

CS6013 - Modern Compilers: Theory and Practise

Dependence testing

V. Krishna Nandivada

IIT Madras

Example dependence testing

```
for i ← 1 to 4 do
  b[i] ← a[4*i] + 2.0
  a[2*i+1] ← 1.0/i
endfor
for i ← 1 to 4 do
  b[i] ← a[3*i-5] + 2.0
  a[2*i+1] ← 1.0/i
endfor
```



Opening remarks

What have we done so far?

- Compiler overview.
- Scanning and parsing.
- JavaCC, visitors and JTB
- Semantic Analysis - specification, execution, attribute grammars.
- Type checking, Intermediate Representation, Intermediate code generation.
- Control flow analysis, interval analysis, structural analysis
- Data flow analysis, intra-procedural and inter-procedural constant propagation.
- Points-to analysis
- Dependence analysis

Announcement:

- Assignment 6 will be out by evening.

Today: Dependence testing



linear Diophantine equation

$$a_1 * x_1 + a_2 * x_2 + \dots + a_n * x_n = c$$

has an integer solution for x_1, x_2, \dots , iff

GCD (a_1, a_2, \dots, a_n) divides c .



GCD test - intuition

- A simple and sufficient test
- if a loop carried dependency exists between $X[a * i + b]$ and $X[c * i + d]$, then $GCD(c, a)$ must divide (db) .



GCD Test Generalization

```

for i1 ← 1 to hi1 do
  for i2 ← 1 to hi2 do
    ...
    for in ← 1 to hin do
      ...
      x[... , a0 + a1 * i1 + ... + an * in, ...] ...
      ...
      x[... , b0 + b1 * i1 + ... + bn * in, ...] ...
      ...
    endfor
  endfor
endfor

```

- may be accessed inside loop nest using indices of multiple loops.
- Array may be multi-dimensional.
- There is a dependence present iff, for each subscript position in the equation

$$a_0 + \sum_{j=1}^n a_j * i_{j_1} = b_0 + \sum_{j=1}^n b_j * i_{j_2}$$

and the following inequalities are satisfied:

$$\forall j = 1 \dots n$$

$$1 \leq i_{j_1} \leq hi_{j_1}$$

$$1 \leq i_{j_2} \leq hi_{j_2}$$



GCD Test formula

- Developed by Utpal Bannerjee and Robert Towle (1976).
- Comparatively weak test (Marks too many accesses as dependent).
- If for any one subscript position

$$GCD \left(\bigcup_{j=1}^n Sep(a_j, b_j, j) \right) \nmid \sum_{j=0}^n (a_j - b_j)$$

where

- GCD - computes the Greatest common divisor for the set of numbers.
- “ $a \nmid b$ ” means that a does not divide b .

$$Sep(a, b, j) = \begin{cases} \{a - b\} & \text{if direction of } j \text{ is } = \\ \{a, b\} & \text{otherwise} \end{cases}$$

then the two references to the array x are independent.

- Other words: dependence \Rightarrow GCD divides the sum.



Dependence testing based on separability

- A pair of array references is separable if in each pair of subscript positions, the expressions found are of the form: $a * x + b1$ and $a * x + b2$.
- A pair of array references is weakly separable if in each pair of subscript positions, the expressions found are of the form: $a1 * x + b1$ and $a2 * x + b2$.



Dependence testing for separable array references

If the two array references are separable, then dependence exists if

- $a = 0$ and $b1 = b2$ or
- $(b1 - b2)/a \leq hi_j$



Dependence testing for weakly separable array references

- For each subscript position, we have equations of the form:
 $a1 * y + b1 = a2 * x + b2$, or $a1 * y = a2 * x + (b2 - b1)$
- Dependence exists if for a particular value of j has a solution that satisfies inequalities given by the loop bounds of loop j .
- List all such constraints for each reference.
- For any given reference if there is only one equation:
 - Say it is given by: $a1 * y = a2 * x + (b2 - b1)$
 - One linear equation, two unknowns:
Solution exists iff $GCD(a1, a2) \% (b2 - b1) = 0$



Dependence testing for weakly separable array references (contd)

- If the set of equations has two members of the form

$$\begin{aligned}a_{11} * y &= a_{21} * x + (b_{21} - b_{11}) \\ a_{12} * y &= a_{22} * x + (b_{22} - b_{12})\end{aligned}$$

Two equations and two unknowns.

If $a_{21}/a_{11} = a_{22}/a_{12}$ then rational solution exists: iff
 $(b_{21} - b_{11})/a_{11} = (b_{22} - b_{12})/a_{12}$.

If $a_{21}/a_{11} \neq a_{22}/a_{12}$ then there is one rational solution.

Once we obtain the solutions, check that they are integers and inequalities are satisfied.

- If set of equations have $n (> 2)$ members, either $n - 2$ are redundant \rightarrow use previous methods.
Else we have more equations compared to the unknowns \rightarrow overdetermined.



Example: analyzing weak separable references

```
for i ← 1 to n do
  for j ← 1 to n do
    f[i] ← g[2*i, j] + 1.0
    g[i+1, 3*j] ← h[i, i] - 1.5
    h[i+2, 2*i-2] ← 1.0/i
  endfor
endfor
```



What did we do today?

- Dependence testing.

