Multi-dimensional Interval Test

Qing Zhang
Dalian Shipping College, Dalian, China

Abstract - In this paper, we propose a sophisticated technique of data dependence analysis for distributed memory parallel environments that is used for converting sequential code into a parallel form targeted for a particular architecture. Two-dimensional arrays with subscripts formed by induction variable in real programs appear quite frequently [10]. We test if there are integer-valued solutions for two-dimensional arrays with subscripts formed by induction variable. It is demonstrated that without direction vectors there are integer-valued solutions for two-dimensional arrays with subscripts formed by induction variable. Furthermore, it is also shown that under a specific direction vector \( \bar{d} \) there are integer-valued solutions for two-dimensional arrays with subscripts formed by induction variable. Finally, we point out that for other direction vectors there are no integer-valued solutions for two-dimensional arrays with subscripts formed by induction variable.

Keywords: Parallelizing compilers, Data dependence analysis, Loop parallelization, I Test, Parallel code debugging.

1 Introduction

In automatic parallelization and parallelizing compilers, achieving a good data dependence analysis is a critical issue in order to reduce the communication overhead and to exploit parallelism of applications as much as possible. It is essential to develop a new analysis technique for data dependence. This motivates us that we are not only to develop traditional dependence tests for parallelizing transformation but also to develop new techniques for multi-dimensional induction variables that frequently occur in nested loops.
a two-dimensional array with subscripts formed by induction variable is shown in Figure 1.

An induction variable is a scalar integer variable, which is used in a loop to simulate do-variables: it is incremented or decremented by a constant in each iteration. Every induction variable can be replaced by a linear function of the loop’s index-variables. This transformation is called induction variable substitution. Since the variable $K$ in Figure 1 is an induction variable, it can be replaced by

$$z \times \left( (I_1 - 1) \times \left( \prod_{p=2}^{d} U_p \right) + (I_2 - 1) \times \left( \prod_{p=3}^{d} U_p \right) + \cdots + I_d \right),$$

where $d$ is the number of common loops and $z$ is one integer variable in Figure 1. Therefore, the code in Figure 1 is transformed into the code in Figure 2 after finishing the processing of induction variable substitution for the variable $K$.

We treat the loop iteration variables referenced in $S_1$ as being different variables from those referenced in $S_2$, subject to common loop bound, because dependence between $S_1$ and $S_2$ in Figure 2 may arise in different iterations of the common loops. Therefore, when we analyze dependence arising from a statement pair nested in $d$ common loops, the problem will involve $n$ unique variables (where $n = 2d$). Furthermore, variables $X_{2k-1}$ and $X_{2k}$ ($1 \leq k \leq d$) are different instances of the same loop iteration variable, and $L_{2k-1} = L_{2k}, U_{2k-1} = U_{2k}$, where $L_{2k-1}, L_{2k}, U_{2k-1}$ and $U_{2k}$ are lower bounds and upper bounds for $X_{2k-1}$ and $X_{2k}$, respectively.

Figure 1: Example of a nested loop with induction variable.

The problem of determining if there is dependence for the array $A$ between $S_1$ and $S_2$ in Figure 2 can be reduced to the problem of checking whether a system of two linear equations with $n$ unknown variables has an integer solution simultaneously, which satisfies the constraints for each variable in the system. Assume that two linear equations in a system are written as

$$z \times ((X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + (X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1})$$

$$\times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1}) \times U_{i_d} + (X_{i_d} - X_{i_1}) = 0$$

$$z \times ((X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + (X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1})$$

$$\times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1}) \times U_{i_d} + (X_{i_d} - X_{i_1}) = 0,$$

where each $U_k$ ($1 \leq k \leq d$) is an integer variable and is one upper bound for the $k^{th}$ loop nest. Because $z$ is the greatest common divisor for all of the coefficients in the left-hand side of (1-0), all of the coefficients in (1-0) are divided by $z$ and (1-0) is rewritten as

$$(X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + (X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1})$$

$$\times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1}) \times U_{i_d} + (X_{i_d} - X_{i_1}) = 0$$

$$(X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + (X_i - X_j) \times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1})$$

$$\times (U_{i_1} \times \cdots \times U_{i_d}) + \cdots + (X_{i_d} - X_{i_1}) \times U_{i_d} + (X_{i_d} - X_{i_1}) = 0,$$

It is postulated that the constraints to each variable in (1-1) are represented as
\[ 1 \leq X_{2k-1} \text{ and } X_{2k} \leq U_k \]

where \( 1 \leq k \leq d \).

Given a data dependence problem of two-dimensional arrays with symbolic coefficients under symbolic bounds, we want to find that (1) under what conditions there are integer-valued solutions and (2) under what conditions there is no integer-valued solution. In this section, we first discuss the case without direction vectors.

### 2.1 Determine Integer-valued Solutions for Two-dimensional Arrays with Symbolic Coefficients Under Symbolic Bounds Without Direction Vectors

A system of linear equations (1–1) with symbolic coefficients is deduced from determining whether there exists dependence for the array \( A \) between \( S_1 \) and \( S_2 \) in Figure 2. A system of linear inequalities (1–2) is derived from lower and upper bounds of loop index variables in Figure 2. If there is no integer-valued solution for linear equations in (1–1) with symbolic coefficients under the limits of (1–2), then there is no dependence. Otherwise, there is dependence for linear equations in (1–1) with symbolic coefficients under the limits of (1–2). Lemma 2–1 is presented to demonstrate that there are integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the limits of (1–2).

**Lemma 2–1:** There are integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the limits of (1–2).

**Proof:** Omitted due to the limitation of space.

We now use the following example to explain how Lemma 2–1 is applied. Consider the do-loop in Figure 3. The front-end of a parallelizing compiler can recognize that the variable \( K \) is one induction variable and the subscript of the array \( A \) is formed by the induction variable \( K \). The equations of the data dependence for the subscript of the array \( A \) are described below.

\[
(X_1 - X_2) \times N + (X_3 - X_4) = 0
\]

\[
(X_1 - X_2) \times N + (X_3 - X_4) = 0.
\]

(Ex1-1)

The I test in [2] is first used to separately test each linear equation in (Ex1–1). If one of the equations is indicated to be no integer-valued solution, then there is no integer-valued solution for all of the equations. Otherwise, the multi-dimensional I test in [28] is next applied to simultaneously check linear equations in (Ex1–1). From the I test, the first linear equation in (Ex1–1) with symbolic coefficients can be rewritten as the following interval equation:

\[
(X_1 - X_2) \times N + (X_3 - X_4) = [0, 0].
\]

(Ex1–2)

According to the I test, the term \(-X_4\) in the interval equation (Ex1–2) is moved to the right-hand side to gain the new interval equation:
From the I test, the term \( X_3 \) in the interval equation (Ex1–3) is moved to the right-hand side to gain the new interval equation:

\[
(X_1 - X_2) \times N + X_3 = [1, \; N].
\]

(Ex1–4)

Because \( N \) is the greatest common divisor for all of the coefficients in the left-hand side of the interval equation (Ex1–4), all of the coefficients in the interval equation (Ex1–4) are divided by \( N \) and the new interval equation is obtained:

\[
(X_1 - X_2) = \left[ 1 - \frac{N}{N}, \; \frac{N-1}{N} \right] = [0, 0].
\]

(Ex1–5)

Repeat the processing of the steps above until the term in left-hand side of the interval equation is reduced to zero item. Therefore, we will obtain the new interval equation \( 0 = [1 - M, \; M - 1] \). Because \( M \geq 1 \), thus \( 1 - M \leq 0 \leq M - 1 \). Therefore, there are integer-valued solutions for the first linear equation in (1–1) with symbolic coefficients. Similar processing is used to test the second linear equation in (Ex1–1). Conclusively, there are integer-valued solutions for the second linear equation.

2.2 Determine Integer-valued Solutions for Two-dimensional Arrays with Symbolic Coefficients Under Symbolic Bounds with Direction Vector

From Lemma 2–1, it is very clear that there are integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the limits of (1–2). Next, under a specific given direction vector and the limits of (1–2), whether there are integer-valued solutions for linear equations in (1–1) with symbolic coefficients will be discussed. The following lemmas are proposed to show that (1) under what kind of a specific direction vector there are no integer-valued solutions and (2) under what kind of a specific direction vector there are integer-valued solutions.

**Lemma 2–2:** There are no integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the limits of (1–2) and a specific direction vector \((<, *, \cdots, *)_d\), where \(d\) is the number of common loops and “*” is any one of \{<, =, >\}.

**Proof:** Omitted due to the limitation of space.

**Lemma 2–3:** There are no integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the limits of (1–2) and a specific direction vector \((>, *, \cdots, *)_d\), where \(d\) is the number of common loops and “*” is any one of \{<, =, >\}.

**Proof:** Similar to Lemma 2–2.

**Lemma 2–4:** There are no integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the limits of (1–2) and a specific direction vector \((=, \theta_2, \cdots, \theta_{d-1}, \theta_d)_d\), where \(d\) is the number of common loops and \(\theta_k\) is any element of \{<, =, >\}

for \(2 \leq k \leq d - 1\) and \(\theta_d\) is any element of \{<, >\}.

**Proof:** Similar to Lemma 2–2.

**Lemma 2–5:** There are integer-valued solutions for linear equations in (1–1) with symbolic coefficients under the
limits of (1–2) and a specific direction vector 
\( (=, =, \cdots, =)_d \), where \( d \) is the number of common loops.

**Proof:** Similar to Lemma 2–2.

\[
K = 0 \\
\text{DO } J = 1, M \\
\text{DO } I = 1, N \\
K = K + 1 \\
S: \ A(K+1, K+1) = B(K, K) \\
\text{ENDDO} \\
\text{ENDDO}
\]

Figure 3: A Fortran do-loop.

We now explain how Lemma 2–2 to Lemma 2–5 is applied to the do-loop in Figure 3. Consider the do-loop in Figure 3. The front-end of a parallelizing compiler can recognize that the variable \( K \) is one induction variable and the subscript of the array \( A \) is formed by the induction variable \( K \). The equations of the data dependence for the subscript of the array \( A \) under any given direction vector \( (<, *) \) are described below.

\[
(X_1 - X_2) \times N + (X_3 - X_4) = 0 \\
(X_1 - X_2) \times N + (X_3 - X_4) = 0.
\]

(Ex2–1)

From Lemma 2–2, suppose that \( \Omega_{1,j} \) is denoted as the sum of the coefficients for the first and second variables in the first linear equation and \( \Omega_{2,j} \) the sum of the coefficients to the first and second variables in the second linear equation. Because \( \Omega_{1,j} \) and \( \Omega_{2,j} \) are both 0, a set of canonical solutions produced by the multi-dimensional direction vector I test in [31] is denoted as: \( \{(1,1)\} \). According to the multi-dimensional direction vector I test, the canonical solution, \( (1, 1) \) yields one new interval equation as follows:

\[
2 \times ((X_1 - X_2) \times N + (X_3 - X_4)) = [0, 0].
\]

(Ex2–2)

Because 2 is the greatest common divisor for all of the coefficients in the left-hand side of the interval equation (Ex2–2), all of the coefficients in the interval equation (Ex2–2) are divided by 2 and the new interval equation is obtained:

\[
(X_1 - X_2) \times N + (X_3 - X_4) = [0, 0].
\]

(Ex2–3)

According to the I test, the term \( X_4 \) in the interval equation (Ex2–3) is moved to the right-hand side to gain the new interval equation:

\[
(X_1 - X_2) \times N + X_3 = [1, N].
\]

(Ex2–4)

From the I test, the term \( X_3 \) in the interval equation (Ex2–4) is moved to the right-hand side to gain the new interval equation:

\[
(X_1 - X_2) \times N = [1 - N, N - 1].
\]

(Ex2–5)

Because \( N \) is the greatest common divisor for all of the coefficients in the left-hand side of the interval equation (Ex2–5), all of the coefficients in the interval equation (Ex2–5) are divided by \( N \) and the new interval equation is obtained:

\[
(X_1 - X_2) = [\left\lfloor \frac{1 - N}{N} \right\rfloor, \left\lfloor \frac{N - 1}{N} \right\rfloor] = [0, 0].
\]

(Ex2–6)

Because the direction vector is "<" for the terms \( X_1 \) and \( X_2 \), according to the direction vector I test [8], the two terms \( X_1 \) and \( X_2 \) are moved to the right-hand side. Therefore, we obtain the new interval equation 0 = [1, \( M - 1 \)]. Because 1 ≤ 0 ≤ \( M - 1 \) is false, thus there are no integer-valued solutions for linear equations in (Ex2–1) with symbolic coefficients under a specific direction vector \( (<, *) \). Similarly, for dealing with other
specific direction vector, Lemmas 2–3 to 2–5 can be applied for dealing with other specific direction vector. Hence, from Lemma 3–5, it is obtained that there are integer-valued solutions for the array $A$ under a specific direction vector ($\ddots = \ddots$). From Lemma 2–2 to Lemma 2–4, consequently, there is no integer-valued solution for the array $A$ under other specific direction vectors. The front end of a parallelizing compiler derives that there only exists loop-independence output dependence for the do-loop and the do-loop can be executed in parallel mode.

### 2.3 Extending Symbolic Subscript Formed by Induction Variable

We can easily extend the expression "$K+C$" for the array $A$ in Figure 1 to "$a*K+C$", where $a$ is an integer variable and is not equal to zero. Since the variable $K$ in Figure 1 is one induction variable, it can be replaced by array $\ldots U_{d1} \ldots$ and is not equal to zero, all of the coefficients in (3–11) are divided by $a \times z$ and (3–11) is exactly equal to (1–1). Therefore, the following theorems can be applied to deal with whether there is dependence for the array $A$ in Figure 4.

**Theorem 2-1**: There are integer-valued solutions for linear equations in (2–11) with symbolic coefficients under the limits of (1–2) and a specific direction vector $(\ddots = \ddots, \ddots = \ddots, \ddots = \ddots)_{d}$, where $d$ is the number of common loops.

**Proof**: Similar to Lemma 2–2. $lacksquare$

**Theorem 3-2**: There are no integer-valued solutions for linear equations in (2–11) with symbolic coefficients under the limits of (1–2) and other given direction vectors.

**Proof**: Similar to Lemma 2–2. $lacksquare$

### 3 Experimental Results

We have tested our method and have performed experiments by hand on the codes abstracted from three numerical packages: Parallel loop, Vector loop and Perfect Benchmark [5, 8, 25]. 10 pairs of two-dimensional array references with different direction vectors are observed to have subscripts formed induction variable. The proposed method is only applied to test those two-dimensional arrays with subscripts formed induction variable. It is found in the experiment that there is no case to satisfy the condition of the proposed method for Vector loop, TRFD and OCSI. It is very clear from the result shown in Table 1 that the proposed method could properly solve whether there are integer-valued solutions for two-dimensional arrays with subscripts formed induction variable.

We also implemented the Omega test and the Power test based on [4, 7] to compare their effects with that of the proposed method. The Omega test and Power test are applied to solve the same 10 pairs of two-dimensional arrays with symbolic coefficients. Clearly, from the result shown in Table 2, the Omega test and Power test could not be applied for determining whether there are integer-valued solutions for two-dimensional arrays with symbolic coefficients. Therefore, the results shown in Table 1 and Table 2 indicate that the precision of the proposed method
is slightly superior to that of the Omega test and the Power test.

Table 1. The result of solving whether there are integer-valued solutions for two-dimensional arrays with subscripts formed induction variable.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>The number of arrays tested</th>
<th>The number of integer-valued solution</th>
<th>The number of no integer-valued solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel loop</td>
<td>10</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Vector loop</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TRFD</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OCSI</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The study in [4] stated that (1) the cost of scanning array subscripts and loop bounds to build a dependence problem is typically 2 to 4 times of the copying cost (the cost of building a system of dependence equations) for the problem, and (2) the dependence analysis cost for more than half of simple arrays tested is typically 2 to 4 times of the copying cost, but the dependence analysis cost for other simple arrays and all of the regular, convex and complex arrays tested is more than 4 times of the copying cost. Based on such results we can figure out that, for simple arrays, the analysis cost of data dependence for a parallelizing/vectorizing compiler occupies generally about 29% to 57% of total compiling time. But, for complex arrays, the analysis cost of dependence testing takes more than 57% of total compiling time. Therefore, enhancing on dependence testing performance may result in significant improvement on compiling performance of a parallelizing/vectorizing compiler.

4 Conclusions

The front-end of a parallelizing compiler can easily recognize induction variable that forms subscripts of two-dimensional arrays. Induction variable can be replaced by a linear function in the loop’s index-variable after the front-end of a parallelizing compiler finishes the processing of induction variable substitution. The proposed method can offer data dependence analysis for arrays with references formed by induction variable. Depending on the application domains, we suggest to applying the proposed method together with the front-end
of a parallelizing compiler to provide data dependence analysis for arrays with references formed by induction variable.

5 References


