) \land i_1 = i_2 \land j_1 < j_2\}\]

Specify transformations as relations/mappings between integer tuples
\[ \{(i, j) \rightarrow (i', j') \mid (i' = j) \land (j' = i)\} \]

Execute iterations in transformed iteration space in lexicographic order

Specifying Loop Fusion in Kelly and Pugh Framework

Specify iteration space as a set of integer tuples
\[ IS_1 = \{(1, i_1, 1) \mid 1 \leq i_1 \leq n\} \]
\[ IS_2 = \{(2, i_2, 1) \mid 1 \leq i_2 \leq n\} \]
\[ IS_3 = \{(3, i_3, 1) \mid 1 \leq i_3 \leq n\} \]
\[ IS = IS_1 \cup IS_2 \cup IS_3 \]

Specify data dependences as mappings between integer tuples (i.e., data dependence relations)
\[ D_{12} = \{(1, i_1, 1) \rightarrow [2, i_2, 1] \mid i_1 = i_2\} \]
\[ D_{23} = \{(2, i_2, 1) \rightarrow [3, i_3, 1] \mid i_2 = i_3 + 1\} \]
\[ D = D_{12} \cup D_{23} \]

Specify transformations as mappings between integer tuples
\[ T_1 = \{(1, i_1, 1) \rightarrow [1, i'_1, 1] \mid i'_1 = i_1\} \]
\[ T_2 = \{(2, i_2, 1) \rightarrow [1, i'_2, 2] \mid i'_2 = i_2\} \]
\[ T_3 = \{(3, i_3, 1) \rightarrow [1, i'_3, 3] \mid i'_3 = i_3\} \]
\[ T = T_1 \cup T_2 \cup T_3 \]
Checking Legality in Kelly & Pugh Framework

For each dependence, \([I] \rightarrow [J]\) the transformed \(I\) iteration must be executed after the transformed \(J\) iteration.

Loop Fusion Example (cont)

Loop reversal is legal for the original loops
- Does not change the direction of any dep in the original code
- Will reverse the direction in the fused loop: \(s_3\delta^f s_2\) will become \(s_2\delta^f s_3\)

After reversal and fusion all original dependences are preserved
Fusion Example

Can we fuse these loop nests?

```plaintext
do i = 1,n
  X(i) = 0
enddo

do j = 1,n
do k = 1,n
  X(k) = X(k)+A(k,j)*Y(j)
enddo
enddo
```

Fusion of these loops would violate this dependence

```plaintext
do i = 1,n
  X(i) = 0
enddo

do k = 1,n
do j = 1,n
  X(k) = X(k)+A(k,i)*Y(i)
enddo
enddo
```

Fusion Example (cont)

Use loop interchange to preserve dependences

```plaintext
do i = 1,n
  X(i) = 0
enddo

do k = 1,n
do j = 1,n
  X(k) = X(k)+A(k,j)*Y(j)
enddo
enddo
```
### Loop Fission (Loop Distribution)

**Idea**
- Split a loop nest into multiple loop nests (the inverse of fusion)

**Example**
```
  do i = 1,n
    A(i) = B(i) + 1
    C(i) = A(i)/2
  enddo
```

**Motivation?**
- Produces multiple (potentially) less constrained loops
- May improve locality
- Enable other transformations, such as interchange

**Legality?**
```
  do i = 1,n
    body1
    body2
  enddo
```

Cycles cannot be preserved because after fission all cross-loop dependences flow from body1 to body2

### Loop Fission (cont)

**Legality**
- Fission is legal when the loop body contains no cycles in the dependence graph

```
  do i = 1,n
    body1
    body2
  enddo
```

Cycles cannot be preserved because after fission all cross-loop dependences flow from body1 to body2
Loop Fission Example

Recall our fusion example

\[
\begin{align*}
& \text{do } i = 1, n \\
& s_1 \quad A(i) = B(i) + 1 \\
& \quad \text{endo} \quad s_1 \delta^f s_2 \\
& \text{do } i = 1, n \\
& s_2 \quad C(i) = A(i)/2 \\
& \quad \text{endo} \quad s_2 \delta^f s_3 \\
& \text{do } i = 1, n \\
& s_3 \quad D(i) = 1/C(i+1) \\
& \quad \text{endo}
\end{align*}
\]

Can we perform fission on this loop?

\[
\begin{align*}
& \text{do } i = 1, n \\
& s_1 \quad A(i) = B(i) + 1 \\
& \quad \text{endo} \quad s_1 \delta^f s_2 \\
& \text{do } i = 1, n \\
& s_2 \quad C(i) = A(i)/2 \\
& \quad \text{endo} \quad s_2 \delta^a s_2 \\
& \text{do } i = 1, n \\
& s_3 \quad D(i) = 1/C(i+1) \\
& \quad \text{endo}
\end{align*}
\]

Loop Fission Example (cont)

If there are no cycles, we can reorder the loops with a topological sort

\[
\begin{align*}
& \text{do } i = 1, n \\
& s_1 \quad A(i) = B(i) + 1 \\
& \quad \text{endo} \quad s_1 \delta^f s_2 \\
& \text{do } i = 1, n \\
& s_3 \quad D(i) = 1/C(i+1) \\
& \quad \text{endo} \quad s_3 \delta^a s_2 \\
& \text{do } i = 1, n \\
& s_2 \quad C(i) = A(i)/2 \\
& \quad \text{endo}
\end{align*}
\]
Two different ways to deal with imperfect loop nesting

Add dimensionality to the schedule for statement order
– Approach taken by Kelly and Pugh

Embed all statements in same iteration space
– Use initial statement order as the tie breaker

Difference
– Has an effect on the scheduling algorithms
– Has an effect on the code generation algorithms

Next Time

Reading
– No reading assignment this week, should be reading about possible benchmarks and automation tools

Homework
– HW4 is due Wednesday 2/22/12
– Project proposal pushed to next Friday, 2/24/12

Lecture
– Tools for specifying and transforming polyhedra: Omega
do $i = 1$ to 5
    do $j = 1$ to 6
        $A(i+1, j-1) = A(i-2, j+1) + \exp(j)$
    end do
deend

write ($i, j$)  \hspace{1cm} read ($i', j'$)

(1) Set up dependence problem

$i+1 = i'-2 \Rightarrow i = i'-3$

$j-1 = j'+1$

$1 \leq i, i' \leq 5 \hspace{1cm} 1 \leq j, j' \leq 6$

$\sqrt{i < i'} \hspace{1cm} OR \hspace{1cm} i = i' \& j < j'$

(2) $dv = target - source$

$= (i', j') - (i, j)$

$= (3, -2)$

$= (\langle, \rangle)$

what if

$dv = (i'-i, j'-j)$

\hspace{1cm}$i < i'$

\hspace{1cm}$j' < j$

$= (\langle, \rangle)$
\[ x \to y \mid f(x) = g(y) \land x \in D_x \land y \in D_y \land \langle x, y \rangle \land x_0 < y_0 \land x_0 = y_0 \land \exists x, y_1 < y_1 \]

\[ x + y \]

Feautrier \( \langle R, x \rangle \langle <, y \rangle \)

For \( i = 1 \) to 10

\[ R: \]

For \( j = 1 \) to 8

\[ S: \]

Kelly & Pugh

\[ \Theta(i) = [0, i] \]

\[ \Theta(j) = [1, j] \]
\[ D_{12} = \exists [1, i_1, 1] \rightarrow [2, i_2, 1] \mid i_1 = i_2 \]

\[ T_{2} = \exists [2, i_2, 1] \rightarrow [1, i_2', 2] \mid i_2' = i_2 \]

New Dependence Relation

\[ D'_{12} = \exists [1, i_1', 1] \rightarrow [1, i_2', 2] \mid i_1' = i_1 \]

\[ \text{target - source} = [1-1, i_2' - i_1', 2-1] \]

\[ = [0, 0, 1] \]

\[ \downarrow \]

\[ D_{23} = \exists [2, i_2, 1] \rightarrow [3, i_3, 1] \mid i_2 = i_3 + 1 \]

\[ D'_{23} = \exists [1, i_2', 2] \rightarrow [1, i_3', 3] \]

\[ \text{target - source} = [1-1, i_3' - i_2'] \]

\[ = [0, -1] \]

\[ \uparrow \]

not legal
do $i = 1$ to $5$
\[ [0, 0, i, 0, j, 0] \rightarrow [0, i', 0, j', 1] \]
\[ 2 + 1 = i' - 2 \]

$R$: $A(i+1, j-1) = \ldots$

$S$: $\ldots = A(i-2, j+1)$

**Dependence Relations**

**Feautrier**

$R, S$ statements

$\mathbf{r}, \mathbf{y}$ iteration vectors

$g(\cdot), g(\cdot)$ access functions

\[ \exists \langle R, x \rangle, \langle S, y \rangle \mid f(x) = g(y) \]

\[ \forall x \in D_R \]
\[ \forall y \in D_S \]
\[ \forall \langle R, x \rangle \prec_p \langle S, y \rangle \]

\[ \langle R, x \rangle \prec_p \langle S, y \rangle \equiv x_0 = y_0 \land x_1 = y_1 \land \ldots \land x_p = y_p \]