ELSEVIER Parallel Computing 26 (2000) 1275–1295

PARALLEL COMPUTING

www.elsevier.com/locate/parco

The infinity Lambda test: A multi-dimensional version of Banerjee infinity test

Weng-Long Chang, Chih-Ping Chu *

Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan, ROC

Received 15 February 1999; received in revised form 21 September 1999; accepted 20 December 1999

Abstract

The Banerjee infinity test accurately determines data dependence for one linear equation under symbolic limits and any given direction vectors (U. Banerjee, Depence Analysis for Supercomputing, Kluwer Academic Publishers, Norwell, MA, 1988; P.M. Petersen, Evaluation of programs and parallelizing compilers using dynamic analysis techniques, Ph.D. Thesis, University of Illinois at Urbana–Champaign, January 1993). For m linear equations with the same constraints, as each linear equation has to be tested separately, the Banerjee infinity test may generally lose the accuracy. In this paper, we proposed the infinity Lambda test – a multi-dimensional version of the Banerjee infinity test. The infinity Lambda test can be applied to deal with data dependence of coupled arrays with symbolic (*unknown* at compile time) bounds. Experiments with benchmark showed that the infinity Lambda test increases the success rate of the Lambda test by approximately 12%. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Parallelizing compilers; Vectorizing compilers; Data dependence analysis

1. Introduction

Coupled subscripts occur quite frequently in real programs in the light of an empirical study reported in [14]. The study shows that two-dimensional array references account for 36% and three-dimensional array references account for 7%

* Corresponding author. Tel.: +886-6-2757575, ext. 62527; fax: +886-6-2747076.
E-mail address: chucp@server2.iie.ncku.edu.tw (C.-P. Chu).

among array references examined. The study also shows that about 46% and 2% have coupled subscripts for examined two-dimensional and three-dimensional array references, respectively. Loops with *symbolic* (*unknown* at compile time) bounds also appear quite frequently in real programs according to an empirical study reported in [6]. The study indicates that 95% of loops checked have symbolic limits. The major findings from these data include: (1) multi-dimensional array references with unknown bounds are very common, (2) coupled subscripts with symbolic bounds emerge quite frequently, and (3) most of the coupled subscripts with unknown bounds appear in two-dimensional arrays. Hence, an efficient and precise dependence testing method for manipulating coupled expressions with *symbolic* bounds is very important.

The question of whether multi-dimensional arrays with linear coupled subscripts under symbolic loop bounds may be parallelized/vectorized depends upon the resolution of those multi-dimensional array aliases. The resolution of coupled multi-dimensional array aliases is to ascertain whether two references to a coupled multi-dimensional array in a general loop with symbolic loop limits may refer to the same element of that array. This problem in general case can be reduced to that of checking whether a system of m linear equations with n unknown variables has a simultaneous integer solution, which satisfies the constraints for each variable in the system. It is assumed that m linear equations in a system are written as

$$\begin{aligned} &a_{1,1}X_1 + a_{1,2}X_2 + \cdots + a_{1,n-1}X_{n-1} + a_{1,n}X_n + a_{1,0} = 0, \\ &\qquad\qquad\vdots \\ &a_{m,1}X_1 + a_{m,2}X_2 + \cdots + a_{m,n-1}X_{n-1} + a_{m,n}X_n + a_{m,0} = 0, \end{aligned} \tag{1.1}$$

where each $a_{i,j}$ is an *integer* for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$.

It is postulated that the constraints to each variable in (1.1) are represented as

$$P_{r,0} + \sum_{s=1}^{r-1} P_{r,s}X_s \leqslant X_r \leqslant Q_{r,0} + \sum_{s=1}^{r-1} Q_{r,s}X_s, \tag{1.2}$$

where $P_{r,0}$, $Q_{r,0}$, $P_{r,s}$ and $Q_{r,s}$ are either *loop-invariant variables* or *loop-invariable constants* for $1 \leqslant r \leqslant n$, but at least one of the $P_{r,0}$, $Q_{r,0}$, $P_{r,s}$ and $Q_{r,s}$ is a *loop-invariant variable* for the r, $1 \leqslant r \leqslant n$.

The bounds of a variable X_r are constants if each of $P_{r,0}$ and $Q_{r,\,0}$ is a loop-invariant constant and each of $P_{r,s}$ and $Q_{r,s}$ is 0. The bounds of a variable X_r are variables if each of $P_{r,0}$, $Q_{r,0}$, $P_{r,s}$ and $Q_{r,s}$ is a loop-invariant constant. The bounds of a variable X_r are symbolic if at least one of $P_{r,0}$, $Q_{r,0}$, $P_{r,s}$ and $Q_{r,s}$ is a loop-invariant variable.

There are several data dependence analysis algorithms for arrays with coupled subscripts in practice. The I test and the Direction Vector I test are a combination of the Banerjee inequalities and the GCD test [7,8]. They figure out integer solutions for one linear equation with constant bounds and given direction vectors. The Lambda test extends the *Banerjee inequalities* to allow m linear equations (1.1) under constant loop limits and any given direction vectors to be tested simultaneously [15]. The generalized Lambda test extends the Lambda test and the *Banerjee algorithm* to allow m linear equations (1.1) with variable loop limits as well as any given direction vectors to be checked simultaneously [5]. If the Lambda test and the generalized Lambda test determine no real solutions for m linear equations with both constant loop bounds and variable loop limits, then they conclude that there are no integer solutions for m linear equations under those constraints. Otherwise, the Lambda and generalized Lambda tests assume that there are integer solutions for m linear equations under those constraints. More precise results are achieved by judging the consistency of a linear system of equalities and inequalities. The Power test is a combination of Fourier–Motzkin variable elimination with an extension of Euclid's GCD algorithm [17,18]. The Omega test combines new methods for eliminating equality constraints with an extension of Fourier–Motzkin variable elimination to integer programming [11]. Though both of the methods gain more accurate outcomes, they have exponential worst-case time complexity.

The *Banerjee infinity test* (the *Banerjee inequalities* and the *Banerjee algorithm* for testing *single* dimension with *symbolic* limits) can handle the system of one linear equation under the limits of (1.2) and any given direction vectors [2,10]. For m linear equations (1.1) with the same constraints (1.2), each linear equation has to be tested separately. The Banerjee infinity test in this case may generally lose the accuracy in many practical cases.

In this paper, the Lambda test and the Banerjee infinity test are integrated to treat whether m linear equations (1.1) under the bounds of (1.2) and given direction vectors have relevant real-valued solutions. An algorithm called the infinity Lambda test has been implemented and several measurements have also been performed.

The rest of this paper is proffered as follows. In Section 2, the problem of data dependence subject to an arbitrary direction vector is reviewed. The summary accounts of the Lambda test, the generalized Lambda test and the Banerjee infinity test are presented. In Section 3, the theoretical aspects and the worst-case time complexity of the infinity Lambda test are described. Experiments with the infinity Lambda test and the results showing the advantages of the infinity Lambda test are given in Section 4. Finally, brief conclusions are drawn in Section 5.

2. Background

The concept of data dependence and the summary accounts of the Lambda test and the generalized Lambda test are mainly introduced in this section.

2.1. Data dependence

It is assumed that S_1 and S_2 are two statements within a general loop. The general loop is presumed to contain d common loops. Statements S_1 and S_2 are postulated to be embedded in $d+p$ loops and $d+q$ loops, respectively. An array A is supposed to appear simultaneously within statements S_1 and S_2. Each iteration of a general loop 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 $\vec{i} = (i_1, \ldots, i_d, \ldots, i_{d+p})$ is denoted $S_1(\vec{i})$; the instance of the statement S_2 during iteration $\vec{j} = (j_1, \ldots, j_d, \ldots, j_{d+q})$ is denoted $S_2(\vec{j})$. If $(i_1, \ldots, i_d, \ldots, i_{d+p})$ is identical to $(j_1, \ldots, j_d, \ldots, j_{d+q})$ or $(i_1, \ldots, i_d, \ldots, i_{d+p})$ precedes $(j_1, \ldots, j_d, \ldots, j_{d+q})$ 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 $S_1(\vec{i}) > S_2(\vec{j})$. If the instance of the statement $S_2(\vec{j})$ uses the element of the array A defined first by the instance of another statement $S_1(\vec{i})$, then $S_2(\vec{j})$ is true-dependent on $S_1(\vec{i})$. If the instance of the statement $S_2(\vec{j})$ defines the element of the array A used first by the instance of another statement $S_1(\vec{i})$, then $S_2(\vec{j})$ is anti-dependent on $S_1(\vec{i})$. If the instance of the statement $S_2(\vec{j})$ redefines the element of the array A defined first by the instance of another statement $S_1(\vec{i})$, then $S_2(\vec{j})$ is output-dependent on $S_1(\vec{i})$.

Definition 2.1. 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 $1 \leqslant k \leqslant d$, $i_k \theta_k j_k$, i.e., the relation θ_k is defined by

$$\theta_k = \begin{cases} < & \text{if } i_k < j_k, \\ = & \text{if } i_k = j_k, \\ > & \text{if } i_k > j_k, \\ * & \text{the relation of } i_k \text{ and } j_k \text{ can be ignored, i.e., can be any one of} \\ & \quad \{<,\ =,\ >\}. \end{cases}$$

2.2. The Lambda test

Coupled references are groups of reference positions sharing one or more index variables [9,17]. Geometrically, each linear equation in (1.1) defines a hyperplane π in R^n spaces. The intersection S of m hyperplanes corresponds to the common solutions to all linear equations in (1.1). Obviously, if S is empty then there is no data dependence. Inspecting whether S is empty is trivial in linear algebra. Constant loop

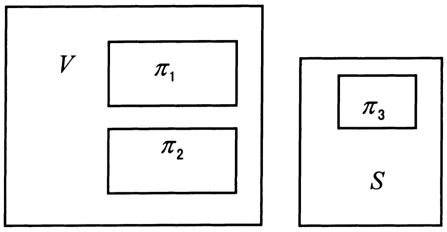

Fig. 1. A geometrical illustration.

bounds and any given direction vectors define a bounded convex set V in R^n. If any of the hyperplanes in (1.1) does not intersect V, then obviously S cannot intersect V. However, even if every hyperplane in (1.1) intersects V, it is still possible that S and V are disjoint. In Fig. 1 from [9], π_1 and π_2 are two such hyperplanes representing two linear equations in (1.1), each of which intersects V. But the intersection of π_1 and π_2 is outside of V. If a new hyperplane which contains the intersection of π_1 and π_2 is found but is disjoint from V, then S and V are immediately gathered not to intersect. In Fig. 1, π_3 is such a new hyperplane. If S and V are disjoint, then there exists a hyperplane which contains S and is disjoint from V. Furthermore, this hyperplane is a linear combination of hyperplanes in (1.1). On the other hand, if S and V intersect, then no such linear combination exists [2,17].

The Banerjee inequalities are first applied to test each hyperplane in (1.1). If every hyperplane intersects V, then the Lambda test is employed to simultaneously check every hyperplane. The Lambda test is an efficient and precise data dependence method to deal with (1.1) beneath V. The Lambda test is actually equivalent to a multi-dimensional version of the Banerjee inequality because it can determine simultaneous constrained real-valued solutions. The test forms linear combinations of coupled references that eliminate one or more instances of index variables when direction vectors are not considered. While direction vectors are considered, the Lambda test generates new linear combinations that use a pair of relative index variables. Simultaneous constrained real-valued solutions exist if and only if the Banerjee inequalities find solutions in all the linear combinations generated [9].

2.3. The generalized Lambda test

The Lambda test is an efficient and precise data dependence method to ascertain whether there exist data dependences for coupled arrays with constant bounds. For coupled arrays with variable limits, the Lambda test simply suggests that variable constraints should be replaced by the closest constant bounds but did not propose any practical scheme. In our previous work, we generalized the Lambda test [5]. The generalized Lambda test is exactly equivalent to a multi-dimensional version of the *Banerjee algorithm* that is applied towards determining whether there exist data dependences for coupled arrays with either variable limits or constant bounds under any given direction vectors. Experiments with benchmark showed that the generalized Lambda test increases the success rate of the original Lambda test by approximately 22% [5].

2.4. Banerjee's infinity test

In checking data dependence of a pair of linear coupled arrays the results must be conservative. One way of stating this criterion is that the dependence arcs for a given iteration space are always a subset of the arcs present in any iteration space that is a superset of the given iteration space. Thus, increasing the iteration space does not violate the conservative criteria.

Banerjee's infinity test is based on such a criteria. It is similar to Banerjee's inequalities and Banerjee's algorithm when it is applied to *rectangular* and *trapezoidal* iteration spaces. If the stride of the loop is a positive constant, then whenever a loop lower limit is not known $-\infty$ is assumed as its value, and whenever the loop upper limit is not known $+\infty$ is assumed as its value. Since the given iteration space is a subset of the new iteration space $[-\infty, +\infty]$, any dependence arcs found over the given iteration space will also be found in the iteration space $[-\infty, +\infty]$. Therefore, we suppose that f_i and g_i are the lower and the upper bound functions for the *i*th variable in (1.1) with *n*-variables under the constraints of (1.2) to which loop-invariant variables are included in the f_i and/or g_i, where $1 \leqslant i \leqslant n$. The original constraints (1.2) for the variable X_i in (1.1) are rewritten as

$$f_i(X_0, \ldots, X_{i-1}) \leqslant X_i \leqslant g_i(X_0, \ldots, X_{i-1}), \tag{2.1}$$

where the loop-invariant variables in $f_i(X_0, \ldots, X_{i-1})$ and $g_i(X_0, \ldots, X_{i-1})$ will be replaced by $-\infty$ or $+\infty$ depending on they are in f_i or g_i, for $1 \leqslant i \leqslant n$.

The only difference from the traditional Banerjee's inequalities and Banerjee's algorithm is that the arithmetic is done in the extended real number system [12] (i.e., $\Re \cup \{-\infty, +\infty\}$) and the traditional conventions are made on the operations (e.g., $n + \infty = \infty$, $n/\infty = 0$ if n is real).

Blume and Eigenmann [4] indicated that determining the cost of symbolic expression comparison is much more difficult than that of linear expression comparison. The worst-case performance of symbolic comparisons is exponential on the size of the expressions compared and upon the number of variables in the program. Petersen [10] pointed out that Banerjee's infinity test breaks dependences in more than 9% for array references with unknown loop bounds.

3. The infinity Lambda test

A data dependence problem is considered where coupled subscripts are linear in terms of loop indexes. Bounds for coupled subscripts are presumed to be symbolic loop limits. (Note: constant loop constraints are a special case of symbolic loop bounds, so symbolic loop limits actually contain constant loop constraints.) Dependence directions may also be given if required. Given the data dependence problem as specified, the infinity Lambda test examines a system of equalities and deduces whether the system has real-valued solutions. In this section, the theoretical aspects and the worst-case time complexity of the infinity Lambda test are provided.

It is presupposed that there are no redundant equations in (1.1). Otherwise, they can simply be eliminated. Furthermore, it is assumed that all array dimensions are coupled. Otherwise, (1.1) can be broken into several disjoint subsystems and partial solutions can be acquired for each subsystem and later merged together to form a complete solution. In practice, the number of coupled dimensions m is very small.

The theories of the proposed method are started to state from the case of $m = 2$, for the convenience of presentation.

3.1. The case of two-dimensional array references

In the case of two-dimensional array references, two equations in (1.1) are $F_1 = 0$ and $F_2 = 0$, where $F_i = a_{i,0} + a_{i,1}X_1 + \cdots + a_{i,n}X_n$ for $1 \leqslant i \leqslant 2$. A linear equation for convenience is directly referred as a hyperplane in R^n. From [9], an arbitrary linear combination of the two equations can be written as $\lambda_1 F_1 + \lambda_2 F_2 = 0$. The domain of (λ_1, λ_2) is the whole R^2 space. Let $F_{\lambda_1,\lambda_2} = \lambda_1 F_1 + \lambda_2 F_2$; that is $F_{\lambda_1,\lambda_2} = (\lambda_1 a_{1,0} + \lambda_2 a_{2,0}) + (\lambda_1 a_{1,1} + \lambda_2 a_{2,1})X_1 + \cdots + (\lambda_1 a_{1,n} + \lambda_2 a_{2,n})X_n$. F_{λ_1,λ_2} is viewed in two ways. With (λ_1, λ_2) fixed, F_{λ_1,λ_2} is a linear function of $(X_1, \ldots, X_n)$ in R^n. With $(X_1, \ldots, X_n)$ fixed, it is a linear function of (λ_1, λ_2) in R^2. Furthermore, the coefficient of each variable in F_{λ_1,λ_2} is a linear function of (λ_1, λ_2) in R^2, i.e., $\Psi^{(i)} = \lambda_1 a_{1,i} + \lambda_2 a_{2,i}$ for $1 \leqslant i \leqslant n$. The equation $\Psi^{(i)} = 0$, $1 \leqslant i \leqslant n$, is called a Ψ equation. Each Ψ equation corresponds to a line in R^2, which is called a Ψ line. Each Ψ line separates the whole space into two closed halfspaces $\Psi_i^+ = \{(\lambda_1, \lambda_2) \mid \Psi^{(i)} \geqslant 0\}$ and $\Psi_i^- = \{(\lambda_1, \lambda_2) \mid \Psi^{(i)} \leqslant 0\}$ that intersect at the Ψ line.

A nonempty set $C \subset R^m$ is a cone if $\varepsilon\vec{\lambda} \in C$ for each $\vec{\lambda} \in C$ and $\varepsilon \geqslant 0$. It is obvious that each cone contains the zero vector. Moreover, a cone that includes at least one nonzero vector $\vec{\lambda}$ must consist of the "ray" of $\vec{\lambda}$, namely $\{\varepsilon\vec{\lambda} \mid \varepsilon \geqslant 0\}$. Such cones can clearly be viewed as the union of rays. There are at most n Ψ lines which together divide R^2 into at most $2n$ regions. Each region contains the zero vector. Any one nonzero element $\vec{\lambda}$ and the zero vector in the region forms the ray of $\vec{\lambda}$, namely $\{\varepsilon\vec{\lambda} \mid \varepsilon \geqslant 0\}$. Therefore, each region can be viewed as the union of the rays. It is very obvious from the definition of a cone that each region is a cone [12,16].

In the following, Lemmas 3.1–3.3 are an extension of Lemmas 1–3 in [9], respectively; Definitions 3.1–3.3 are cited from [1,9] directly.

Lemma 3.1. *Suppose that an unbounded convex set V is defined simply by the limits of* (2.1). (*The dependence directions will be taken account of later.*) *If $F_{\lambda_1,\lambda_2} = 0$ intersects V for every (λ_1, λ_2) in every Ψ line, then $F_{\lambda_1,\lambda_2} = 0$ also intersects V for every (λ_1, λ_2) in R^2.*

Proof. (1) From the well-known intermediate value theorem, $F_{\lambda_1,\lambda_2} = 0$ intersects V if and only if $\min(F_{\lambda_1,\lambda_2}) \leqslant 0 \leqslant \max(F_{\lambda_1,\lambda_2})$.

(2) Since V is defined by the constraints of (2.1) and F_{λ_1,λ_2} is continuous on V, when (λ_1, λ_2) is fixed, there exist the minimum and maximum points for each variable in F_{λ_1,λ_2} such that $\min(F_{\lambda_1,\lambda_2}) = L_b$ and $\max(F_{\lambda_1,\lambda_2}) = U_b$. Thus we directly compute the limits for $F_{\lambda_1,\lambda_2}(X_1, \ldots, X_n) = \Psi^{(1)}X_1 + \cdots + \Psi^{(n)}X_n$. The method of calculating the constraints is actually equivalent to the Banerjee infinity test (*Banerjee's inequalities* and *Banerjee's algorithm* for testing single dimension with symbolic limits)

[2,10]. *Banerjee's algorithm* for checking loops with unknown bounds is first applied towards figuring out the constraints.

Let $h_n(X_1,\dots,X_n) = F_{\lambda_1,\lambda_2}(X_1,\dots,X_n)$. Then we have

$$
\begin{aligned}
&h_n(x_1,\dots,x_n)\\
&\quad \leqslant h_{n-1}(x_1,\dots,x_{n-1}),\\
&\qquad\quad \text{where } h_{n-1}(x_1,\dots,x_{n-1}) = \begin{cases} h_n(x_1,\dots,x_{n-1},\ g_n(x_1,\dots,x_{n-1})) \\ \quad \text{if the coefficient of } x_n > 0, \\ h_n(x_1,\dots,x_{n-1},\ f_n(x_1,\dots,x_{n-1})) \\ \quad \text{if the coefficient of } x_n < 0, \end{cases}\\
&\quad \vdots\\
&\quad \leqslant h_j(x_1,\ \dots,\ x_j),\\
&\qquad\quad \text{where } h_j(x_1,\dots,x_j) = \begin{cases} h_{j+1}(x_1,\dots,x_j,\ g_{j+1}(x_1,\dots,x_j)) \\ \quad \text{if the coefficient of } x_{j+1} > 0, \\ h_{j+1}(x_1,\dots,x_j,\ f_{j+1}(x_1,\dots,x_j)) \\ \quad \text{if the coefficient of } x_{j+1} < 0, \end{cases}\\
&\quad \vdots\\
&\quad \leqslant h_1(x_1) \leqslant U_b,\\
&\qquad\quad \text{where } U_b = \begin{cases} h_1(g_1(x_0)) \text{ if the coefficient of } x_1 > 0, \\ h_1(f_1(x_0)) \text{ if the coefficient of } x_1 < 0. \end{cases}
\end{aligned}
$$

Similarly, $L_b \leqslant F_{\lambda_1,\lambda_2}(X_1,\dots,X_n)$ can also be inferred. We thus achieve $L_b \leqslant F_{\lambda_1,\lambda_2}(X_1,\dots,X_n) \leqslant U_b$.

Similarly, *Banerjee's inequalities* for checking loops with unknown bounds can be also employed towards figuring out the same constraints. Therefore, we have

$$L_b = \sum_{1\leqslant j\leqslant n} LA_j + (\lambda_1 a_{1,0} + \lambda_2 a_{2,0}) \quad\text{and}\quad U_b = \sum_{1\leqslant j\leqslant n} UA_j + (\lambda_1 a_{1,0} + \lambda_2 a_{2,0}),$$

where

$$LA_j = \begin{cases} -\infty & \text{if } \lambda_1 a_{1,j} + \lambda_2 a_{2,j} \neq 0, \\ 0 & \text{if } \lambda_1 a_{1,j} + \lambda_2 a_{2,j} = 0 \end{cases} \quad\text{and}$$

$$UA_j = \begin{cases} +\infty & \text{if } \lambda_1 a_{1,j} + \lambda_2 a_{2,j} \neq 0, \\ 0 & \text{if } \lambda_1 a_{1,j} + \lambda_2 a_{2,j} = 0. \end{cases}$$

We obtain $L_b \leqslant F_{\lambda_1,\lambda_2}(X_1,\dots,\ X_n) \leqslant U_b$ from the processing of *Banerjee's inequalities*.

(3) We have $L_b = \min(F_{\lambda_1,\lambda_2}) \leqslant 0 \leqslant \max(F_{\lambda_1,\lambda_2}) = U_b$ for any point (λ_1,λ_2) on every Ψ line according to the assumption of the lemma. It is immediately concluded that $\min(F_{\lambda_1,\lambda_2}) \leqslant 0 \leqslant \max(F_{\lambda_1,\lambda_2})$ for every point (λ_1,λ_2) on the boundaries of each cone.

(4) Every point in each cone can be expressed as a linear combination of some points on the boundary of the same cone, as being a well-known fact in the convex

theory. Any point (λ_5, λ_6) in a cone is assumed to be capable of being represented as $(\varepsilon\lambda_1 + \tau\lambda_3,\ \varepsilon\lambda_2 + \tau\lambda_4)$, where (λ_1, λ_2) and (λ_3, λ_4) are points in the boundary of the cone and $\varepsilon \geqslant 0$ and $\tau \geqslant 0$. Because

$$\begin{aligned} F_{\lambda_5,\lambda_6}(X_1,\ldots,X_n) &= F_{\varepsilon\lambda_1+\tau\lambda_3,\varepsilon\lambda_2+\tau\lambda_4}(X_1,\ldots,X_n) \\ &= (\varepsilon\lambda_1 + \tau\lambda_3)(a_{1,1}X_1 + \cdots + a_{1,n}X_n) + (\varepsilon\lambda_2 + \tau\lambda_4) \\ &\quad \times (a_{2,1}X_1 + \cdots + a_{2,n}X_n) \\ &= (\varepsilon\lambda_1 a_{1,1} + \varepsilon\lambda_2 a_{2,1})X_1 + \cdots + (\varepsilon\lambda_1 a_{1,n} + \varepsilon\lambda_2 a_{2,n})X_n \\ &\quad + (\tau\lambda_3 a_{1,1} + \tau\lambda_4 a_{2,1})X_1 + \cdots + (\tau\lambda_3 a_{1,n} + \tau\lambda_4 a_{2,n})X_n \\ &= \varepsilon((\lambda_1 a_{1,1} + \lambda_2 a_{2,1})X_1 + \cdots + (\lambda_1 a_{1,n} + \lambda_2 a_{2,n})X_n) \\ &\quad + \tau((\lambda_3 a_{1,1} + \lambda_4 a_{2,1})X_1 + \cdots + (\lambda_3 a_{1,n} + \lambda_4 a_{2,n})X_n) \\ &= \varepsilon * F_{\lambda_1,\lambda_2}(X_1,\ldots,X_n) + \tau * F_{\lambda_3,\lambda_4}(X_1,\ldots,X_n), \end{aligned}$$

we thus secure

$$\begin{aligned} &\min(F_{\lambda_5,\lambda_6}) = \varepsilon * \min(F_{\lambda_1,\lambda_2}) + \tau * \min(F_{\lambda_3,\lambda_4}) \leqslant 0 \\ &(\text{since } \min(F_{\lambda_1,\lambda_2}) \leqslant 0,\ \min(F_{\lambda_3,\lambda_4}) \leqslant 0,\ \varepsilon \geqslant 0 \text{ and } \tau \geqslant 0) \text{ and} \\ &\max(F_{\lambda_5,\lambda_6}) = \varepsilon * \max(F_{\lambda_1,\lambda_2}) + \tau * \max(F_{\lambda_3,\lambda_4}) \geqslant 0 \\ &(\text{since } \max(F_{\lambda_1,\lambda_2}) \geqslant 0,\ \max(F_{\lambda_3,\lambda_4}) \geqslant 0,\ \varepsilon \geqslant 0 \text{ and } \tau \geqslant 0). \end{aligned}$$

We hence obtain $\min(F_{\lambda_5,\lambda_6}) \leqslant 0 \leqslant \max(F_{\lambda_5,\lambda_6})$ for any point (λ_5, λ_6) in each cone. Of course, it is also true in the whole R^2 space. Therefore, for any point (λ_1, λ_2) in R^2 space, $F_{\lambda_1,\lambda_2} = 0$ intersects V in R^n space. □

If the constraints of (2.1) plus dependence directions define V, we have a similar lemma. *Banerjee algorithm* for testing loops with unknown limits will use the following definition cited from [1,2,10] to denote the new limits for each pair of relative variables with a given dependence direction.

Definition 3.1a. Given m linear equations (1.1) beneath the constraints of (2.1) and a specific direction vector $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, where d refers to the number of common loops. If $\theta_k \in \{<, >\}$, $1 \leqslant k \leqslant d$, then the bounds of (2.1) for each pair of relative variables will be redefined, assuming $X_{2k-1}\theta_k X_{2k}$ and X_{2k-1} and X_{2k} refer to the same loop indexed variable. The new constraints for X_{2k-1} and X_{2k} are either (3.1) or (3.2).

$$\begin{aligned} &\text{If } \theta_k \in \{<\}, \text{ then} \\ &f_{2k-1}(X_1,\ldots,X_{2k-2}) \leqslant X_{2k-1} \leqslant g_{2k-1}(X_1,\ldots, X_{2k-2}) \\ &\text{and} \\ &f_{2k}(X_1,\ldots,X_{2k-1}) = 1 + X_{2k-1} \leqslant X_{2k} \leqslant g_{2k}(X_1,\ldots,X_{2k-1}). \end{aligned} \tag{3.1}$$

$$\begin{aligned} &\text{If } \theta_k \in \{>\}, \text{ then} \\ &f_{2k-1}(X_1,\ldots,X_{2k-2}) \leqslant X_{2K-1} \leqslant g_{2k-1}(X_1,\ldots,X_{2k-2}) \\ &\text{and} \\ &f_{2k}(X_1,\ldots,X_{2k-1}) \leqslant X_{2K} \leqslant X_{2k-1} - 1 = g_{2k}(X_1,\ldots,X_{2k-1}). \end{aligned} \tag{3.2}$$

Banerjee inequalities for testing loops with symbolic bounds will apply the following definition cited from [1,2,9,10] to define the new limits for each pair of relative variables with a given dependence direction.

Definition 3.1b. Given m linear equations (1.1) beneath the constraints of (2.1) and a specific direction vector $\vec{\theta} = (\theta_1, \ldots, \theta_d)$, where d refers to the number of common loops. If $\theta_k \in \{=, >, <\}, 1 \leqslant k \leqslant d$, then the bounds of (2.1) for each pair of relative variables will be redefined, assuming $X_{2k-1}\theta_k X_{2k}$ and X_{2k-1} and X_{2k} refer to the same loop indexed variable. The new constraints for X_{2k-1} and X_{2k} are shown below, where A and B are equal to $\lambda_1 a_{1,2k-1} + \lambda_2 a_{2,2k-1}$ and $\lambda_1 a_{1,2k} + \lambda_2 a_{2,2k}$, respectively.

Case 1: $\theta_k =$'$=$'

$$f_{2k-1}(X_0, \ldots, X_{2k-2}) = f_{2k}(X_0, \ldots, X_{2k-1}) = \begin{cases} -\infty & \text{if } A + B > 0, \\ +\infty & \text{if } A + B < 0, \\ d & \text{if } A + B = 0, \end{cases}$$

$$g_{2k-1}(X_0, \ldots, X_{2k-2}) = g_{2k}(X_0, \ldots, X_{2k-1}) = \begin{cases} +\infty & \text{if } A + B > 0, \\ -\infty & \text{if } A + B < 0, \\ e & \text{if } A + B = 0, \end{cases}$$

where d and e are arbitrary values in $[-\infty, +\infty]$.

Case 2: $\theta_k =$'>'. If $A > 0$ and $B < 0$, then we have

$$f_{2k-1}(X_0, \ldots, X_{2k-2}) = \begin{cases} -\infty & \text{if } A + B > 0, \\ +\infty & \text{if } A + B < 0, \\ d & \text{if } A + B = 0, \end{cases}$$

$$f_{2k}(X_0, \ldots, X_{2k-1}) = \begin{cases} -\infty & \text{if } A + B > 0, \\ +\infty & \text{if } A + B < 0, \\ e & \text{if } A + B = 0, \end{cases}$$

otherwise, we have

$$f_{2k-1}(X_0, \ldots, X_{2k-2}) = \begin{cases} -\infty & \text{if } A > 0, \\ +\infty & \text{if } A < 0, \\ d & \text{if } A = 0, \end{cases}$$

$$f_{2k}(X_0, \ldots, X_{2k-1}) = \begin{cases} -\infty & \text{if } B > 0, \\ +\infty & \text{if } B < 0, \\ e & \text{if } B = 0, \end{cases}$$

where $-\infty \leqslant e \leqslant +\infty$ and $e + 1 \leqslant d \leqslant +\infty$.

If $A < 0$ and $B > 0$, then we have

$$g_{2k-1}(X_0, \ldots, X_{2k-2}) = \begin{cases} +\infty & \text{if } A + B > 0, \\ -\infty & \text{if } A + B < 0, \\ d & \text{if } A + B = 0, \end{cases}$$

$$g_{2k}(X_0,\ldots,X_{2k-1}) = \begin{cases} +\infty & \text{if } A+B>0, \\ -\infty & \text{if } A+B<0, \\ e & \text{if } A+B=0, \end{cases}$$

otherwise, we have

$$g_{2k-1}(X_0,\ldots,X_{2k-2}) = \begin{cases} +\infty & \text{if } A>0, \\ -\infty & \text{if } A<0, \\ d & \text{if } A=0, \end{cases}$$

$$g_{2k}(X_0,\ldots,X_{2k-1}) = \begin{cases} +\infty & \text{if } B>0, \\ -\infty & \text{if } B<0, \\ e & \text{if } B=0, \end{cases}$$

where $-\infty \leqslant e \leqslant +\infty$ and $e+1 \leqslant d \leqslant +\infty$.

Case 3: $\theta_k =$'<'. If $A<0$ and $B>0$, then we have

$$f_{2k-1}(X_0,\ldots,X_{2k-2}) = \begin{cases} -\infty & \text{if } A+B>0, \\ +\infty & \text{if } A+B<0, \\ d & \text{if } A+B=0, \end{cases}$$

$$f_{2k}(X_0,\ldots,X_{2k-1}) = \begin{cases} -\infty & \text{if } A+B>0, \\ +\infty & \text{if } A+B<0, \\ e & \text{if } A+B=0, \end{cases}$$

otherwise, we have

$$f_{2k-1}(X_0,\ldots,X_{2k-2}) = \begin{cases} -\infty & \text{if } A>0, \\ +\infty & \text{if } A<0, \\ d & \text{if } A=0, \end{cases}$$

$$f_{2k}(X_0,\ldots,X_{2k-1}) = \begin{cases} -\infty & \text{if } B>0, \\ +\infty & \text{if } B<0, \\ e & \text{if } B=0, \end{cases}$$

where $-\infty \leqslant d \leqslant +\infty$ and $d+1 \leqslant e \leqslant +\infty$.

If $A>0$ and $B<0$, then we have

$$g_{2k-1}(X_0,\ldots,X_{2k-2}) = \begin{cases} +\infty & \text{if } A+B>0, \\ -\infty & \text{if } A+B<0, \\ d & \text{if } A+B=0, \end{cases}$$

$$g_{2k}(X_0,\ldots,X_{2k-1}) = \begin{cases} +\infty & \text{if } A+B>0, \\ -\infty & \text{if } A+B<0, \\ e & \text{if } A+B=0, \end{cases}$$

otherwise, we have

$$g_{2k-1}(X_0,\ldots,X_{2k-2}) = \begin{cases} +\infty & \text{if } A > 0, \\ -\infty & \text{if } A < 0, \\ d & \text{if } A = 0, \end{cases}$$

$$g_{2k}(X_0,\ldots,X_{2k-1}) = \begin{cases} +\infty & \text{if } B > 0, \\ -\infty & \text{if } B < 0, \\ e & \text{if } B = 0, \end{cases}$$

where $-\infty \leqslant d \leqslant +\infty$ and $d+1 \leqslant e \leqslant +\infty$.

Then we should discuss how to decide the minimum and maximum limits for each pair of relative variables when their corresponding dependence directions are given. Each dependence direction is known to relate a unique pair of loop indices, which are associated with one of the common loops. Obviously, we should choose the new constraints redefined as stated in Definitions 3.1a and 3.1b for each pair of relative variables and the original bounds (2.1) for other variables not to be constrained by dependence directions such that F_{λ_1,λ_2} has the minimum value and the maximum value.

Let $\Phi_{(2k-1,2k)}$ be the sum of the coefficients of X_{2k-1} and X_{2k} in F_{λ_1,λ_2}, where X_{2k-1} and X_{2k} are related by a dependence direction, i.e., $\Phi_{(2k-1,2k)} = \lambda_1(a_{1,2k-1} + a_{1,2k}) + \lambda_2(a_{2,2k-1} + a_{2,2k})$ [9]. The minimum point and maximum point of F_{λ_1,λ_2} in V, in the presence of dependence directions, depend not only on the sign of the coefficient of each variable but also on the sign of $\Phi_{(2k-1,2k)}$, as clearly undertaken from statements above. From [9], the equation $\Phi_{(2k-1,2k)} = 0$ is called a Φ equation. Each Φ equation corresponds to a Φ line in R^2. There are at most $n/2$ Φ lines. All Φ lines and Ψ lines divide R^2 space into at most $3n$ regions. Each region is still a cone.

Lemma 3.2. *Suppose that an unbounded convex set V is denoted by the limits of* (2.1) *as well as dependence directions. If $F_{\lambda_1,\lambda_2} = 0$ intersects V for every (λ_1,λ_2) in every Φ line and every Ψ line, then $F_{\lambda_1,\lambda_2} = 0$ also intersects V for every (λ_1,λ_2) in R^2.*

Proof. Follow the same arguments as that for Lemma 3.1. □

As a matter of fact, it suffices to test a single point in each Φ line or each Ψ line for determining whether F_{λ_1,λ_2} intersects V for every (λ_1,λ_2) in those lines.

Lemma 3.3. *Suppose that an unbounded convex set V is denoted by the limit of* (2.1) *and dependence directions. Given a line in R^2 corresponding to an equation $a\lambda_1 + b\lambda_2 = 0$, if $F_{\lambda_1,\lambda_2} = 0$ intersects V in R^n for any fixed point $(\lambda_1^0,\lambda_2^0) \neq (0,0)$ in the line, then for every (λ_1,λ_2) in the line, $F_{\lambda_1,\lambda_2} = 0$ also intersects V.*

Proof. Note that $F_{\lambda_1,\lambda_2} = 0$ intersects V if and only if $\min(F_{\lambda_1,\lambda_2}) \leqslant 0 \leqslant \max(F_{\lambda_1,\lambda_2})$ on V. Every point in the line can be expressed as $(\varepsilon\lambda_1^0, \varepsilon\lambda_2^0)$, where $-\infty < \varepsilon < +\infty$. If $\varepsilon \geqslant 0$, then $\min(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0}) = \varepsilon * \min(F_{\lambda_1^0,\lambda_2^0})$ and $\max(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0}) = \varepsilon * \max(F_{\lambda_1^0,\lambda_2^0})$. It is thus concluded that $\min(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0}) \leqslant 0 \leqslant \max(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0})$. Similarly, if $\varepsilon < 0$, then $\min(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0}) = \varepsilon * \max(F_{\lambda_1^0,\lambda_2^0})$ and $\max(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0}) = \varepsilon * \min(F_{\lambda_1^0,\lambda_2^0})$. It is immediately derived that $\min(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0}) \leqslant 0 \leqslant \max(F_{\varepsilon\lambda_1^0,\varepsilon\lambda_2^0})$. Therefore, for any (λ_1, λ_2) in the line, F_{λ_1,λ_2} intersects V. $\square$

Definition 3.2. Given an equation of the form $a\lambda_1 + b\lambda_2 = 0$ where a, b are not 0 simultaneously, a canonical solution of the equation is defined as follows:

$$(\lambda_1, \lambda_2) = (1, 0) \quad \text{if } a = 0,$$

$$(\lambda_1, \lambda_2) = (0, 1) \quad \text{if } b = 0,$$

$$(\lambda_1, \lambda_2) = (1, 1) \quad \text{if both of } a, b \text{ are } 0,$$

$$(\lambda_1, \lambda_2) = (b, -a) \quad \text{if neither of } a, b \text{ is } 0.$$

Definition 3.3. The Λ set is denoted to be the set of all canonical solutions to Φ equations and Ψ equations. The hyperplane in R^n corresponding to $\lambda_1 F_1 + \lambda_2 F_2 = 0$, where (λ_1, λ_2) is a canonical solution in the Λ set, is called a λ plane.

There are at most n Ψ equations if V is denoted by the bounds of (2.1) only. There are at most n Ψ and $n/2$ Φ equations if V is defined by the limits of (2.1) and dependence directions. Each of the equations generates a canonical solution according to Definition 3.2. Each canonical solution forms a λ plane in light of Definition 3.3. Obviously, λ planes tested are at most n if V is defined by the constraints of (2.1) only, and are at most $3n/2$ if V is denoted by the bounds of (2.1) as well as dependence directions. If V is denoted by constant loop constraints only, then there are no more than n hyperplanes in the set [9]. If constant loop bounds as well as dependence directions define V, then there are no more than $3n/2$ hyperplanes in the set [9]. It is at once concluded that the number of λ planes tested by the infinity Lambda test is the same as that of λ planes checked by the Lambda test.

The Banerjee infinity test is first applied to test each hyperplane inferred from every dimension of a two-dimensional coupled array. If every hyperplane intersects V, then the infinity Lambda test is employed to simultaneously check every hyperplane. The infinity Lambda test examines the subscripts from two coupled dimensions, and then figures out the Λ set from Ψ equations and Φ equations. Each element in the Λ set determines a λ plane. Each λ plane is tested to see if it intersects V, by checking its minimum and maximum values as done in the Banerjee infinity test for testing each *single* dimension with unknown limits. If any λ plane does not intersect V, then there is no data dependence. If every λ plane intersects V, then data dependence should be assumed, unless further tests on integer solutions are to be performed. For efficiency, computation of the Λ set and the test on λ planes are performed alternately, i.e., a new element in the Λ set is computed only after it has been tested that the previous λ plane intersects V. Obviously, repeated canonical solutions can be ignored.

We use the following example to explain the enhanced power of the infinity Lambda test over the Lambda test, when it is used to test data dependence for unknown bounds and given dependence directions.

Consider the loop with induction variable in Fig. 2.

If we want to determine whether there exists output data dependence of the array A with direction vector $(<,<)$, then the nonlinear expressions of the array A with direction vector $(<,<)$ can be transformed into the following nonlinear equations:

$$N \times X_1 - N \times X_2 + X_3 - X_4 = 0 \quad (\text{ex1}),$$
$$N \times X_1 - N \times X_2 + X_3 - X_4 = 0 \quad (\text{ex2})$$

subject to the symbolic bounds $1 \leqslant X_1$ and $X_2 \leqslant M$, and $1 \leqslant X_3$ and $X_4 \leqslant N$, where M and N are unknown limits, and the limits of a direction vector $X_1 < X_2$ and $X_3 < X_4$.

The Φ equations are generated if the infinity Lambda test based on computational principles of checking symbolic limits of *Banerjee's inequalities* is applied to resolve the problem. The Φ equations are $0 \times \lambda_1 + 0 \times \lambda_2 = 0$ and $0 \times \lambda_1 + 0 \times \lambda_2 = 0$. The Λ set from the Φ equations is easily determined: $\Lambda = \{(1,1)\}$. The canonical solution $(1,1)$ in the Λ set gives the λ plane: $2 \times N \times (X_1 - X_2) + 2 \times (X_3 - X_4) = 0$. Let $h_4(X_1, \ldots, X_4) = 2 \times N \times (X_1 - X_2) + 2 \times (X_3 - X_4)$, then according to Case 3 of Definition 3.1b, we have

$$h_4(x_1, \ldots, x_4) \leqslant 2 \times N \times (1 \times (d) + (-1) \times (d+1)) + 2 \times (X_3 - X_4)$$

(because the coefficient of $X_1 > 0$, the coefficient of $X_2 < 0$, and the sum of the coefficients for X_1 and X_2 is equal to 0), and

$$h_4(x_1, \ldots, x_4) \leqslant -2 \times N + 2 \times (1 \times d + (-1) \times (d+1)) = -2 - 2 \times N$$

(because the coefficient of $X_3 > 0$, the coefficient of $X_4 < 0$, and the sum of the coefficients for X_3 and X_4 is equal to 0).

The maximum bound for the λ plane is immediately inferred to be $-2 - 2 \times N$. Similarly, the minimum limit to the λ plane is at once concluded to be $-\infty$. Since the maximum value for the λ plane is less than 0, the infinity Lambda test based on computational principles of checking symbolic limits of *Banerjee's inequalities* in light of Lemmas 3.1–3.3 infers that there is no real-valued solution.

The same Φ equations are also generated if the infinity Lambda test based on computational principles of checking symbolic limits of *Banerjee's algorithm* is applied to resolve the same problem. The Φ equations are $0 \times \lambda_1 + 0 \times \lambda_2 = 0$ and $0 \times \lambda_1 + 0 \times \lambda_2 = 0$. The Λ set from the Φ equations is easily determined: $\Lambda = \{(1,1)\}$. The canonical solution $(1,1)$ in the Λ set gives the λ plane:

```
K = 0
DO J = 1, M              DO J = 1, M
  DO I =1, N               DO I = 1, N
     K = K + 1
     A(K, K) = ...            A(I+N*(J-1), I+N*(J-1)) = ...
  ENDDO                    ENDDO
ENDDO                    ENDDO
```

Fig. 2. Before and after induction variable substitution.

$2 \times N \times (X_1 - X_2) + 2 \times (X_3 - X_4) = 0$. In the light of the second step in the proof of Lemma 3.1, let $h_4(X_1, \dots, X_4) = 2 \times N \times (X_1 - X_2) + 2 \times (X_3 - X_4)$. Then we have

$$h_4(x_1, \dots, x_4) \leqslant h_3(x_1, \dots, x_3),$$

where

$$\begin{aligned} h_3(x_1, \dots, x_3) &= 2 \times N \times (x_1 - x_2) \\ &\quad + 2 \times (x_3 - (1 + x_3)) \quad (\text{since the coefficient of } x_4 < 0), \\ &\;\vdots \\ &\leqslant h_1(x_1) = 2 \times N \times (x_1 - (1 + x_1)) - 2 = -2 - 2 \times N \\ &\qquad\qquad (\text{since the coefficient of } x_2 < 0). \end{aligned}$$

The maximum value for the λ plane is immediately inferred to be $-2 - 2 \times N$. Similarly, the minimum value to the λ plane can also be derived to be $-\infty$. Since the maximum value for the λ plane is less than 0, the infinity Lambda test in light of Lemmas 3.1–3.3 infers that there is no real-valued solution.

3.2. The case of multi-dimensional array references

We take account of m linear equations in (1.1) with $m > 2$ for generalizing the infinity Lambda test. All m linear equations are assumed to be connected; otherwise they can be partitioned into smaller systems. As stated before, we can hypothesize that there are no redundant equations. By Dongar et al. [9], an arbitrary linear combination of m linear equations can be written as $\sum_{i=1}^{m} \lambda_i F_i = 0$, where $F_i = \sum_{j=1}^{n} a_{i,j} X_{i,j}$. Let $F_{\lambda_1,\dots,\lambda_m} = \sum_{i=1}^{m} \lambda_i F_i$, and then

$$F_{\lambda_1,\dots,\lambda_m} = \left(\sum_{j=1}^{m} \lambda_j a_{j,1}\right) X_1 + \cdots + \left(\sum_{j=1}^{m} \lambda_j a_{j,n}\right) X_n.$$

It is to be determined whether $F_{\lambda_1,\dots,\lambda_m} = 0$ intersects V in R^n space for arbitrary $(\lambda_1, \dots, \lambda_m)$. The coefficient of each variable in $F_{\lambda_1,\dots,\lambda_m}$ is a linear function of $(\lambda_1, \dots, \lambda_m)$ in R^m, which is $\Psi^{(i)} = \sum_{j=1}^{m} \lambda_j a_{j,i}$ for $1 \leqslant i \leqslant n$. The equation $\Psi^{(i)} = 0, 1 \leqslant i \leqslant n$, is called a Ψ equation. A Ψ equation corresponds to a hyperplane in R^m, called a Ψ plane. Each Ψ plane divides the whole space into two closed halfspaces $\Omega_i^+ = \{(\lambda_1, \dots, \lambda_m) | \Psi^{(i)} \geqslant 0\}$ and $\Omega_i^- = \{(\lambda_1, \dots, \lambda_m) | \Psi^{(i)} \leqslant 0\}$ Let $\Phi_{(2k-1,2k)}$ be the sum of the coefficients of X_{2k-1} and X_{2k} in $F_{\lambda_1,\dots,\lambda_m}$, where X_{2k-1} and X_{2k} are related by a dependence direction, i.e., $\Phi_{(2k-1,2k)} = \sum_{i=1}^{m} \lambda_i (a_{i,2k-1} + a_{i,2k})$. The equation $\Phi_{(2k-1,2k)} = 0$, is called a Φ equation. A Φ equation corresponds to a hyperplane in R^m, which is called a Φ plane. Each Φ plane separates the whole space into two closed halfspaces $\delta^+_{(2k-1,2k)} = \{(\lambda_1, \dots, \lambda_m) | \Phi_{(2k-1,2k)} \geqslant 0\}$ and $\delta^-_{(2k-1,2k)} = \{(\lambda_1, \dots, \lambda_m) | \Phi_{(2k-1,2k)} \leqslant 0\}$. If V is defined by the constraints of (2.1) only, then a nonempty set $\bigcap_{i=1}^{n} \Omega_i$, where $\Omega_i \in \{\Omega_i^+, \Omega_i^-\}$, is called a λ region. If the bounds of (2.1) as well as dependence directions denote V, then a nonempty set $(\bigcap_{i=1}^{n} \Omega_i) \cap (\bigcap_{k=1}^{d} \delta_{(2k-1,2k)})$, where $\Omega_i \in \{\Omega_i^+, \Omega_i^-\}$ and $\delta_{(2k-1,2k)} \in \{\delta^+_{(2k-1,2k)}, \delta^-_{(2k-1,2k)}\}$, is called a λ region. The intersection of $\delta_{(2k-1,2k)}$ is taken for all pairs of index variables, which are related by a dependence direction. Every λ region is a cone in R^m space. The λ regions in R^m space have several lines as the frame of their boundaries. Each line (called a λ line) is the intersection of some Ψ and Φ planes.

In the following, Lemmas 3.4 and 3.5 are an extension of Lemmas 5 and 6 in [9], respectively.

Lemma 3.4. *If $F_{\lambda_1,\ldots,\lambda_m} = 0$ intersects V for every $(\lambda_1, \ldots, \lambda_m)$ in every λ line, then $F_{\lambda_1,\ldots,\lambda_m} = 0$ also intersects V for every $(\lambda_1, \ldots, \lambda_m)$ in R^m space.*

Proof. Similar to Lemma 3.1. □

Lemma 3.5. *Given a line in R^m space which crosses the origin of the coordinates, if $F_{\lambda_1,\ldots,\lambda_m} = 0$ intersects V in R^n space for any fixed point $(\lambda_1^0, \ldots, \lambda_m^0) \neq (0, \ldots, 0)$ in the line, then for every $(\lambda_1, \ldots, \lambda_m)$ in the line, $F_{\lambda_1,\ldots,\lambda_m} = 0$ also intersects V.*

Proof. Similar to Lemma 3.3. □

There is a finite set of hyperplanes in R^m space such that S intersects V if and only if every hyperplane in the set intersects V. If V is denoted by constant loop constraints only, then there are no more than $\begin{pmatrix} n \\ m-1 \end{pmatrix}$ hyperplanes in the set [9]. If V is defined by the limits of (2.1) only, then the number of hyperplanes in the set is also at most $\begin{pmatrix} n \\ m-1 \end{pmatrix}$. If V is denoted by constant loop constraints as well as dependence directions, then there are no more than $\begin{pmatrix} 3n/2 \\ m-1 \end{pmatrix}$ hyperplanes in the set [9]. If V is defined by the limits of (2.1) and dependence directions, then the number of hyperplanes is also at most $\begin{pmatrix} 3n/2 \\ m-1 \end{pmatrix}$ in the set. It is right away derived that the number of λ planes tested in the infinity Lambda test is the same as that of λ planes checked in the Lambda test. The detail of the infinity Lambda test in the general case is not considered since the discussion is similar to the case of $m = 2$.

3.3. Time complexity

The common phases for the Lambda test and the infinity Lambda test include: (1) calculating λ values and (2) examining each λ plane. λ values are easily determined according to Φ equations, Ψ equations and Definition 3.2. It is clear that the time complexity to computing a λ value is $O(y)$ from Definition 3.2, where y is a constant. Each λ value corresponds to a λ plane. The time complexity of determining a λ plane from one λ value is $O(n)$, where n is the number of variables in coupled references. Each λ plane is tested to see if it intersects V, by checking its minimum and maximum values. The extreme values can be calculated from the *Banerjee inequalities* and also computed from the second step in the proof of Lemma 3.1. The second step in the proof of Lemma 3.1 is exactly equivalent to the Banerjee infinity test (the *Banerjee algorithm* and the *Banerjee inequalities* for testing single dimension with symbolic bounds). The time complexity of the *Banerjee inequalities* and the Banerjee infinity test is both $O(z)$, where z is the number of variables in the λ plane [2,10]. Hence, the time complexity of the Lambda test and the infinity Lambda test for examining a λ plane is at once derived to be $O(z+n+y)$. The number of λ planes checked in the Lambda test and in the infinity Lambda test is the same, and is at most $\binom{3n/2}{m-1}$, where m is the number of coupled dimensions and n is the number of variables in coupled references, in light of statements in Section 3.3 [9]. Therefore, the worst-case time complexity for the Lambda test and the infinity Lambda test is immediately inferred to be

$$O\left(\begin{bmatrix} 3n/2 \\ m-1 \end{bmatrix} * (z+n+y)\right).$$

Two-dimensional arrays with coupled subscripts appear quite frequently in real programs, as clearly indicated from statements in Section 1. The number of λ planes examined to each two-dimensional array tested is at most $3n/2$ according to statements in Section 3.2 [9]. If the Lambda test and the infinity Lambda test are applied to deal with the array, then their worst-case time complexity is $O((3n/2)*(z+n+y))$. The number of λ planes checked is almost 1 due to the regularity of coefficients in coupled subscripts in real two-dimensional arrays tested. Hence, the worst-case time complexity of the Lambda test and the infinity Lambda test for testing those real two-dimensional arrays is nearly equal to $O(z+n+y)$. The infinity Lambda test preserves the efficiency of the Lambda test.

4. Experimental results

We tested the infinity Lambda test and performed experiments on Personal Computer Intel 80486 through the codes cited from five numerical packages EISPACK, LINPACK, Parallel loops, Livermore loops and Vector loops [3,13,19]. The codes include more than 37 000 lines of statements, and 17 433 pairs of array references consisting of the same pair of array references with different direction vectors and unknown limits were found to have coupled subscripts. The infinity Lambda test checked that there were no data dependences for 2092 pairs of coupled arrays beneath symbolic limits as well as any given direction vectors.

The infinity Banerjee test is only used to check those arrays with one-dimensional arrays. Also, the Lambda test is only applied towards determining those arrays with coupled subscripts under constant bounds. The infinity Lambda test in our experiments is only applied to test those arrays with coupled subscripts under symbolic bounds. The infinity Lambda test found 2092 cases that had no data dependence. The improvement rate can be affected by two factors. First, the frequency of coupled subscripts. Second, the "success rate" of the infinity Lambda test, by which we mean how often an infinity Lambda test detects a case where there is no data dependence. Let b be the number of the coupled subscripts found in our experiments, and let c be the number that is detected to have no data dependence from the coupled subscripts. Thus the success rate is denoted to be equal to c/b. In our experiments, 17 433 pairs of array references were found to have coupled subscripts, and 2092 of them were found to have no data dependence. So the success rate in our experiments was about equal to 12%. The infinity Lambda test increases the success rate of the Lambda test. The increasing success rate was about 12%.

In our experiments, it is found that there are different frequencies of coupled subscripts in different benchmark codes. Success rate for each benchmark is shown in Table 1 in which each row shows how many cases for each benchmark were checked to have no data dependence. For instance, the first row reveals that 7722 pairs of array references from EISPACK were found to have coupled subscripts, and 540 of them were found to have no data dependence. Therefore, the success rate of the infinity Lambda test for EISPACK is 7%. For all of benchmark codes in our experiments, the success rate to each benchmark was from 7% to 17%. This indicates that for multi-dimensional arrays with coupled subscripts the success rate of the infinity Lambda test varies with the benchmark tested.

In our experiments, the Power test was also employed to resolve 17 433 pairs of arrays with coupled references and unknown limits. The Power test was found to detect 1932 cases with no data dependences. Table 2 shows the accuracy of the infinity Lambda test over the Power test for those 17 433 pairs of array with results. It is very clear from Table 2 that the infinity Lambda test is slightly superior to the Power test in terms of analyzing accuracy.

Suppose that k_g and k_p are the execution times to treat data dependence problem of a coupled-subscript array for the infinity Lambda test and Power test, subsequently. The speed-up in Table 3 is defined to be the set of k_p/k_g. Each row in Table 3 shows how many times the execution time of the Power test took longer than the execution time of the infinity Lambda test. For example, the first row shows that there were 63 subroutines in which the execution time of the

Table 1
The success rate of the infinity Lambda test for array references in each benchmark

Benchmark	Pairs of arrays checked	Pairs of examined arrays without data dependence	Success rate (%)
EISPACK	7722	540	7
LINPACK	1458	160	11
Parallel loops	7867	1336	17
Livermore loops	215	32	15
Vector loops	171	24	14

Table 2
The accuracy of the infinity Lambda test when compared with the Power test

Data dependence testing methods	Proved independent
Infinity Lambda test	2092
Power test	1932

Table 3
The speed-up of the infinity Lambda test when compared with the Power test

Speed-up	Total number of subroutines
10.1–20.3	63
6.2–9.5	10

Power test took from 10.1 to 20.3 times longer than that of the infinity Lambda test. For all of the subroutines in our experiments, the execution time of the Power was indicated to take from 6.2 to 20.3 times longer than the execution time of the infinity Lambda test. This indicates that for multi-dimensional arrays with coupled subscripts the efficiency of the infinity Lambda test is much better than that of the Power test. This is because the Power test cannot deal with the loops with induction variable.

5. Conclusions

The infinity Lambda test enhances data dependence analysis significantly when there are coupled subscripts in multi-dimensional array references with *unknown* limits. The infinity Lambda test only ascertains whether real-valued solutions exist because, like the Lambda test or the generalized Lambda test, it is based on equality consistency checking. The infinity Lambda test is exactly equivalent to a multi-dimensional version of the Banerjee infinity test (the *Banerjee algorithm* and the *Banerjee inequalities* for testing *single* dimension with *unknown* bounds) because it can determine simultaneous constrained real-valued solutions. Petersen [10] pointed out that Banerjee's infinity test breaks dependences in more than 9% of the cases when it is applied before Banerjee's inequalities and algorithm, and the application of Banerjee's inequalities and algorithm contributes nothing extra. Li et al. [9] found that the Lambda test for coupled array references under constant bounds usually increases the cost of the *Banerjee inequalities* by a factor of 2 or less. It is found that the generalized Lambda test for coupled array references under variable bounds usually increases the cost of the *Banerjee algorithm* by a factor of 2 or less [5]. The infinity Lambda test for coupled array references beneath symbolic limits usually also increases the cost of the *Banerjee infinity test* by a factor of 2 or less, as shown from our time complexity.

The Power test is a combination of Fourier–Motzkin variable elimination with an extension of Euclid's GCD algorithm [17,18]. The Omega test combines new methods for eliminating equality constraints with an extension of Fourier–Motzkin variable elimination [11]. The two tests currently have the highest precision and the widest applicable range for checking linear arrays in the field of data dependence testing. However, the cost of the Power and Omega tests is very expensive because the worst-case of Fourier–Motzkin variable elimination is exponential in the number of free variables [11,17,18]. It is pointed out [15,17,18] that using Fourier–Motzkin variable elimination for dependence testing takes from 22 to 28 times longer than the

Banerjee inequalities. In [17,18] it is also indicated that the Lambda test is a very precise and efficient method for testing two-dimensional coupled arrays with constant bounds. The Range test currently has the highest precision and the widest applicable range for checking nonlinear arrays in the field of data dependence testing [4].

The generalized Lambda test, a generalized version of the Lambda test proposed in [5], is an efficient and precise method to ascertain whether there exists data dependence for coupled arrays with constant or variable loop bounds. The infinity Lambda test extends the applicable range of the Lambda test and the generalized Lambda test to deal with arrays references with unknown bounds. Therefore, integrating the Lambda test, the generalized Lambda test and the infinity Lambda test seems to be a practical scheme to analyze data dependence for coupled-subscript array references.

References

[1] U. Banerjee, Dependence Analysis for Supercomputing, Kluwer Academic Publishers, Norwell, MA, 1988.

[2] U. Banerjee, Dependence Analysis, Kluwer Academic Publishers, Norwell, MA, 1997.

[3] D. Callahan, J. Dongarra, D. Levine, Test suite for vectorizing compilers, Version 3.0, Argonne National Laboratory, 1991.

[4] W. Blume, R. Eigenmann, Nonlinear and symbolic data dependence testing, IEEE Transaction on Parallel and Distributed Systems 9 (12) (1998) 1180–1194.

[5] W.-L. Chang, C.-P. Chu, J. Wu, The generalized Lambda test, in: Proceedings of the First Merged Symposium on IPPS/SPDP, Orlando, FL, March 1998, pp. 181–186.

[6] M. Haghighat, C. Polychronopoulos, Symbolic dependence analysis for high-performance parallelizing compilers, in: Parallel and Distributed Computing: Advances in Languages and Compilers for Parallel Processing, MIT Press, Cambrige, MA, 1991, pp. 310–330.

[7] X. Kong, D. Klappholz, K. Psarris, The i test, IEEE Transaction on Parallel and Distributed Systems 2 (3) (1991) 342–359.

[8] X. Kong, D. Klappholz, K. Psarris, The direction vector i test, IEEE Transaction on Parallel and Distributed Systems 4 (11) (1993) 1280–1290.

[9] Z. Li, P.-C. Yew, C.-Q. Zhu, An efficient data dependence analysis for parallelizing compilers, IEEE Transaction on Parallel and Distributed Systems 1 (1) (1990) 26–34.

[10] P.M. Petersen, Evaluation of programs and parallelizing compilers using dynamic Analysis techniques, Ph.D. Thesis, University of Illinois at Urbana–Champaign, January 1993.

[11] W. Pugh, A practical algorithm for exact array dependence analysis, Communication of the ACM 35 (8) (1992) 102–114.

[12] W. Rudin, Principles of Mathematical Analysis, International Series in Pure and Applied Mathematics, McGraw-Hill, New York, 1964.

[13] B.T. Smith, et al., Matrix Eigensystem Routines-Eispack Guide, Springer, Heidelberg, 1976.

[14] Z. Shen, Z. Li, P.-C. Yew, An empirical study of Fortran programs for parallelizing compilers, IEEE Transaction on Parallel and Distributed Systems 1 (3) (1990) 356–364.

[15] R. Triolet, F. Irigoin, P. Feautrier, Direct parallelization of call statements, in: Proceedings of the SIGPLAN Symposium on Compiler Construction, Palo Alto, CA, June 1986, pp. 176–185.

[16] W.J. Vaughan, A residuals management model of the iron and steel industry: a linear programming approach, University of Microfilms International, Ann Arbor, MI, 1986.

[17] M. Wolfe, C.-W. Tseng, The power test for data dependence, IEEE Transaction on Parallel and Distributed Systems 3 (5) (1992) 591–601.

[18] M. Wolfe, High Performance Compilers for Parallel Computing, Addison-Wesley, Redwood City, CA, 1996.

[19] J. Dongar, M. Furtney, S. Reinhardt, Parallel loops – a test Suite for parallel compilers: description and example results, Parallel Computing 17 (1991) 1247–1255.