



High-Performance Embedded Systems-on-a-Chip

Lecture 15: ACL-2-SARE

Sanjay Rajopadhye

Computer Science, Colorado State University

Exact Data-flow Analysis of ACL's

- Bubble sort example (on the board)
- Notations (recap again)
- The theory (Feautrier's algorithm)
- Another Example: Forward Substitution
- Exercise

Parameterized Polyhedra

- $\mathcal{P}_1 = \{i, j, n \mid 0 \leq j \leq i; j \leq n; 1 \leq n\}$

Parameterized Polyhedra

- $\mathcal{P}_1 = \{i, j, n \mid 0 \leq j \leq i; j \leq n; 1 \leq n\}$
- 3-D polyhedron with
2 vertices: $[0, 0, 1]$ and $[0, 1, 1]$,
and 3 rays: $[1, 0, 0]$, $[0, 0, 1]$ and $[0, 1, 1]$

Parameterized Polyhedra

- $\mathcal{P}_1 = \{i, j, n \mid 0 \leq j \leq i; j \leq n; 1 \leq n\}$
- 3-D polyhedron with
2 vertices: $[0, 0, 1]$ and $[0, 1, 1]$,
and 3 rays: $[1, 0, 0]$, $[0, 0, 1]$ and $[0, 1, 1]$
- also a 2-D polyhedron (parameterized by n):
2 vertices, $\{[0, 0], [n, 0]\}$ and a ray, $[1, 0]$.

Parameterized Polyhedra

- $\mathcal{P}_1 = \{i, j, n \mid 0 \leq j \leq i; j \leq n; 1 \leq n\}$
- 3-D polyhedron with
2 vertices: $[0, 0, 1]$ and $[0, 1, 1]$,
and 3 rays: $[1, 0, 0]$, $[0, 0, 1]$ and $[0, 1, 1]$
- also a 2-D polyhedron (parameterized by n):
2 vertices, $\{[0, 0], [n, 0]\}$ and a ray, $[1, 0]$.
- Consider
 $\mathcal{P}_2 = \{i, j, n, m \mid 0 \leq j \leq i \leq n; j \leq m; 1 \leq n, m\}$ as a
2-D polytope (no rays) parameterized by n and m .

Parameterized Polyhedra

- $\mathcal{P}_1 = \{i, j, n \mid 0 \leq j \leq i; j \leq n; 1 \leq n\}$
- 3-D polyhedron with
2 vertices: $[0, 0, 1]$ and $[0, 1, 1]$,
and 3 rays: $[1, 0, 0]$, $[0, 0, 1]$ and $[0, 1, 1]$
- also a 2-D polyhedron (parameterized by n):
2 vertices, $\{[0, 0], [n, 0]\}$ and a ray, $[1, 0]$.
- Consider
 $\mathcal{P}_2 = \{i, j, n, m \mid 0 \leq j \leq i \leq n; j \leq m; 1 \leq n, m\}$ as a
2-D polytope (no rays) parameterized by n and m .
- What are its vertices?

Depends on the relative values of n and m : either a triangle (if $n \leq m$) or a trapezium (otherwise).

$$\begin{cases} \{1 \leq n \leq m\} & \Rightarrow \{[0, 0], [n, 0], [n, n]\} \\ \{1 \leq m < n\} & \Rightarrow \{[0, 0], [n, 0], [n, m], [m, m]\} \end{cases}$$

Set of vertices of a parameterized polytope (polyhedron) is a piecewise affine function of the parameters

- Only control structure is the `for` loop
- Lower (cf. upper) bounds of loops are affine expressions of surrounding loops, or `max` (cf. `min`) of such expressions
- Body of a loop is either (i) a loop, or (ii) an assignment statement, or (iii) a sequence of (any of) these two
- Program variables: either index-vars, or params, or (multidimensional) array variables (scalars are zero-dimensional arrays)
- Assignment statement: lhs variable is an array var, accessed by an affine function of indices (and params); rhs is an expression containing such (indexed) array vars.

Example: Forward Substitution

```
S1:      x[1] := b[1]/a[1,1];  
        for i = 2 to n do  
S2:          s := 0;  
            for j = 1 to i-1 do  
S3:                s := s + x[j] * a[i, j];  
            enddo  
S4:          x[i] := (b[i] - s) / a[i,i]  
        enddo
```

- each assignment statement is given a unique label, S_i , and
- represents many instances of operations,
- one instance for each valid value of surrounding indices.
- set of valid values is a polyhedron D_i (parameterized by program params)
- Each operation (i.e., each value computed by the program) is uniquely identified by $\langle S_i, z \rangle$ for $z \in D_i$
- Define n_{ij} as the number of loop indices common to two statements S_i and S_j

Even more notation

- **lexicographic order**, \prec
- **program execution order**, defined as $\langle S_1, z_1 \rangle \triangleleft \langle S_2, z_2 \rangle$ iff

$$\begin{cases} z'_1 \prec z'_2 & \text{if } S_1 \text{ appears **before** } S_2 \text{ in pgm txt} \\ z_1 \prec z_2 & \text{otherwise} \end{cases}$$

- Lmax (cf. Lmin) is the lexicographic maximum (cf. minimum) of two or more vectors.
- **Remark:** Lmax (or Lmin) of all points in a polytope is one of its vertices: **piecewise affine function of its params.**
- **Remark:** Lmax (cf. Lmin) of two or more piecewise affine functions is also a piecewise affine function.

- Variables of the SARE = statements of the ACL

- Variables of the SARE = statements of the ACL
- Variable domains = domains of the statements

- Variables of the SARE = statements of the ACL
- Variable domains = domains of the statements
- Consider an ACL:

$$S_i : X[f_i^0(z, p)] = g(\dots Y[f_i^k(z, p)] \dots)$$

- Variables of the SARE = statements of the ACL
- Variable domains = domains of the statements
- Consider an ACL:

$$S_i : X[f_i^0(z, p)] = g(\dots Y[f_i^k(z, p)] \dots)$$

- **Key question:**
Which instance of which statement wrote the value of y read by the z -th instance of S_i ?

- Must be some statement that has Y on the left.
- For each candidate, of the form $S_j : Y[f_j^0(z', p)] = \dots$, the solution must satisfy:

$$\left\| \begin{array}{l} z \in D_i \\ z' \in D_j \\ f_i^k(z, p) = f_j^0(z', p) \\ \langle S_j, z' \rangle \triangleleft \langle S_i, z \rangle \end{array} \right. \quad (0)$$

- This gives us a set of points: our solution is the **most recent one: lexicographic max**
- Points satisfying Eqn (1) constitute a union of polyhedra
- Overall solution: piecewise affine function of z and pgm params

Example: Forward Substitution

```
S1:      x[1] := b[1]/a[1,1];  
        for i = 2 to n do  
S2:          s := 0;  
            for j = 1 to i-1 do  
S3:                s := s + x[j] * a[i, j];  
            enddo  
S4:          x[i] := (b[i] - s) / a[i,i]  
        enddo
```

Find the source of s in $S3$

Two statements write into s ,

$\text{Src}(s, S3) = \text{Last}(\langle S2, f_1(i, j) \rangle, \langle S3, f_2(i, j) \rangle)$, where

$$f_1(i, j) = \text{Last} \left\{ (i' \mid i, j) \left\| \begin{array}{l} i' \in \mathcal{D}_2 \\ (i, j) \in \mathcal{D}_3 \\ \langle S2, i' \rangle \triangleleft \langle S3, (i, j) \rangle \end{array} \right. \right\}$$

$$f_2(i, j) = \text{Last} \left\{ (i', j' \mid i, j) \left\| \begin{array}{l} (i', j') \in \mathcal{D}_3 \\ (i, j) \in \mathcal{D}_3 \\ \langle S3, (i', j') \rangle \triangleleft \langle S3, (i, j) \rangle \end{array} \right. \right\}$$

Now, $\langle S2, i' \rangle \triangleleft \langle S3, (i, j) \rangle$ iff $i' \leq i$.

Hence

$$\begin{aligned} f_1(i, j) &= \text{Last} \left\{ (i' \mid i, j) \left\| \begin{array}{l} i' \in \mathcal{D}_2 \\ (i, j) \in \mathcal{D}_3 \\ i' \leq i \end{array} \right. \right\} \\ &= \text{Last} \{ (i' \mid i, j) \mid 2 \leq i' \leq i \leq n; 1 \leq j < i \leq n \} \\ &= i \end{aligned}$$

For $f_2(i, j)$, $\langle S3, (i', j') \rangle \triangleleft \langle S3, (i, j) \rangle$ simplifies to $i > i' \vee (i' = i \wedge j' < j)$,
a disjunction of two sets of inequality constraints.

Now, $f_2(i, j) =$

$$\begin{aligned} & \text{Last} \left(\left\{ (i', j' | i, j) \parallel \begin{array}{l} (i', j') \in \mathcal{D}_3 \\ (i, j) \in \mathcal{D}_3 \\ i' < i \end{array} \right\} \cup \left\{ (i', j' | i, j) \parallel \begin{array}{l} (i', j') \in \mathcal{D}_3 \\ (i, j) \in \mathcal{D}_3 \\ i' = i; j' < j \end{array} \right\} \right) \\ = & \text{Last} \left(\text{Last} \left\{ (i', j' | i, j) \parallel \begin{array}{l} (i', j') \in \mathcal{D}_3 \\ (i, j) \in \mathcal{D}_3 \\ i' < i \end{array} \right\}, \text{Last} \left\{ (i', j' | i, j) \parallel \begin{array}{l} (i', j') \in \mathcal{D}_3 \\ (i, j) \in \mathcal{D}_3 \\ i' = i; j' < j \end{array} \right\} \right) \\ = & \text{Last} (\text{Last}\{(i', j' | i, j) \parallel 1 \leq j' < i' < i \leq n; 1 \leq j < i \leq n\} \\ & \text{Last}\{(i', j' | i, j) \parallel 1 \leq j < i' = i \leq n; j' < j\}) \end{aligned}$$

So $f_2(i, j) = \mathbf{Last}(f'_2(i, j), f''_2(i, j))$

where

$$f'_2(i, j) = \begin{cases} \{i, j | i = 2; j = 1\} & : \perp \\ \{i, j | 1 \leq j < i \leq n; i \geq 3\} & : (i - 1, i - 2) \end{cases}$$

and

$$f''_2(i, j) = \begin{cases} \{i, j | 1 = j < i \leq n\} & : \perp \\ \{i, j | 1 < j < i \leq n\} & : (i, j - 1) \end{cases}$$

and hence

$$\begin{aligned} f_2(i, j) &= \begin{cases} \{i, j | i = 2; j = 1\} & : \text{Last}(\perp, \perp) \\ \{i, j | i \geq 3; j = 1\} & : \text{Last}((i - 1, i - 2), \perp) \\ \{i, j | 1 < j < i \leq n\} & : \text{Last}((i - 1, i - 2), (i, j - 1)) \end{cases} \\ &= \begin{cases} \{i, j | i = 2; j = 1\} & : \perp \\ \{i, j | j = 1; i \geq 3\} & : (i - 1, i - 2) \\ \{i, j | 1 < j < i \leq n\} & : (i, j - 1) \end{cases} \end{aligned}$$

Finally, remember that

$$\text{Src}(s, S3) = \text{Last}(\langle S2, f_1(i, j) \rangle, \langle S3, f_2(i, j) \rangle)$$

$$= \text{Last} \left(\langle S2, i \rangle, \left\langle S3, \begin{cases} \{i, j | i = 2; j = 1\} & : \perp \\ \{i, j | j = 1; i \geq 3\} & : (i - 1, i - 2) \\ \{i, j | 1 < j < i \leq n\} & : (i, j - 1) \end{cases} \right\rangle \right)$$

$$= \text{Last} \left(\langle S2, i \rangle, \begin{cases} \{i, j | i = 2; j = 1\} & : \langle S3, \perp \rangle \\ \{i, j | j = 1; i \geq 3\} & : \langle S3, (i - 1, i - 2) \rangle \\ \{i, j | 1 < j < i \leq n\} & : \langle S3, (i, j - 1) \rangle \end{cases} \right)$$

$$= \begin{cases} \{i, j | i = 2; j = 1\} & : \text{Last}(\langle S2, i \rangle, \langle S3, \perp \rangle) \\ \{i, j | j = 1; i \geq 3\} & : \text{Last}(\langle S2, i \rangle, \langle S3, (i - 1, i - 2) \rangle) \\ \{i, j | 1 < j < i \leq n\} & : \text{Last}(\langle S2, i \rangle, \langle S3, (i, j - 1) \rangle) \end{cases}$$

$$= \begin{cases} \{i, j | i = 2; j = 1\} & : \langle S2, i \rangle \\ \{i, j | j = 1; i \geq 3\} & : \langle S2, i \rangle \\ \{i, j | 1 < j < i \leq n\} & : \langle S3, (i, j - 1) \rangle \end{cases}$$

$$= \begin{cases} \{i, j | 1 = j < i \leq n\} & : \langle S2, i \rangle \\ \{i, j | 1 < j < i \leq n\} & : \langle S3, (i, j - 1) \rangle \end{cases}$$

Exercise: Backward substitution

```
S1:      x[n] := b[n]/u[n,n];
         for i = n-1 down to 1 do
S2:           s := 0;
           for j = i+1 to n do
S3:               s := s + x[j] * u[i, n-j]
           enddo
S4:           x[i] := (b[i] - s) / u[i,i]
         enddo
```