Automating Scheduling

Logistics
– HW8 due tomorrow, April 11th

Previously
– Polyhedral operations using the constraint representation

Today
– Automating scheduling with the Farkas lemma

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
Affine Scheduling

A schedule maps each iteration to a virtual time

$$\theta(\vec{i}) = T \begin{pmatrix} \vec{i} \\ \vec{p} \\ 1 \end{pmatrix}$$

- The number of rows in $T$ is the dimensionality of the schedule.
- The number of rows in $T$ is also the number of outermost sequential loops.

Scheduling in the Polyhedral Model

Legality
- The schedule must respect all the dependences.
- Let’s turn dependence relations into constraints on the schedule solution set.
  - If iteration $\vec{i}_R$ of statement R needs to execute before iteration $\vec{i}_S$ of statement S, then the schedules for statement R and S need to satisfy the following constraint:

$$\theta_R(\vec{i}_R) < \theta_S(\vec{i}_S)$$

One-dimensional schedules

$$\theta_R(\vec{i}_R) < \theta_S(\vec{i}_S)$$
**Constraint for schedule legality**

**Time delta**
- between statement instances with dependences,
- needs to be non-negative over the dependence polyhedron

\[ \Delta_{R,S} = \theta_S(i_S) - \theta_R(i_R) - 1 \geq 0 \]

<Example dependence polyhedron done on paper>

---

**Turning this observation into scheduling constraints**

**Affine form of Farkas lemma**
- Let \( D \) be a nonempty polyhedron defined by \( A\vec{i} + \vec{b} \geq 0 \).
- Any affine function \( f(i) \) is non-negative everywhere in \( D \) if and only if it is a positive affine combination of the constraints for \( D \):

\[ f(i) = \lambda_0 + \vec{\lambda}^T (A\vec{i} + \vec{b}) \]

with \( \lambda_0 \geq 0 \) and \( \vec{\lambda} \geq \vec{0} \)

where \( \lambda_0 \) and \( \vec{\lambda}^T \) are called the Farkas multipliers.
Building intuition about the Farkas lemma

Fig. 2. An illustration of Farkas’ lemma. The affine form \( h(x, y) = 2 \cdot (x - 1) \) is nonnegative within the shaded polyhedron. Thus, it can be expressed as a nonnegative affine combination of the faces of that polyhedron: \( h(x, y) = 2 \cdot (y - 4) \).

Using the Farkas lemma

Assume the following dependence polyhedron

\[
D_{R \rightarrow S} = \{ (i, j) \mid \begin{bmatrix} i \\ j \\ \bar{p} \\ 1 \end{bmatrix} \geq 0 \text{ and } B \begin{bmatrix} i \\ j \\ \bar{p} \\ 1 \end{bmatrix} = \bar{0} \}
\]

Assume a schedule function of the form

\[
\theta_R(i) = \bar{v}^T \bar{i} + \bar{b}
\]

\[
\theta_S(j) = \bar{w}^T \bar{j} + \bar{c}
\]

We need \( \Delta_{R,S} = \theta_S(i) - \theta_R(j) - 1 \geq 0 \)
The process of determining set of legal schedules

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

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

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

$$\theta_S(i) - \theta_R(j) - 1 = \lambda_0 + \bar{\lambda}^T (A' \begin{bmatrix} i \\ j \\ p \\ 1 \end{bmatrix})$$

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

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

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

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

Example of using the Farkas lemma

Original code

```java
    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
- How do we select a best schedule?
- Other approaches for automating the scheduling problem.

Schedule
- HW8 due tomorrow, April 11th
for (i = 0; i < 5; i++) {
    A[i] = A[i-1] + sm(i);
}

Dep relation
E[i] -> [i'] | 0 ≤ i < 5 ∧
            | 0 ≤ i' < 5 ∧
            | i = i' - 1 ∧
            | i ≤ i' |

Dep polyhedron

initial schedule
θ(i) = i

R+ S statement are the same

need \[ \Delta_{R+S} = \theta(i') - \theta(i) - 1 \geq 0 \]

= i' - i - 1 \geq 0

= (i+1) - i - 1 \geq 0

0 \geq 0 ✓

θ(i) = i + 1

θ(i) = i + k

legal schedules

any constant
Specific Example

\[ D = \{ [x, y] \mid -y + x + 3 \geq 0 \land y - 4 \geq 0 \land -y - x + 11 \geq 0 \} \]

\[ f([x, y]) = 2(x - 1) \]

\[ 2(x - 1) = \lambda_0 + \lambda_1(-y + x + 3) + \lambda_2(y - 4) \]
\[ + \lambda_3(-y - x + 11) \]

\[ 0 = (\lambda_0 + 2 + 3\lambda_1 - 4\lambda_2 + 11\lambda_3) + (\lambda_1 - 2 - \lambda_3)x \]
\[ + (-\lambda_1 + \lambda_2 - \lambda_3)y \]

\[ \lambda_0 + 2 + 3\lambda_1 - 4\lambda_2 + 11\lambda_3 = 0 \]

\[ \lambda_1 - 2 - \lambda_3 = 0 \Rightarrow \lambda_1 = 2 + \lambda_3 \]

\[ -\lambda_1 + \lambda_2 - \lambda_3 = 0 \Rightarrow \lambda_2 = \lambda_1 + \lambda_3 = 2 + 2\lambda_3 \]
(a) Dependence polyhedron for distance vector $(1, -1)$

\[
D = \exists [i, j] \rightarrow [i', j'] \mid \begin{align*}
&i = i' - 1 \\
&j = j' + 1 \\
&1 \leq i, i' \leq 6 \\
&1 \leq j, j' \leq 5
\end{align*}
\]

(b) Change all equality constraints to inequality constraints:

\[
0 = \exists [i, j] \rightarrow [i', j'] \mid \begin{align*}
&i - i' + 1 \geq 0 \\
&i + i' - 1 \geq 0 \\
&j - j' + 1 \geq 0 \\
&j + j' - 1 \geq 0 \\
&i' - 1 \geq 0 \\
&j' - 1 \geq 0 \\
&6 - i \geq 0 \\
&6 - j \geq 0
\end{align*}
\]

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

Let \( \Theta([x, y]) = v_0 x + v_1 y + b_0 \)

\[
\begin{align*}
&v_0 i' + v_1 j' + b_0 - v_0 i - v_1 j - b_0 - 1 = \lambda_0 + \vec{\lambda}^T A' \\
&\text{RI: } v_0 i' + v_1 j' - v_0 i - v_1 j - 1 = \lambda_0 + \lambda_1 i - \lambda_3 i' + \lambda_5 \\
&\text{R2: } v_0 i' + v_1 j' - v_0 i - v_1 j - 1 = \lambda_0 + \lambda_2 j - \lambda_4 j' + \lambda_5
\end{align*}
\]