COMP 515: Advanced Compilation for Vector and Parallel Processors

Prof. Krishna Palem
Prof. Vivek Sarkar
Department of Computer Science
Rice University
{palem,vsarkar}@rice.edu

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

Acknowledgments

- Slides from previous offerings of COMP 515 by Prof. Ken Kennedy
  - http://www.cs.rice.edu/~ken/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 ?

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

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 vectorize S1 by Index set splitting as shown

```
    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 vectorized

Coupling makes these tests imprecise

```
      DO I = 1,100
              DO J = 1, I
S1                    A(J+20,I) = A(J,19) + B
              ENDDO
      ENDDO
```

- We will report dependence even if there isn't any
- But such cases are very rare

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

- Using breaking conditions the compiler can generate alternative code

```
      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

- General Dependence:
  - Let $(D_1, D_2, \ldots, D_n)$ 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 $S_2 \; \delta \; S_1$ if $f(x) = g(y)$ can be solved for iteration vectors x,y that agree with D.

The General Case

- We must relax our restrictions on f and g to let them be arbitrary functions.
- A dependency exists if

$$h(x_1, x_2, ..., x_n, y_1, y_2, ..., y_n) = f(x_1, x_2, ..., x_n, y_1, y_2, ..., y_n) - g(x_1, x_2, ..., x_n, y_1, y_2, ..., y_n) = 0$$

- has an integral solution
- The above equation is known as a Diophantine equation.
- We will also impose the constraint that the solution must occur within loop bounds

Linear Diophantine Equations

- **For simplicity, assume that**

$$f(x) = a_0 + a_1 x_1 + \ldots + a_n x_n \qquad \text{f(x) = a0 + a1 x1 + …}$$

$$g(x) = b_0 + b_1 y_1 + \ldots + b_n y_n \qquad \text{g(y) = b0 + b1 y1 + …}$$

- **Then, we're looking for a solution of**

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

h(x,y)

- **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

- A basic result tells us that there are values for $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$ so that

$$a_1 x_1 - b_1 y_1 + \ldots + a_n x_n - b_n y_n = \gcd(a_1, \ldots, a_n, b_1, \ldots, b_n)$$

  What's more, gcd($a_1, \ldots, a_n, b_1, \ldots, b_n$) is the smallest number this is true for.

- As a result, the equation has a 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
- Exercise: try this result on the A(4*i+2) & A(4*i+4) example

Real Solutions

- Unfortunately, the gcd test is less useful then it might seem.
- Useful technique is to show that the equation has no solutions in region of interest ==> explore real solutions for this purpose
- Solving h(x) = 0 is essentially an integer programming problem. Linear programming techniques are used as an approximation.
- Since the function is continuous, the Intermediate Value Theorem says that a solution exists iff:

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

Banerjee Inequality

- 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^+ and a^- are both >= 0 and are called the positive part and negative part of a, so that $a = a^+ - a^-$

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 true.

Proof: In the book.

Banerjee Inequality

- **Definitions:**
  - $H_i^-(<) = -(a_i^- + b)^+(U_i - 1) + [(a_i^- + b_i)^- + a_i^+]L_i - b_i$
  - $H_i^+(<) = (a_i^+ - b_i)^+(U_i - 1) - [(a_i^+ - b_i)^+ + a_i^-]L_i - b_i$
  - $H_i^-(=) = -(a_i - b_i)^-U_i + (a_i - b_i)^+L_i$
  - $H_i^+(=) = (a_i - b_i)^+U_i - (a_i - b_i)^-L_i$
  - $H_i^-(>) = -(a_i - b_i)^-(U_i - 1) + [(a_i - b_i^+)^+ + b_i^-]L_i + a_i$
  - $H_i^+(>) = (a_i + b_i)^+(U_i - 1) - [(a_i + b_i^-)^- + b_i^+]L_i + a_i$
  - $H_i^-(*) = a_i^-U_i^x + a_i^+L_i^x - b_i^+U_i^y + b_i^-L_i^y$
  - $H_i^+(*) = a_i^+U_i^x - a_i^-L_i^x + b_i^-U_i^y - b_i^+L_i^y$

Banerjee Inequality

- 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

- **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

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.

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 IMV.

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 dependency doesn't exist.

Homework #2 (Written Assignment)

- **Solve exercise 3.2 in book**
  - As before, dependence "type" refers to flow/anti/output
- Due in class on Sep 24th
  - 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.

Course Project Logistics

- Goal of course project is to perform an in-depth study of a research problem related to the course
  - Should include a theoretical component
  - Practicality can be demonstrated by hand-coded source-to-source transformations
- We will assign you a senior PhD student or research scientist as a mentor for your project
  - Kumud, Anitha – Mentor: Rishi Surendran
  - Nick, Sriraj, Yiwei – Mentor: Zoran Budimlic
- September 17 & 19 are self-study days for you to meet with your mentor and develop your project proposal (due by Sep 20)
- Final project presentations scheduled in class on Dec 3 & 5

Worksheet 1 (to be done in groups)

```
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