Programming Languages and Compilers (CS516) - Homework #3

Hari Amoor

April 1, 2020

Contents

1 For each of the following loops, specify the nature of each loop dependency (if any).

- Here, the statement S defined as `A(2i) = A(i) + 1` has a true dependence on itself. We supply direction vector $[<]$, but we cannot supply a distance vector due to the inconsistency of the dependency.

- Here, the statement S defined as `A(2i) = A(7i) + 1` has an anti-dependence on itself. We supply direction vector $[<]$, but we cannot supply a distance vector due to the inconsistency of the dependency.

- Here, the given algorithm does not have any loop dependencies.

- Here, the statement S defined as `A(i) = A(10 - i) - 5` has a true dependence on itself. We supply direction vector $[<]$, but we cannot supply a distance vector.

- Here, the statement `A(i, j) = 2A(i - 1, j + 3)` has an anti dependence on itself. We supply distance vector $\begin{bmatrix} < & > \end{bmatrix}$ and direction vector $\begin{bmatrix} 1 & -3 \end{bmatrix}$.

- Let S be the statement `A(i) = ...` and T be the statement `... = A(j + 1)`. T has a true dependence on S with direction vector $\begin{bmatrix} < & > \end{bmatrix}$ and distance vector $\begin{bmatrix} 1 & -1 \end{bmatrix}$.

- Let S be the statement `A(i) = ...` and T be the statement `... = A(j + i)`. S has a loop-independent dependence on T; thus, any direction or distance vector would be vacuous.

- By the Theorem of Simple Dependence Testing (Lecture 15, Slides 12-13), the instruction `A(i, j, i) = 2A(i, j+1, i-1)` has a dependency iff there exists $(i, j) \in I$ s.t. the following are satisfied (they clearly are not):

$$\begin{aligned} i &= i \\ j &= j + 1 \\ i &= i - 1 \end{aligned}$$

2 Assume the given sequential code and assess opportunities for concurrency.

- The supplied table is labelled for each pair of statements S_i, S_j with δ_k^j if there is a dependence between S_i and S_j. We supply the directed graph G with vertices $V = \{v_i\}$, where each v_i corresponds to S_i; the edge $e = (v_i, v_j)$ exists with designation supplied from the given table for (S_i, S_j) iff it is non-zero.

	S_1	S_2	S_3	S_4	S_5
S_1	$\vec{0}$	$\vec{0}$	$\vec{0}$	δ_1^{-1}	$\vec{0}$
S_2	$\vec{0}$	$\vec{0}$	$\vec{0}$	δ_1^{-1}	$\vec{0}$
S_3	$\vec{0}$	$\vec{0}$	$\vec{0}$	$\vec{0}$	$\vec{0}$
S_4	δ_1	δ_∞	$\delta 1$	$\vec{0}$	$\vec{0}$
S_5	δ_{inf}	$\vec{0}$	$\vec{0}$	$\vec{0}$	$\vec{0}$

- We *condense* the previously-supplied graph, i.e. to maximal strongly-connected substructures, and produce the following vectorization:

$S_3 : A(2:100, 1:100, 1:100) = A(1:99, 1:100, 2:101) + B(2:100, 2:101) * 2$
$\{S_1, S_2, S_4\}$: Execute synchronously while preserving iteration space
$S_5 : E(2:100) = D(2:100) + 3$

- With the Advanced vectorization algorithm we obtain the following vectorization:

$S_3 : A(2:100, 1:100, 1:100) = A(1:99, 1:100, 2:101) + B(2:100, 2:101) * 2$
$S_1 : D(2:100) = 100$
$\{S_2, S_4\}$: Execute synchronously
$S_5 : E(2:100) = D(2:100) + 3$

We additionally provide the following representation of a graph with two verices in Figure 3. Note that the vertices v_i correspond to the components $\{S_1, S_3, S_5\}, \{S_2, S_4\}$ respectively.

$$\begin{array}{ccc} & v_1 & v_2 \\ v_1 & \vec{0} & \vec{0} \\ v_2 & \delta 1 & \vec{0} \end{array}$$

3 Desceribe how the given lattice can be used for dependence analysis between procedures.

We supply an algorithm to use *lattice-theoretic* operators to decide whether concurrency can be achieved. Let M be an oracle machine for the $\wedge$ opoerator in some lattice L that defines the given two-dimensional array. On input of procedures a, b, compute $c = a \wedge b$. Finally, a, b to run concurrently iff no pair of statements from each a and b have are dependent; this can be done using a *vectorization* algorithm.