COMP 515: Advanced Compilation for Vector and Parallel Processors

Prof. Vivek Sarkar
Department of Computer Science
Rice University
vsarkar@rice.edu

https://wiki.rice.edu/confluence/display/PARPROG/COMP515

Dependence Testing

Allen and Kennedy, Chapter 3 (contd)

The General Problem

```
DO i1 = L1, U1
  DO i2 = L2, U2
            …
    DO in = Ln, Un
 S1                   A(f1(i1,…,in),…,fm(i1,…,in)) = …
 S2                   … = A(g1(i1,…,in),…,gm(i1,…,in))
    ENDDO
    ...
  ENDDO
ENDDO
```

Under what conditions is the following true for iterations α and β ?

$f_i(\alpha) = g_i(\beta)$ **for all** $i, 1 \le i \le m$

Note that the number of equations equals the rank of the array,

and the number of variables is twice the number of loops that enclose both array references (two iteration vectors)

Index Set Splitting

```
DO I = 1,100
   DO J = 1, I
S1       A(J+20) = A(J) + B
   ENDDO
ENDDO
```

For values of

$$I < \frac{|d| - (U_0 - L_0)}{U_1 - L_1} = \frac{20 - (-1)}{1} = 21$$

there is no dependence carried by loop J

Index Set Splitting

- This condition can be used to partially parallelize the loop by splitting the range as shown below:

```
    DO I = 1,20
         DO J = 1, I
S1a              A(J+20) = A(J) + B
         ENDDO
    ENDDO

    DO I = 21,100
         DO J = 1, I
S1b              A(J+20) = A(J) + B
         ENDDO
  ENDDO
```

Now the inner loop for the first nest can be parallelized

Breaking Conditions

- Consider the following example

```
        DO I = 1, L
S1          A(I + N) = A(I) + B
        ENDDO
```

- If L<=N, then there is no dependence from S_1 to itself
- L<=N is called the Breaking Condition

Using Breaking Conditions

- By using breaking conditions, the compiler can generate alternative code for the sequential and parallel cases

```
      IF (L<=N) THEN
              A(N+1:N+L) = A(1:L) + B
      ELSE
              DO I = 1, L
S1                    A(I + N) = A(I) + B
              ENDDO
      ENDIF
```

Restatement of Dependence Analysis Problem with Direction Vectors

- **General Dependence:**
  - Let $D = (D_1, D_2, \ldots, D_n)$ be a direction vector, and consider the following loop nest

```
      DO i1 = L1, U1
            DO i2 = L2, U2
                      …
                      DO in = Ln, Un
S1                              A(f(i)) = …
S2                              … = A(g(i))
                                ENDDO
                      …
            ENDDO
```

Then $S_2 \, \delta \, S_1$ for direction vector D if f(x) = g(y) can be solved for iteration vectors x,y that are consistent with D, and are also within loop bounds

Using Linear Diophantine Equations to solve the MIV case

- **For simplicity, assume that**

$$f(x) = a_0 + a_1 x_1 + \ldots + a_n x_n$$

$$g(y) = b_0 + b_1 y_1 + \ldots + b_n y_n$$

- **Define h(x,y) = f(x) – g(y). Then looking for solutions for f(x) = g(y) is equivalent to looking for solutions of h(x,y) = 0 i.e.,**

$$h(x,y) = a_0 - b_0 + a_1 x_1 - b_1 y_1 + \ldots + a_n x_n - b_n y_n = 0$$

- **Rearranging terms, we get the linear Diophantine Equation:**

$$a_1 x_1 - b_1 y_1 + \ldots + a_n x_n - b_n y_n = b_0 - a_0$$

Linear Diophantine Equations & GCD Test

- Observation: for integer values $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$ it must be the case that

  $$a_1 x_1 - b_1 y_1 + \ldots + a_n x_n - b_n y_n$$

  is a multiple of

  $$\gcd(a_1, \ldots, a_n, b_1, \ldots, b_n)$$

- As a result, the equation

  $$a_1 x_1 - b_1 y_1 + \ldots + a_n x_n - b_n y_n = b_0 - a_0$$

- has an integer solution iff $\gcd(a_1, \ldots, a_n, b_1, \ldots, b_n)$ divides $b_0 - a_0$
  - But the solution may not be in the region (loop iteration values) of interest

- Time for Worksheet 1!

Real Solutions

- Unfortunately, the GCD test is less useful then it might seem.
- A more useful approach is to show that the equation has no solutions in region R of interest (loop bounds and direction vectors)

  → explore real solutions for this purpose

  → absence of real solutions guarantees absence of integer solutions
- Solving h(x,y) = 0 is essentially an integer programming problem, but linear programming techniques can be used as an approximation.
- Since the function is continuous, the Intermediate Value Theorem says that a solution exists in region R iff:

$$\min_R h \leq 0 \leq \max_R h$$

Positive and Negative Parts of Real Numbers

- Thus, we need an easy way to calculate $\min_R h$ and $\max_R h$.

- Definitions:

$$h_i^+ = \max_{Ri} h(x_i, y_i) \qquad a^+ = \begin{cases} a & a \geq 0 \\ 0 & a < 0 \end{cases}$$

$$h_i^- = \min_{Ri} h(x_i, y_i) \qquad a^- = \begin{cases} |a| & a < 0 \\ 0 & a \geq 0 \end{cases}$$

- $a^+ = \max(a,0)$ and $a^- = -\min(a,0)$ are both >= 0 and are called the positive part and negative part of a
- Claim: $a = a^+ - a^-$. Why?

Banerjee Inequality

Lemma 3.2. Let t,l,u,z be real numbers. If l <= z <= u, then

$$-t^{-}u + t^{+}l \leq tz \leq t^{+}u - t^{-}l$$

Furthermore, there are numbers z_1 and z_2 in [l,u] that make each of the inequalities hold as equalities i.e., the first inequality holds as an equality for $z = z_1$ and the second inequality holds as an equality for $z = z_2$

Proof: In the book.

Building intuition: consider cases when t = +1 and t = -1

Banerjee Inequality (contd)

Definitions:

$$H_i^-(=) = -(a_i - b_i)^- U_i + (a_i - b_i)^+ L_i$$

$$H_i^+(=) = (a_i - b_i)^+ U_i - (a_i - b_i)^- L_i$$

$$H_i^-(<) = -(a_i^- + b)^+ (U_i - 1) + [(a_i^- + b_i)^- + a_i^+] L_i - b_i$$

$$H_i^+(<) = (a_i^+ - b_i)^+ (U_i - 1) - [(a_i^+ - b_i)^+ + a_i^-] L_i - b_i$$

$$H_i^-(>) = -(a_i - b_i)^- (U_i - 1) + [(a_i - b_i^+)^+ + b_i^-] L_i + a_i$$

$$H_i^+(>) = (a_i + b_i)^+ (U_i - 1) - [(a_i + b_i^-)^- + b_i^+] L_i + a_i$$

$$H_i^-(*) = a_i^- U_i^x + a_i^+ L_i^x - b_i^+ U_i^y + b_i^- L_i^y$$

$$H_i^+(*) = a_i^+ U_i^x - a_i^- L_i^x + b_i^- U_i^y - b_i^+ L_i^y$$

Banerjee Inequality (contd)

- Now for the main lemma:
- Lemma 3.3: Let D be a direction vector, and h be a dependence function. Let $h_i(x_i, y_i) = a_i x_i - b_i y_i$ and R_i be as described above. Then h_i obtains its minimum and maximum on R_i, and we have

$$\min_{Ri} h_i = h_i^- = H_i^-(D_i)$$

$$\max_{Ri} h_i = h_i^+ = H_i^+(D_i)$$

Banerjee Inequality (contd)

- **Proof of 3.3:**

  We must check for all cases of D_i .

  If D_i = '=', then $x_i=y_i$ and $h_i=(a_i-b_i)\,x_i$. We clearly satisfy the hypothesis of lemma 3.2, so

$$-(a_i - b_i)^- U_i + (a_i - b_i)^+ L_i = H_i^-(=) \le h \le (a_i - b_i)^+ U_i - (a_i - b_i)^- L_i = H_i^+(=)$$

  Furthermore, h_i actually obtains these bounds by lemma 3.2. Thus, the result is established.

Banerjee Inequality (contd)

If D_i = '<", we have that L_i <= x_i < y_i <= U_i. Rewrite this as L_i <= x_i <= y_i -1 <= U_i - 1 in order to satisfy the conditions for lemma 3.2. Also, rewrite h as

$$h_i = a_i x_i - b_i y_i = a_i x_i - b_i(y_i - 1) - b_i$$

Then, we can use 3.2 to first minimize $a_i x_i$ and get:

$$-a_i^-(y_i - 1) + a_i^+ L_i - b_i(y_i - 1) - b_i \le h_i \le a_i^+(y_i - 1) - a_i^- L_i - b_i(y_i - 1) - b_i$$

Minimizing the $b_i(y_i-1)$ term then gives us:

$$-(a_i^- + b_i)^+(U_i - 1) + (a_i^- + b_i)^- L_i + a_i^+ L_i - b_i = H_i^-(<) \le h_i$$
$$\le (a_i^+ - b_i)^+(U_i - 1) - (a_i^+ - b_i)^- L_i - a_i^- L_i - b_i = H_i^+(<)$$

The other cases are similar, as you will see in Homework 2!

Banerjee Inequality

- **Theorem 3.3 (Banerjee). Let D be a direction vector, and h be a dependence function. h = 0 can be solved in the region R iff:**

$$\sum_{i=1}^{n} H_i^-(D_i) \le b_0 - a_0 \le \sum_{i=1}^{n} H_i^+(D_i)$$

Proof: Immediate from Lemma 3.3 and the Intermediate Value Theorem

Example

```
DO I = 1, N
   DO J = 1, M
      DO K = 1, 100
         A(I,K) = A(I+J,K) + B
      ENDDO
   ENDDO
ENDDO
```

Testing (I, I+J) for D = (=,<,*):

$$H_1^-(=) + H_2^-(<) = -(1-0)^- N + (1-1)^+ 1 - (0^- + 1)^+ (M-1) + [(0^- + 1)^- + 0^+]1 - 1 = -M \le 0$$

$$\le H_1^+(=) + H_2^+(<) = (1-1)^+ N - (1-1)^- 1 + (0^+ - 1)^+ (M-1) - [(0^+ - 1)^- + 0^-]1 - 1 \le -2$$

This is impossible, so the dependence cannot exist!

Homework #2 (Written Assignment)

- **Solve exercise 3.6 in book**
  - **This is case 4 of Lemma 3.3**
  - **Read Definitions 3.1, 3.2, 3.3 and Lemmas 3.1, 3.2, 3.3 before starting**
- **Due in class on Tuesday, Sep 22nd**

- Honor Code Policy: All submitted homeworks are expected to be the result of your individual effort. You are free to discuss course material and approaches to problems with your other classmates and the professors, but you should never misrepresent someone else's work as your own. If you use any material from external sources, you must provide proper attribution.

Worksheet 1 (Lecture 5)

Name: ____________________ **Netid:** ____________________

```
DO I = 1, N
S1        A(4*I+2) = …
S2        …        = A(4*I+4)
ENDDO
```

- **Use the GCD test to determine whether there is a dependence between S1 and S2.**