One-dimensional I test and direction vector I test with array references by induction variable

Minyi Guo
School of Computer Science and Engineering,
University of Aizu, Aizu-Wakamatsu City,
965 8580 Fukushima, Japan

State Key Lab for Novel Software Technology,
Nanjing University, PRC
Fax: +81 242 37 2744 E-mail: minyi@u-aizu.ac.jp

Weng-Long Chang*
Department of Computer Science and Information Engineering,
National Kaohsiung University of Applied Sciences,
415 Chien Kung Road, 807 Kaohsiung, Taiwan
E-mail: changwl@ mail.stut.edu.tw
*Corresponding author

Jian Lu
State Key Lab for Novel Software Technology,
Nanjing University, PRC
E-mail: lj@nju.edu.cn

Minglu Li
Department of Computer Science and Engineering,
Shanghai JiaoTong University,
Shanghai, China
E-mail: li-ml@cs.stut.edu.cn

Abstract: In this paper, theoretical aspects to demonstrate the accuracy of the Interval Test (the I test and the direction vector I test) to be applied for resolving the problem stated above is presented. Also, it is proved from the proposed theoretical aspects that under a specific direction vector $\vec{\theta} = (=_1, \ldots, =_d)$ there are *integer-valued* solutions for one-dimensional arrays with subscripts formed by induction variable and under other specific direction vectors there are *no* integer-valued solutions. Experiments with benchmarks, cited from Parallel loop, Vector loop and TRFD (Perfect benchmark), reveal that our framework can properly enhance the precision of data dependence analysis for one-dimensional arrays with subscripts mentioned above.

Keywords: parallelising/vectorising compilers; data dependence analysis; loop parallelisation; loop vectorisation; automatic loop transformation.

Reference to this paper should be made as follows: Guo, M., Chang, W-L., Lu, J. and Li, M. (2005) 'One-dimensional I test and direction vector I test with array references by induction variable', *Int. J. High Performance Computing and Networking*, Vol. 3, No. 4, pp.219–226.

Biographical notes: Minyi Guo is currently an Associate Professor at the Department of Computer Software, at the University of Aizu, Japan. Guo has served as programme committee or organising committee member for many international conferences. He is the Editor-in-Chief of the *Journal of Embedded Systems*. He is also on the editorial board of the *International Journal of High Performance Computing and Networking*, the *Journal of Embedded Computing*, the *Journal of Parallel and Distributed Scientific and Engineering Computing* and the *International Journal of Computer and Applications*. Guo's research interests include parallel and distributed processing, parallelising compilers, data parallel languages, data mining, molecular computing and software engineering. He is a member of the ACM, IEEE, and IEICE.

Weng-Long Chang received his PhD degree in computer science and information engineering from the National Cheng Kung University, Taiwan, Republic of China, in 1999. He is currently an Assistant Professor at the Southern Taiwan University of Technology. His research interests include molecular computing and languages and compilers for parallel computing.

Jian Lu received his BS degree, MS degree, and PhD degree from Nanjing University of PR China in 1982, 1984, 1988, respectively. Currently, he is a Professor at Department of Computer Science and Technology, Vice Director of the Institute of Computer Software at Nanjing University and Director of State Key Laboratory of Novel Software Technology of P.R. China. His research interests include formal methods, mobile agent technology and software architecture.

Dr. Minglu Li is a Full Professor and Vice Char of Department of Computer Science and Engineering of Shanghai Jiao Tong University (SJTU). Now, he is also Director of Grid Computing Center of SJTU, a member of Expert Committee of ChinaGrid Program of Ministry of Education of China, PI of ShanghaiGrid of Science and Technology Commission of Shanghai Municipality(STCSM), an Associate Editor of *International Journal of Grid and Utility Computing*, an member of editorial board of *International Journal of Web Services Research*, a member of executive committee of Technical Committee on *Services Computing of IEEE.*

1 INTRODUCTION

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. This task becomes even more important in distributed memory systems where, in addition to accomplishing a high parallelism of computation and low communication overhead among the processors, it is essential to develop a new analysis technique for data dependence.

There are several well-known data dependence analysis algorithms applicable for *one-dimensional* arrays under *constant* bounds or *variable* bounds: the GCD test (Banerjee, 1988, 1993, 1997), Banerjee's method (Banerjee, 1988, 1993, 1997), the I test and the direction vector I test (Kong et al., 1991; Niedzielski and Psarris, 1999; Psarris et al., 1991, 1993; Psarris and Kyriakopoulos, 1999), the extended I test (Chang and Chu, 1998), the generalised direction vector I test (Chang and Chu, 2001) and the interval reduction test (Huang and Yang, 2000). There are also several well-known data dependence analysis algorithms applicable for *multi-dimensional coupled* arrays under *constant* bounds or *variable* bounds: the generalised GCD test (Banerjee, 1988, 1993, 1997), the Lambda test (Li et al., 1990), the generalised Lambda test (Chang et al., 1999), the multi-dimensional I test (Chang et al., 2001), the multi-dimensional direction vector I test (Chang et al., 2002), the Power test (Wolfe and Tseng, 1992) and the Omega test (Pugh, 1992). There are several well-known data dependence analysis algorithms applicable for arrays with linear subscripts with symbolic coefficients, or with non-linear subscripts under *symbolic* bounds: the infinity Banerjee test (Petersen, 1993), the Range test (Blume and Eigenmann, 1998), the infinity Lambda test (Chang and Chu, 2000), the access range test (Paek, 1997; Hoeflinger, 1998) and one analysis method for pointers and induction variables (Wu, 2001).

In this paper, we propose a sophisticated technique of data dependence analysis. This approach is to test dependence if there are *integer-valued* solutions for one-dimensional arrays with subscripts formed by induction variable. Without direction vectors, there are integer-valued solutions for one-dimensional arrays with subscripts formed by induction variable. Furthermore, it is also shown that under a specific direction vector $\vec{\theta} = (=_1, \ldots, =_d)$ there are integer-valued solutions for one-dimensional arrays with subscripts formed by induction variable.

Shen et al. (1992) had indicated that in real programmes one-dimensional arrays with subscripts formed by induction variable occur quite frequently. A *d*-nested loop accessing a one-dimensional array with subscripts formed by induction variable is shown in Figure 1.

```
      K=0
      FOR I_1 = L_1 TO U_1
        ⋮
        FOR I_d = L_d TO U_d
          K=K+z
S:        A(K+C) = ⋯
S:        ⋯ = A(K+C)⋯
            ⋮
        ENDFOR
          ⋮
ENDFOR
```

Figure 1 *An example of a nested loop with induction variable*

Induction variable is a one scalar integer variable, which is used in a loop, to simulate loop's index variables: it is incremented or decremented by a constant amount in each iteration. Every induction variable can be replaced by a linear function in loop's index variables. The transformation, which does so, is called induction variable substitution. Since the variable K in Figure 1 is one induction variable, it can be replaced $z\times((I_1-L_1)\times(\prod_{p_1=2}^{d}(U_{p_1}-L_{p_1}+1))+(I_2-L_2)\times(\prod_{p_2=3}^{d}(U_{p_2}-L_{p_2}+1)+\cdots+(i_d-L_d+1))$ where d is the number of common loops and z is one integer variable (Banerjee, 1988, 1993, 1997; Paek, 1997; Hoeflinger, 1998). 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.

$K = 0$

$FOR\ \ I_1 = L_1$ TO U_1

⋮

$FOR\ \ I_d = L_d$ TO U_d

⋯

$S_1 : A(z\times((I_1-L_1)\times(\prod_{P_1=2}^{d}(U_{P_1}-L_{P_1}+1))+(I_2-L_2)\times(\prod_{P_2=3}^{d}(U_{P_2}-L_{P_2}+1)+\cdots+(I_d-L_d+1))+C)=\cdots$

$S_2 : \quad \cdots = A(z\times((I_1-L_1)\times(\prod_{P_1=2}^{d}(U_{P_1}-L_{P_1}+1))+(I_2-L_2)\times(\prod_{P_2=3}^{d}(U_{P_2}-L_{P_2}+1))+\cdots+(I_d-L_d+1))+C)\cdots$

⋮

ENDFOR

⋮

ENDFOR

⋮

Figure 2 *The transformed loop after induction variable substitution for the induction variable K*

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

The problem of determining whether there exists dependence for the array A between S_1 and S_2 in Figure 2 can be reduced to that of checking whether one system of a linear equation with n unknown variables has a simultaneous integer solution, which satisfies the constraints for each variable in the system. It is assumed that one linear equation in a system is written as:

$$\begin{aligned} z\times(&(X_1-X_2)\times((U_2-L_2+1)\times\cdots\times(U_d-L_d+1)) \\ &+(X_3-X_4)\times((U_3-L_3+1)\times\cdots\times(U_d-L_d+1)) \\ &+\cdots+(X_{2j-1}-X_{2j})\times((U_{j+1}-L_{j+1}+1)\times\cdots \\ &\times(U_d-L_d+1))+\cdots+(X_{2d-3}-X_{2d-2}) \\ &\times(U_d-L_d+1)+(X_{2d-1}-X_{2d}))=0, \end{aligned} \tag{1}$$

where each L_k and each U_k are an integer variable and are, respectively, one lower bound and one upper bound for the k-the loop for ($1 \le k \le d$). Because z is the greatest common divisor for all the coefficients in the left-hand side of equation (1), all the coefficients in equation (1) are divided by z and equation (1) is rewritten as

$$\begin{aligned} (X_1-X_2)&\times((U_2-L_2+1)\times\cdots\times(U_d-L_d+1)) \\ &+(X_3-X_4)\times((U_3-L_3+1)\times\cdots\times(U_d-L_d+1)) \\ &+\cdots+(X_{2j-1}-X_{2j})\times((U_{j+1}-L_{j+1}+1)\times\cdots\times(U_d-L_d+1)) \\ &+\cdots+(X_{2d-3}-X_{2d-2})\times(U_d-L_d+1)+(X_{2d-1}-X_{2d})=0. \end{aligned} \tag{2}$$

It is postulated that the constraints to each variable in equation (2) are represented as

$$L_k \le X_{2k-1} \text{ and } X_{2k} \le U_k, \tag{3}$$

where ($1 \le k \le d$). Let us use an example to make clear the illustrations stated above. Consider the nested do-loop in Figure 3. The variable K in Figure 3(a) is incremented by *one* in each iteration in the nested loop. So it is a one induction variable. After finishing the processing of induction variable substitution for the induction variable K, the result is shown in Figure 3(b). In Figure 3(b), the lower and upper bounds of the first (outer) loop and the second (inner) loop are, respectively, 1 and 10. Therefore, the bounds of the do-loop are constants. This do-loop executes 100 iterations by consecutively assigning the values 1, 2, …, 10 to J and I and executing the body (the main statement S) exactly once in each iteration. The net effect of the do-loop execution is then the ordered execution of the statements:

$A(1) = B(1)$

$A(2) = B(2)$

$\vdots$

$A(100) = B(100).$

```
(a)
      K=0
   DO J = 1, 10
     DO I =1, 10
        K = K + 1
S:   A(K) = B(K)
     ENDDO
   ENDDO

(b)
      K=0
   DO J = 1, 10
     DO I =1, 10
        K= I+10*(J-1)
S:   A(I+10*(J-1)) = B( I+10*(J-1))
     ENDDO
   ENDDO
```

(a) (b)

Figure 3 *A nested do-loop in Fortran (a) a nested do-loop with induction variable and (b) the transformed loop after induction variable substitution for the induction variable K*

To ascertain whether two references to the one-dimensional array A may refer to the same element of A, it has to be checked if the following linear equation $10 \times X_1 - 10 \times X_2 + X_3 - X_4 = 0$ has a simultaneous integer solution under the constant bounds $1 \le X_1$, $X_2 \le 10$, and $1 \le X_3$, $X_4 \le 10$. It is well known that the problem of finding *integer-valued* solutions to a system of linear equations is NP-hard. Therefore, in practice, most well-known data dependence analysis algorithms are used to solve as many particular cases of this problem as possible.

The rest of this paper proffers the following: In Section 2, the summary accounts of data dependence and interval equation are presented. In Section 3, the theoretical aspects of determining whether there are *integer-valued* solutions for one linear equation (2) with *symbolic* coefficients under the bounds of equation (3) are described. Experimental results are given in Section 4. Finally, brief conclusions are drawn in Section 5.

2 BACKGROUND

The summary accounts of data dependence and interval equation are introduced briefly in this section.

2.1 Data dependence

It is assumed that S_1 and S_2 are two statements within the loop in Figure 2. The loop is presumed to contain d common loops. Statements S_1 and S_2 are postulated to be embedded in d common loops. An array A is supposed to appear simultaneously within statements S_1 and S_2. If S_2 uses the element of the array A defined first by S_1, then S_2 is true-dependent on S_1. If S_2 defines the element of the array A used first by S_1, then S_2 is anti-dependent on S_1. If S_2 redefines the element of the array A defined first by S_1, then S_2 is output-dependent on S_1.

Each iteration of a loop nested is identified by an iteration vector, whose elements are the values of the iteration variables for that iteration. For example, the instance of the statement S_1 during iteration $\vec{i} = (i_1, \ldots, i_d)$ is denoted $S_1(\vec{i})$; the instance of the statement S_2 during iteration $\vec{j} = (j_1, \ldots, j_d)$ is denoted $S_2(\vec{j})$. If $(i_1, \ldots, i_d)$ is identical to $(j_1, \ldots, j_d)$ or $(i_1, \ldots, i_d)$ precedes $(j_1, \ldots, j_d)$ lexicographically, then $S_1(\vec{i})$ is said to precede $S_2(\vec{j})$, denoted $S_1(\vec{i}) < S_2(\vec{j})$. Otherwise, $S_2(\vec{j})$ is said to precede $S_1(\vec{i})$, denoted $S_1(\vec{i}) > S_2(\vec{j})$. In the following, Definition 1 defines direction vectors.

Definition 1: A vector of the form $\vec{\theta} = (\theta_1, \ldots, \theta_d)$ is termed as a direction vector**.** The direction vector $(\theta_1, \ldots, \theta_d)$ is said to be the direction vector from $S_1(\vec{i})$ to $S_2(\vec{j})$ if for $(1 \le k \le d)$ $i_k \theta_k j_k$, i.e., the relation θ_k is one element in a set $\{<,=,>\}$.

2.1 Interval equation

A linear equation with *symbolic* coefficients under the bounds of equation (3) will be said to be *integer* solvable if the equation has an *integer-valued* solution satisfying the bounds of each variable. Definitions 2 and 3, respectively, define integer interval and interval equation (Kong et al., 1991; Psarris et al., 1993).

Definition 2: Let $[\alpha_1, \alpha_2]$ represent the integer intervals from α_1 to α_2, i.e., the set of all integers between α_1 and α_2.

Definition 3: Let $a_1, \ldots, a_{n-1}, a_n$, L and U be integers. A linear equation

$$a_1X_1 + a_2X_2 + \ldots + a_{n-1}X_{n-1} + a_nX_n = [L, U], \tag{4}$$

which is referred to as an interval equation will be used to denote the set of ordinary equations consisting of:

$$\begin{aligned} a_1X_1 + a_2X_2 + \ldots + a_{n-1}X_{n-1} + a_nX_n &= L \\ a_1X_1 + a_2X_2 + \ldots + a_{n-1}X_{n-1} + a_nX_n &= L+1 \\ &\vdots \\ a_1X_1 + a_2X_2 + \ldots + a_{n-1}X_{n-1} + a_nX_n &= U. \end{aligned}$$

An interval equation (4) will be said to be integer solvable if one of the equations in the set, which it defines, is integer solvable. The immediate way to determine this is to test if an integer in between L and U is divisible by the GCD of the coefficients of the left-hand-side terms. If $L > U$ in an interval equation (4), then there are no integer solutions for this interval equation. If the expression on the left-hand side of an interval equation (4) is reduced to zero items, in the processing of testing, then the interval equation (4) will be said to be integer solvable if and only if $L \le 0 \le U$.

3 DETERMINING INTEGER-VALUED SOLUTIONS FOR ONE-DIMENSIONAL ARRAYS WITH SYMBOLIC COEFFICIENTS UNDER SYMBOLIC BOUNDS

Given the data dependence problem of one-dimensional arrays with *symbolic* coefficients and symbolic bounds, we want to find that

- under what conditions there are *integer-valued* solutions
- under what conditions there are *no integer-valued* solutions.

We start to discuss from the case without direction vectors for convenience of presentation.

3.1 Determine integer-valued solutions for one-dimensional arrays with symbolic coefficients under symbolic bounds without direction vector

One linear equation (2) with *symbolic* coefficients is deduced from determining whether there exists dependence for the array A between S_1 and S_2 in Figure 2. One bound (3) is derived from lower and upper bounds of loop index variables in Figure 2. If there is no *integer-valued* solution for one linear equation (2) with *symbolic* coefficients under the limits of equation (3), then there is no dependence. Otherwise, there is dependence for one linear equation (2) with *symbolic* coefficients under the limits of equation (3). Theorem 1 is presented to show that there are integer-valued solutions for one linear equation (2) with *symbolic* coefficients under the limits of equation (3).

Theorem 1: *There are integer-valued solutions for one linear equation (2) with symbolic coefficients under the limits of equation (3).*

Proof:

1 From the I test in Kong et al. (1991), one linear equation (2) with *symbolic* coefficients can be rewritten as the following interval equation:

$$\begin{aligned}&(X_1-X_2)\times((U_2-L_2+1)\times\cdots\times(U_d-L_d+1))\\&+(X_3-X_4)\times((U_3-L_3+1)\times\cdots\times(U_d-L_d+1))\\&+\cdots+(X_{2j-1}-X_{2j})\times((U_{j+1}-L_{j+1}+1)\times\cdots\\&\times(U_d-L_d+1))+\cdots+(X_{2d-3}-X_{2d-2})\\&\times(U_d-L_d+1)+(X_{2d-1}-X_{2d})=[0,0].\end{aligned}\tag{5}$$

2 According to the I test, the term $-X_{2d}$ in the interval equation (5) is moved to the right-hand side to gain the new interval equation:

$$\begin{aligned}&(X_1-X_2)\times((U_2-L_2+1)\times\cdots\times(U_d-L_d+1))\\&+(X_3-X_4)\times((U_3-L_3+1)\times\cdots\times(U_d-L_d+1))\\&+\cdots+(X_{2j-1}-X_{2j})\times((U_{j+1}-L_{j+1}+1)\times\cdots\\&\times(U_d-L_d+1))+\cdots+(X_{2d-3}-X_{2d-2})\times(U_d-L_d+1)\\&+X_{2d-1}=[L_d,U_d].\end{aligned}\tag{6}$$

3 In light of the I test, the term X_{2d-1} in the interval equation (6) is moved to the right-hand side to gain the new interval equation.

4 Because (U_d-L_d+1) is the greatest common divisor for all the coefficients in the left-hand side of the interval equation (7), all the coefficients in the interval equation (7) are divided by (U_d-L_d+1) and the new interval equation is obtained:

$$\begin{aligned}&(X_1-X_2)\times((U_2-L_2+1)\times\cdots\times(U_d-L_d+1))\\&+(X_3-X_4)\times((U_3-L_3+1)\times\cdots\times(U_d-L_d+1))\\&+\cdots+(X_{2j-1}-X_{2j})\times((U_{j+1}-L_{j+1}+1)\times\cdots\\&\times(U_d-L_d+1))+\cdots+(X_{2d-3}-X_{2d-2})\\&\times(U_d-L_d+1)=[L_d-U_d,U_d-L_d].\end{aligned}\tag{7}$$

$$\begin{aligned}&(X_1-X_2)\times((U_2-L_2+1)\times\cdots\times(U_d-L_d+1))\\&+(X_3-X_4)\times((U_3-L_3+1)\times\cdots\times(U_d-L_d+1))\\&+\cdots+(X_{2j-1}-X_{2j})\times((U_{j+1}-L_{j+1}+1)\times\cdots\\&\times(U_d-L_d+1))+\cdots+(X_{2d-3}-X_{2d-2})\times(U_d-L_d+1)\\&=\left[\left\lceil\frac{L_d-U_d}{U_d-L_d+1}\right\rceil,\left\lfloor\frac{U_d-L_d}{U_d-L_d+1}\right\rfloor\right]=[0,0].\end{aligned}\tag{8}$$

5 Repeat the processing of step (2)–(4) until the term in left-hand side of the interval equation is reduced to zero item. Therefore, we will obtain the new interval equation $0=[L_1-U_1,U_1-L_1]$. Because $U_1\geq L_1$, $L_1-U_1\leq 0\leq U_1-L_1$ is true. Thus, it is at once derived that there are *integer-valued* solutions for one linear equation (2) with *symbolic* coefficients under the limits of equation (3). □

The following example is used to explain the power of Theorem 1. Consider the do-loop in Figure 4(a). The front-end of one parallel compiler can recognise that the variable K is one induction variable and the subscript of the array A is formed by the induction variable K. After finishing the processing of induction variable substitution for the induction variable K, the result is shown in Figure 4(b). Because the coefficient of the subscript of the array A is an *unknown* variable (at compile time), the Power test and the Omega test cannot be applied to deal with the problem. However, in light of Theorem 1, it is at once concluded that there exists output dependence for the array A under without considering any direction vector. Therefore, it is indicated from the result that the precision of Theorem 1 is superior to that of the Omega test and the Power test.

```
   K=0
   DO J = 1, M
     DO I =1, N
       K = K + 1
S:     A(K) = B(K)

     ENDDO
   ENDDO
```
(a)

```
   K=0
   DO J = 1, M
     DO I =1, N
       K= I + N * (J-1)
S:     A(I + N * (J-1)) =
       B(I + N *(J-1))
     ENDDO
   ENDDO
```
(b)

Figure 4 *An example of do-loop in Fortran program (a) a nested do-loop with induction variable and (b) the transformed loop after induction variable substitution for the induction variable K*

3.2 Determine integer-valued solutions for one-dimensional arrays with symbolic coefficients under symbolic bounds and direction vector

From Theorem 1, it is very clear that there are integer-valued solutions for one linear equation (2) with *symbolic* coefficients under the limits of equation (3). Next, under a specific given *direction vector* and the limits of equation (3), whether there are *integer-valued* solutions for one linear equation (2) with *symbolic* coefficients will be discussed. The following theorems are proposed to show that

- under what a specific direction vector there are *no integer-valued* solutions
- under what a specific direction vector there are *integer-valued* solutions.

Theorem 2: *There are no integer-valued solutions for one linear equation (2) with symbolic coefficients under the limits of equation (3) and a specific direction vector* $(<, *, \ldots, *)_d$, *where d is the number of common loops and '*' is any one of* $\{<, =, >\}$.

Proof:

1 From the I test and the direction vector I test in Kong et al. (1991) and Psarris et al. (1993), one linear equation (2) with *symbolic* coefficients can be rewritten as the following interval equation:

$$\begin{aligned}&(X_1 - X_2)\times((U_2 - L_2 + 1)\times\cdots\times(U_d - L_d + 1))\\&\quad + (X_3 - X_4)\times((U_3 - L_3 + 1)\times\cdots\times(U_d - L_d + 1))\\&\quad + \cdots + (X_{2j-1} - X_{2j})\times((U_{j+1} - L_{j+1} + 1)\\&\quad \times\cdots\times(U_d - L_d + 1)) + \cdots + (X_{2d-3} - X_{2d-2})\\&\quad \times(U_d - L_d + 1) + (X_{2d-1} - X_{2d}) = [0,0].\end{aligned} \tag{9}$$

2 Repeat the processing of step (2) to step (4) in Theorem 1, until the term in left-hand side of the interval equation is reduced to two items. Therefore, we will obtain the new interval equation:

$$(X_1 - X_2) = [0,0]. \tag{10}$$

3 Because the direction vector is '<' for the terms X_1 and X_2, according to the direction vector I test in Psarris et al. (1993), the two terms X_1 and X_2 are moved to the right-hand side. Therefore, we obtain the new interval equation $0 = [1, U_1 - L_1]$. Because $1 \le 0 \le U_1 - L_1$ is false, it is, therefore, inferred that there are *no integer-valued* solutions for one linear equation (2) with *symbolic* coefficients under the limits of equation (3) and a specific direction vector $(<, *, \ldots, *)_d$. □

Theorem 3: *There are no integer-valued solutions for one linear equation (2) with symbolic coefficients under the limits of equation (3) and a specific direction vector* $(<, *, \ldots, *)_d$, *where d is the number of common loops and '*' is any one of* $\{<, =, >\}$.

Proof: Similar to Theorem 2. □

Theorem 4: *There are no integer-valued solutions for one linear equation (2) with symbolic coefficients under the limits of equation (3) and a specific direction vector* $(=, \theta_2, \ldots, \theta_{d-1}, \theta_d)_d$, *where d is the number of common loops and* θ_k *is any element of* $\{<, =, >\}$ *for* $2 \le k \le d-1$ *and* θ_k *is any element of* $\{<, >\}$.

Proof: Similar to Theorem 2. □

Theorem 5: *There are integer-valued solutions for one linear equation (2) with symbolic coefficients under the limits of equation (3) and a specific direction vector* $(=, =, \ldots, =)_d$, *where d is the number of common loops.*

Proof: Similar to Theorem 2. □

We now use the do-loop in Figure 4 to explain the power of Theorem 2 to Theorem 5. Consider the do-loop in Figure 4(a) and Figure 4(b). Because the coefficient of the subscript of the array A is an *unknown* variable (at compile time), the Power test and the Omega test cannot be applied to deal with the problem under any given *direction vectors*. However, in light of Theorem 5, it is right away inferred that there are integer-valued solutions for the array A under a specific direction vector (=, =). According to Theorem 2 to Theorem 4, it is at once concluded that there are no integer-valued solutions for the array A under other specific direction vectors. Therefore, the front-end of one parallel compiler at once derived that there only exists loop-independence output dependence for the do-loop. Because the source statement and sink statement to the dependence relation is the same statement, the dependence relation can be ignored. The do-loop can be executed in parallel mode. Hence, it is indicated from the result that the precision of Theorem 5 is superior to that of the Omega test and the Power test.

3.3 Extending symbolic subscript formed by induction variable

The expression '$K + C$' for the array A in Figure 1 to '$a \times K + C$', where a is an integer variable and is not equal to zero can be easily extended. Since the variable K in Figure 1 is one induction variable, it can be replaced by

$$\begin{aligned}&z\times((I_1 - L_1)\times(\textstyle\prod_{p_1=2}^{d}(U_{p_1} - L_{p_1} + 1)) + (I_2 - L_2)\\&\quad\times(\textstyle\prod_{p_2=3}^{d}(U_{p_2} - L_{p_2} + 1)) + \cdots + (i_d - L_d + 1))\end{aligned}$$

where d is the number of common loops and z is one integer variable (Banerjee, 1997; Paek, 1997; Hoeflinger, 1988).

Therefore, the extended expression of the array A is transformed into the code in Figure 5, after finishing the processing of induction variable substitution.

$K = 0$

$FOR\ I_1 = 1$ TO U_1

$\vdots$

$FOR\ I_d = 1$ TO U_d

$\cdots$

$$S_1 : A(a\times z\times((I_1 - L_1)\times(\prod_{P_1=2}^{d}(U_{P_1} - L_{P_1} + 1)) + (I_2 - L_2)\times (\prod_{P_2=3}^{d}(U_{P_2} - L_{P_2} + 1)) + \cdots + (I_d - L_d + 1)) + C) = \cdots$$

$$S_2 : \quad \cdots = A(a\times z\times((I_1 - L_1)\times(\prod_{P_1=2}^{d}(U_{P_1} - L_{P_1} + 1)) + (I_2 - L_2)\times (\prod_{P_2=3}^{d}(U_{P_2} - L_{P_2} + 1)) + \cdots + (I_d - L_d + 1)) + C)\cdots$$

$\vdots$

$ENDFOR$

$\vdots$

$ENDFOR$

$\vdots$

Figure 5 *The result is obtained after finishing the processing of induction variable substitution for the extended expression of the array A*

The problem of determining whether there exists dependence for the array A between S_1 and S_2 in Figure 5 is actually equal to that of checking whether there are integer-valued solutions for one *new* linear equation (11) under the constraints of equation (3). It is assumed that the new linear equation (11) is written as

$$\begin{aligned} &a\times z\times((X_1 - X_2)\times((U_2 - L_2 + 1)\times\cdots\times(U_d - L_d + 1)) \\ &\quad + (X_3 - X_4)\times((U_3 - L_3 + 1)\times\cdots\times(U_d - L_d + 1)) \\ &\quad + \cdots + (X_{2j-1} - X_{2j})\times((U_{j+1} - L_{j+1} + 1)\times\cdots \\ &\quad \times(U_d - L_d + 1)) + \cdots + (X_{2d-3} - X_{2d-2}) \\ &\quad \times(U_d - L_d + 1) + (X_{2d-1} - X_{2d})) = 0, \end{aligned} \tag{11}$$

where each L_k and each U_k are an integer variable and are, respectively, one lower bound and one upper bound for the k-th loop for $1 \le k \le d$. Because $a \times z$ is not equal to zero, all the coefficients in equation (11) are divided by $a \times z$ and equation (11) is exactly equal to equation (2). Therefore, Theorem 5 also can be applied to deal with whether there is dependent relation for the array A between S_1 and S_2 in Figure 5.

4 EXPERIMENTAL RESULTS

The proposed theorems have been tested and experimented on the codes abstracted from three numerical packages: Parallel Loop, Vector Loop and TRFD (Perfect Benchmark) (Dongarra et al., 1991; Levine et al., 1991; Eigenman et al., 1988). One hundred and sixty four pairs of one-dimensional array references were observed to have subscripts formed by *induction variable*. The proposed theorems are only applied to test those one-dimensional arrays with subscripts formed by *induction variable*. It is very clear from the result shown in Table 1 that the proposed theorems could properly solve whether there are *definitive results* for one-dimensional arrays with subscripts formed by induction variable.

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

Benchmark	*The number of arrays tested*	*The number of integer-valued solutions*	*The number of no integer-valued solutions*
Parallel loop	144	36	108
Vector loop	10	2	8
TRFD	10	2	8

The Omega test and the Power test have been implemented based on Wolfe and Tseng (1992) and Pugh (1992), to compare their effects with that of the proposed theorems. The Omega test and the Power test were applied to resolve the same 164 pairs of one-dimensional arrays with symbolic coefficients. It is very clear from the result shown in Table 2 that the Omega test and the Power test could not be applied for determining whether there are *integer-valued* solutions for one-dimensional arrays with *symbolic* coefficients. Therefore, it is indicated from the results shown in Tables 1 and 2 that the precision of the proposed theorems is superior to that of the Omega test and the Power test.

Table 2 *The result of the Omega and Power tests for solving whether there are integer-valued solutions for 164 pairs of one-dimensional arrays with symbolic coefficients*

Dependence method	*The number of arrays tested*	*The number of integer-valued solution*	*The number of no integer-valued solutions*
The Omega test	164	–	–
The Power test	164	–	–

–: represents that the Omega and Power tests could not deal with them.

5 CONCLUSIONS

The front-end of a parallelising compiler can easily recognise induction variable that forms subscripts of one-dimensional arrays. Induction variable can be replaced by a linear function in form of loop's index-variable after the front-end of a parallelising compiler finishes the processing of induction variable substitution. This paper presents theoretical aspects to demonstrate the accuracy of the Interval Test (the I test and the direction vector I test) to be applied for resolving the problem stated above. Also, it is proved from the proposed theoretical aspects that under a specific direction vector $\vec{\theta} = (=_1, \ldots, =_d)$ there are *integer-valued* solutions for one-dimensional arrays with subscripts formed by induction variable and under other specific direction vectors there are *no* integer-valued solutions. Experiments with benchmarks, cited from Parallel loop, Vector loop and TRFD (Perfect benchmark), reveal that this framework can properly enhance the precision of data dependence analysis for one-dimensional arrays with subscripts mentioned above. The proposed theorems can improve the precision of data dependence analysis for one-dimensional arrays with references formed by induction variable. Depending on the application domains, it is suggested that the proposed theorems be applied together with the front-end of a parallelising compiler to provide data dependence analysis for one-dimensional arrays with references formed by induction variable.

ACKNOWLEDGEMENT

This work was partially supported by the 973 Program of China under Grant 2002CB312002 and NSFC of Jiangsu Province under Grant BK2002203.

REFERENCES

Banerjee, U. (1988) *Dependence Analysis for Supercomputing*, Kluwer Academic Publishers, Norwell, Massachusetts.

Banerjee, U. (1993) *Loop Transformations for Restructuring Compilers*: *The Foundations*, Kluwer Academic Publishers, Norwell, Massachusetts.

Banerjee, U. (1997) *Dependence Analysis*, Kluwer Academic Publishers, Norwell, Massachusetts.

Blume, W. and Eigenmann, R. (1998) 'Nonlinear and symbolic data dependence testing', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 9, No. 12, pp.1180–1194.

Chang, Chu and Wu, J. (1999) 'The generalized lambda test: a multi-dimensional version of banerjee's algorithm', *International Journal of Parallel and Distributed Systems and Networks*, Vol. 2, No. 2, pp.69–78.

Chang, W-L. and Chu, C-P. (1998) 'The extension of the I test', *Parallel Computing*, Vol. 24, pp.2101–2127.

Chang, W-L. and Chu, C-P. (2000) 'The infinity lambda test: a multi-dimensional version of Banerjee's infinity test', *Parallel Computing*, Vol. 26, No. 10, August, pp.1275–1295.

Chang, W-L. and Chu, C-P. (2001) 'The generalized direction vector I test', *Parallel Computing*, Vol. 27, No. 8, July, pp.1117–1144.

Chang, W-L., Chu, C-P. and Wu, J-H. (2001a) 'A multi-dimensional version of the I test', *Parallel Computing*, Vol. 27, No. 13, September, pp.1783–1799.

Chang, W-L., Chu, C-P. and Wu, J-H. (2002) 'A precise dependence analysis for multi-dimensional arrays under specific dependence direction', *Journal of System and Software*, Vol. 63, No. 2, pp.99–112.

Dongarra, J., Reinhardt, M.F. and Russell, S.J. (1991) 'Parallel loops – a test suite for parallelizing compilers: description and example results', *Parallel Computing*, Vol. 17, pp.1247–1255.

Eigenman, R., Hoeflinger, J. and Padua, D. (1998) 'On the automatic parallelization of the perfect benchmarks', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 1, January, pp.5–23.

Hoeflinger, J.P. (1998) *Inter-procedural Parallelization using Memory Classification Analysis*, PhD Thesis, University of Illinois at Urbana-Champaign.

Huang, T-C. and Yang, C-M. (2000) 'Data dependence analysis for array references', *The Journal of System and Software*, Vol. 52, pp.55–65.

Kong, X., Klappholz, D. and Psarris, K. (1991) 'The I test', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 2, No. 3, July, pp.342–359.

Levine, D., Callahan, D. and Dongarra, J. (1991) 'A comparative study of automatic vectorizing compilers', *Parallel Computing*, Vol. 17, pp.1223–1244.

Li, Z., Yew, P-C. and Zhu, C-Q. (1990) 'An efficient data dependence analysis for parallelizing compilers', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 1, No. 1, January, pp.26–34.

Niedzielski, D. and Psarris, K. (1999) 'An analytical comparison of the I-test and Omega test', *LCPC'99: Twelfth International Workshop on Languages and Compilers for Parallel Computing*, pp.251–270.

Paek, Y. (1997) *Compiling for Distributed Memory Multiprocessors based on Access Region Analysis*, PhD Thesis, University of Illinois at Urbana-Champaign.

Petersen, P.M. (1993) *Evaluation of Programs and Parallelizing Compilers Using Dynamic Analysis Techniques*, PhD Thesis, University of Illinois at Urbana-Champaign, January.

Psarris, K. and Kyriakopoulos, K. (1999) 'Data dependence testing in practice', *Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques*, pp.264–273.

Psarris, K., Klappholz, D. and Kong, X. (1991) 'On the accuracy of the banerjee test', *Journal of Parallel and Distributed Computing*, Vol. 12, No. 2, June, pp.152–158.

Psarris, K., Kong, X. and Klappholz, D. (1993) 'The direction vector I test', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 4, No. 11, pp.1280–1290.

Pugh, W. (1992) 'A practical algorithm for exact array dependence analysis', *Communication of the ACM*, Vol. 35, No. 8, August, pp.102–114.

Shen, Z., Li, Z. and Yew, P-C. (1992) 'An empirical study of Fortran programs for parallelizing compilers', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 1, No. 3, July, pp.356–364.

Wolfe, M. and Tseng, C-W. (1992) 'The power test for data dependence', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 3, No. 5, September, pp.591–601.

Wu, P. (2001) *Analysis of Pointers and Induction Variable and Container Objects for Dependence Tests*, PhD Thesis, University of Illinois at Urbana-Champaign, UIUCDCS-R-2001-2209.