Compiler Design Notes

Ayush Raina

February 20, 2025

Iteration Space and Iteration Vector

Consider the code block below:

```
for(i = 0;i < N; ++i) {
    for(j = 0;j < N; ++j) {
        /* Do Some Work */
    }
}
```

$I_s = \{(i, j) : 0 \leq i, j < N\}$ and each point (i, j) in this space is called an iteration vector. i, j are called **iteration variables** or **loop induction values**. Iteration vectors tuples contains loop induction values starting from outermost loop to innermost loop.

Consider the following code block:

```
for(i = 0;i < N; ++i) {
    for(j = 0;j < i; ++j) {
        for(k = 0;k < j; ++k) {
            /* Do Some Work */
        }
    }
}
```

In this case $I_s = \{(i, j, k) : 0 \leq i < N, 0 \leq j < i, 0 \leq k < j\}$, and each point (i, j, k) in this space is called an iteration vector. $i, j, k \in \mathcal{Z}$ are called **iteration variables** or **loop induction values**.

Lexicographical Order

$(i, j) < (i', j')$ if $i < i'$ or $i = i'$ and $j < j'$. In General for n dimensional space, $(i_1, i_2, \ldots, i_n) < (i'_1, i'_2, \ldots, i'_n)$ if $\exists\, k \in \{1, 2, \ldots, n\}$ such that $i_k < i'_k$ and $i_i = i'_i$ for $i < k$.

Lexicographical Ordering of Iteration Vectors

Consider the following code block:

```
for(i = 0;i < N; ++i) {
    for(j = 0;j < N; ++j) {
        /* Do Some Work */
    }
}
```

The lexicographical ordering of iteration vectors is as follows: $(0, 0), (0, 1), (0, 2), \ldots, (0, N-1), (1, 0), (1, 1), \ldots, (1, N-1), \ldots, (N-1, 0), (N-1, 1), \ldots, (N-1, N-1)$. In this course we will plot i on y-axis and j on x-axis.

Lexicographically > 0

Iteration vectors $(a_1, a_2, \ldots, a_n)$ are said to be lexicographically > 0 if first non zero loop induction value is positive. For example, $(0, 0, 0)$ is lexicographical ≥ 0, $(0, 0, 1)$ is lexicographical > 0.

Consider the following code block:

```
for(i = 0;i < N; ++i) { /* Loop Header */
    for(j = 0;j < i; ++j) { /* Loop Header */
        A[i][j] = A[i-1][j] + A[i][j-1]; /* Loop Body */
    }
}
```

Above code blocks has N^2 iteration vectors. Hence $N^2!$ possible ways to arrange these vectors. How to find the valid orderings of these vectors?. A valid ordering means an ordering which does not change the semantics of the program.

Dependence Analysis

RAW - Read After Write
WAW - Write After Write
WAR - Write After Read

These are some of the dependencies that can occur in a program. Using this information we can take out the invalid orderings of the iteration vectors. In above program there are 2 reads R_1, R_2, one write W_1. Two instructions are dependent in one of them is write and other is read, same in the case of load and store. In this we have Write(W_1) after Read R_1, R_2. In this program to do W_1 at (i, j) using R_1 at (i', j'), the following conditions must be satisfied:

- $i = i' - 1$
- $j = j'$

Similarly we have dependence between W_1 and R_2. To do W_1 at (i, j) using R_2 at (i', j'), the following conditions must be satisfied:

- $i = i'$
- $j = j' - 1$