FILOMAT (Niš) 9:2 (1995), 333–343
Filomat '94, Niš, October 22-24, 1994
Geometry. Computer Sciences

DEPENDENCE TESTING ON LOOPS WITH BOUNDS WHICH ARE FUNCTIONS OF OUTER LOOP INDICES

Suzana Stojković

ABSTRACT. *Parallelizing compilers are compilers which translate sequential programs into parallel ones. Program loops are the most frequent sources of parallelism in sequential programs. Because of that, parallelizing compilers first must detect loops which can be run in parallel. Different iterations of the same loop can execute in parallel if they process different data. Parallel loops can be identified by detecting data dependencies across the loop body. For data dependence testing a few algorithms were developed. In this paper GCD test and Banerjee's test are presented. These algorithms are applicable when bounds of loop indices are constant. This paper shows how Banerjee's test can be exploited when the inner loop bounds are functions of outer loops indices. We, first, must compute minimums of the lower and maximums of the upper loop bounds. We solved this problem when the loop bounds are linear functions. We show that this minimums and maximums are dependent on the data dependence direction vector. We have also modified Banerjee's test, slightly.*

1. Introduction

Developments in semiconductor technology tend to reduce dimension and price of electronic components, but to grow their speed. Hardware performances become better every day. Now, supercomputers are developed.

Fast hardware development lead to a software crisis. A new problem appears: how to exploit all hardware performances. Because of that, parallel algorithms have been developed, in the last few years. Programmer who designs parallel algorithms must be familiar with hardware

architecture for which these algorithms are meant. This leads to the idea that the parallelization can be done by compilers. Now, parallelizing compilers are very popular area of computer science.

The major problem of parallelizing compilers is to detect parallelism during sequential programs. Program loops are the most frequent source of parallelism. Because of that, first problem is to detect loops which can be

run in parallel. Different iterations of the same loop can execute in parallel if they process different data. The key to identification of parallel loops is to detect data dependencies across loop body.

There are three types of data dependencies exist:

(1) *Data true dependence* - exists when a variable computed in statement S_1 is used in some next statement S_2. We say that S_2 is data true dependent on S_1, and write this as $S_1\delta^t S_2$.

(2) *Data anti dependence* - exists when a variable is used in statement S_1 and it is defined in some next statement S_2. We say that S_2 is data anti dependent on S_1, and write this as $S_1\delta^a S_2$.

(3) *Data output dependence* - exists if the same variable is defined by statements S_1 and S_2. We say that S_2 is data output dependent on S_1, and write this as $S_1\delta^o S_2$.

Detection of dependence is not difficult if only scalar variables figure in the loop. Difficulties are caused by subscripted variables. For example, we will try to determine all dependencies which exist in the next loop:

$$\begin{aligned} L: &\quad DO\ 10\ \mathrm{I} = 5, 10 \\ S_1: &\quad \mathrm{A}(I+3) = 2 * A(I-4) \\ S_2: &\quad \mathrm{B}(I) = A(I) + C10 \\ &\quad 10\ CONTINUE \end{aligned}$$

First dependence which can be identified is the dependence between statements S_1 and S_2. Elements of array A defined in statement S_1 are used in statement S_2. We can say that S_2 is true dependent on S_1 ($S_1\delta^t S_2$).

On the basis of the above, we can say that $S_1\delta^t S_1$. However, if we look a bounds of the loop index I, we will see that the statement S_1 defines elements A(8),...,A(13), but uses elements A(1),...,A(6), and dependence $S_1\delta^t S_1$ does not exist.

2. Data dependence testing algorithms

Below example shows that data dependence testing algorithms must analyze several more different elements, like: dependence among the statements in the loop; dependence in the appropriate region, etc. These algorithms find data dependence direction vectors [2,4], too.

Data dependence direction vector, θ, defines relations between values of loop indices for which dependence exists. The dimension of vector θ (m) is the number of loops which enclose the statements S_1 and S_2. The elements of vector are members of the set $\{<, =, >, *\}$. We will assume that certain loop include two statement S_1 and S_2. Lets us label the I-th iteration of

statement S_1 as $S_1(I)$. Let also $S_1(I)\delta_\theta S_2(J)$. The appropriate element of vector θ is:

- $<$, if $I < J$;
- $>$, if $I > J$;
- $=$, if $I = J$;
- *, if relation between I an J is unknown.

For the loop L: m=1 and $S_1\delta_\theta S_2$ where is $\theta = (<)$, because the element of array A which was defined in the first iteration of statement S_1, will be used in forth iteration of statement S_2 $(1 < 4)$.

Let us consider two statements (S_1 and S_2) which are enclosed by m loops:

$$
\begin{array}{c}
DO\ 10\ \mathrm{I}_1 = T_1, U_1 \\
DO\ 10\ \mathrm{I}_2 = T_2, U_2 \\
\vdots \\
DO\ 10\ \mathrm{I}_m = T_m, U_m \\
\vdots \\
\ldots\ \mathrm{A}(I)\ \ldots \\
\vdots \\
\ldots\ \mathrm{A}(J)\ \ldots \\
\vdots \\
10\ CONTINUE
\end{array}
$$

Indices I and J are functions defined as:

$$I = f_1(I_1, I_2, \ldots, I_m) = f_1(I) \tag{1}$$

$$J = f_2(I_1, I_2, \ldots, I_m) = f_2(J) \tag{2}$$

The dependence between S_1 and S_2 exists in those iterations in which I equals J. The goal of these algorithms is to determine whether the equation

$$f_1(I) = f_2(J) \tag{3}$$

has got integer solutions. This equation is a dependence equation.

Functions f_1 and f_2, in most cases, are linear:

$$f_1 = \sum_{k=1}^{m} a_k I_k + a_0 \tag{4}$$

$$f_2 = \sum_{k=1}^{m} b_k J_k + b_0 \tag{5}$$

In this case the dependence equation is a diophantine equation and can be stated as:

$$\sum_{k=1}^{m} a_k I_k - \sum_{k=1}^{m} b_k J_k = b_0 - a_0 \tag{6}$$

Whether the diophantine equation has a solution can be detected by GCD test. The dependence equation can be written as follows:

$$\sum_{k=1,\theta_k='='}^{m} (a_k - b_k) I_k + \sum_{k=1,\theta_k\neq'='}^{m} a_k I_k - \sum_{k=1,\Theta_k\neq'='}^{m} b_k J_k = b_0 - a_0 \tag{7}$$

Let $g = GCD(\{a_k - b_k, \theta_k ='='\}, \{a_k, \theta_k \neq'='\}, \{b_k, \theta_k \neq'='\})$.

GCD test: The dependence equation has solutions **iff** $g \mid (b_0 - a_0)$ or $a_0 = b_0$.

Note that this test does not answer the question whether the solutions exist in the given region. This question can be answered second by a group of data dependence testing algorithms - *Banerjee's test.*

This test needs to introduce a positive part of real number r (r^+), and the negative part of r (r^-), as:

$$r^+ = \begin{cases} 1 & r > 0 \\ 0 & r \leq 0 \end{cases} \tag{8}$$

$$r^- = \begin{cases} -1 & r < 0 \\ 0 & r \geq 0 \end{cases} \tag{9}$$

Banerjee's criteria: The data dependence for a given vector θ does exist if the GCD test is satisfied and the next inequality is satisfied, too:

$$\sum_{k=1}^{m} LC_k \leq b_0 - a_0 \leq \sum_{k=1}^{m} UC_k \tag{10}$$

where:

(11)

$$LC_k = \begin{cases} -(a_k^- + b_k)^+(U_k - T_k - 1) + (a_k - b_k)T_k - b_k & \text{for } \theta_k ='<' \\ -(a_k - b_k)^-(U_k - T_k) + (a_k - b_k)T_k & \text{for } \theta_k ='=' \\ -(b_k^+ - a_k)^+(U_k - T_k - 1) + (a_k - b_k)T_k + a_k & \text{for } \theta_k ='>' \\ -(a_k^- + b_k^+)(U_k - T_k) + (a_k - b_k)T_k & \text{for } \theta_k =' *' \end{cases}$$

(12)

$$UC_k = \begin{cases} (a_k^+ - b_k)^+(U_k - T_k - 1) + (a_k - b_k)T_k - b_k & \text{for } \theta_k = '<' \\ (a_k - b_k)^+(U_k - T_k) + (a_k - b_k)T_k & \text{for } \theta_k = '=' \\ (b_k^- - a_k)^+(U_k - T_k - 1) + (a_k - b_k)T_k + a_k & \text{for } \theta_k = '>' \\ (a_k^+ + b_k^-)(U_k - T_k) + (a_k - b_k)T_k & \text{for } \theta_k = ' *' \end{cases}$$

Here we present a generalized algorithm for determining dependence relations for the given loop. Data dependence testing is done hierarchically. First, we begin with the assumption that data dependence direction vector is unknown ($\theta_k = ' *'$,k[1:m]), and determine if the dependence exists for any vector θ. When the dependence is determined for the unknown vector θ, we have to concretize for which vectors θ it exists. The analysis has to be repeated, but with changed vector θ. In the vector θ, leftmost star will be changed with $' <'$, latter with '=', and at the last one with $' >'$. When we determine independence in some step, this vector θ need not be refined. The tree of analysis for m=2 is shown on the next figure:

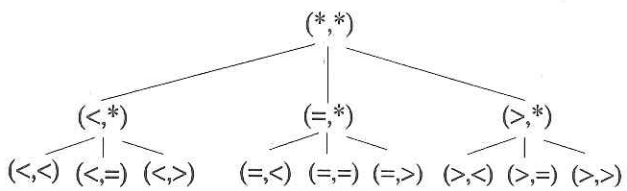

Figure 1.

The order of analysis is determined by PREORDER traversal of tree. If independence is determined for some node, the subtree whose root is that node, need not be analyzed.

3. Dependence testing on loops in which inner loop bounds are functions of outer loop indices

In practice, the loop with constant loop indices bounds are very infrequent. Loops of the form:

$$\begin{aligned} &DO\ 10\ \mathrm{I} = 1, N \\ &\quad DO\ 10\ \mathrm{J} = I + 1, N \end{aligned}$$

or

$$DO\ 10\ \mathrm{I} = 1, N$$
$$DO\ 10\ \mathrm{J} = 1, N - I + 1$$

are more frequent.

In the general case, we can assume that the inner loop bounds are functions of outer loop indices. We well look at the next, generalized loop:

$$DO\ 10\ \mathrm{I1} = T1, U1$$
$$DO\ 10\ \mathrm{I2} = t2(I1), u2(I1)$$
$$\vdots$$
$$DO\ 10\ \mathrm{I}m = tm(I1, I2, ..., Im-1), u2(I1, I2, ..., Im-1)$$
$$\{loopbody\}$$
$$10\ CONTINUE$$

For the application of Banerjee's test, we must compute loop indices bounds, in first. In the phase of compilation that is impossible because these values are different for all different iterations of outer loops. Because of that, we introduce the worst case assumptions: for the lower bound of index I_i we take t_imin, but for the upper bound u_imax. Our problem, now, is reduced to the determination of minimums of functions t_i, and maximums of functions u_i. We will assume that the functions t_i and u_i are linear. In that case, the functions t_i and u_i can be described as follows:

$$t_i = T_{i0} + \sum_{j=1}^{i-1} T_{ij} I_j \tag{13}$$

$$u_i = U_{i0} + \sum_{j=1}^{i-1} U_{ij} I_j \tag{14}$$

If we know the minimums and maximums of indices I_j (j[1,i-1]) we can compute the lower and upper bounds of index I_i:

$$T_i = t_{imin} = T_{i0} + \sum_{j=1}^{i-1} (T_{ij}^{+} I_{jmin} - T_{ij}^{-} I_{jmax}) \tag{15}$$

$$U_i = u_{imax} = U_{i0} + \sum_{j=1}^{i-1} (U_{ij}^{+} I_{jmax} - U_{ij}^{-} I_{jmin}) \tag{16}$$

I_jmin and I_jmax are the lower and upper bounds of index I_j (T_j, U_j). From there, the values T_i and U_i can be computed from the next formulas:

$$T_i = T_{i0} + \sum_{j=1}^{i-1} (T_{ij}^{+} T_j - T_{ij}^{-} U_j) \tag{17}$$

$$U_i = U_{i0} + \sum_{j=1}^{i-1} (U_{ij}^{+} U_j - U_{ij}^{-} T_j) \tag{18}$$

These formulas determine the order of computation of bounds, too. Obviouly, bounds T_i and U_i can be computed if the bounds T_j and U_j for $j \in [1, i-1]$ are known.

4. Influence of data dependence direction vector on loops

Banerjee's test checks data dependence for all data dependence direction vector, independently. It imposes a question: can the data dependence direction vector influence the coefficients T_i and U_i?

If $\theta_i = '>'$, the lower bound of index I_i can not be equal to the value computed by formula (17), because there is not a value of index I_i smaller than t_{imin}. Because of that, for $\theta_i = '>'$, the lower bound of index I_i is necessary to grow for 1.

Similarly, it can be shown that for $\theta_i = '<'$, the upper bound of index I_i is necessary to reduce for 1.

Definitive formulas for computation the loop indices bounds are:

$$T_i(\theta) = T_{i0} + \sum_{j=1}^{i-1} (T_{ij}^{+} T_j(\theta) - T_{ij}^{-} U_j(\theta)) + CT_i(\theta) \tag{19}$$

$$U_i(\theta) = U_{i0} + \sum_{j=1}^{i-1} (U_{ij}^{+} U_j(\theta) - U_{ij}^{-} T_j(\theta)) + CU_i(\theta) \tag{20}$$

where:

$$CT_i(\theta) = \begin{cases} 0 & \text{for } \theta_i \in (*, =, >) \\ 1 & \text{for } \theta_i = < \end{cases} \tag{21}$$

$$CU_i(\theta) = \begin{cases} 0 & \text{for } \theta_i \in (*, =, <) \\ -1 & \text{for } \theta_i = > \end{cases} \tag{22}$$

T_i is dependent on these elements of direction vector j for which is $j \leq i$. Thist means that the order of computing of bounds T_i and U_i is identical to the order of dependence testing for corresponding vectors (see figure 1.).

It makes a question: whay do we have to correct the bounds T_i and U_i adding the coefficients CT_i and CU_i, when all this job is done by Banerjee's test, too? In our cases, inner loop bounds are the functions of outer loop bounds. Because of that, the coefficients CT_i and CU_i influence on values of all coefficients T_j and U_j for j¿i. The original Banerjee's test is not taking that influence into consideration.

5. Banerjee's test modification

As we desctibed, we and Banerjee's test, too, correct the lower and upper bounds dependently on corresponding direction vector θ. If we use the our computed loop bounds for dependence testing by Banerjee's test, we take these corrections into consideration two times. Because of that, we have to do a little modification of Banerjee's test. We will take the cases i='¿' and i='¡', because in these cases we must modificate Banerjee's inequalities.

Banerjee's test begins from assumptions:

$$T_i \leq I_i \leq J_i - 1 \leq U_i - 1 \qquad \text{for } \theta_i = < \tag{23}$$

$$T_i + 1 \leq J_i + 1 \leq I_i \leq U_i \qquad \text{for } \theta_i = > \tag{24}$$

Instead of these, we take in next assumptions:

$$T_i(\theta) \leq I_i \leq J_i - 1 \leq U_i(\theta) \qquad \text{for } \theta_i = < \tag{25}$$

$$T_i(\theta) \leq J_i + 1 \leq I_i \leq U_i(\theta) \qquad \text{for } \theta_i = > \tag{26}$$

In that case modificated Banerjee's coefficients have a next form:

(27)

$$LC_i = \begin{cases} -(a_i^- + b_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) - b_i & for\ \theta_k = '<' \\ -(a_i - b_i)^-(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) & for \theta_k = '=' \\ -(b_i^+ - a_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) + b_i & for\ \theta_k = '>' \\ -(a_i^- + b_k^+)(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) & for \theta_k = ' *' \end{cases}$$

(28)

$$UC_k = \begin{cases} (a_i^+ - b_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) - b_i & for\ \theta_k = '<' \\ (a_i - b_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i & for \theta_k = '=' \\ (b_i^- - a_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) + b_i & for\ \theta_k = '>' \\ (a_i^+ + b_i^-)(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) & for\ \theta_k = ' *' \end{cases}$$

Proof:

We need a next lemma [1] for proving of our assertion:

Lemma 1. Let f(x,y)=ax+by denote a linear function, and $U \geq q_0$ a number.

1) $min(ax + by : 0 \leq y \leq x \leq U) = -(a - b^-)^- U$

2) $max(ax + by : 0 \leq y \leq x \leq U) = (a + b^+)^+ U$

Let f1 and f2 be the index functions defined by equations (1) and (2), respectively; and let is:

$$h(I, J) = f_1(I) - f_2(J). \tag{29}$$

In given region, dependence exists for given vector θ **iff** the function h(I,J) has a null into that region.

Thus,

$$min(h(I, J)) \leq 0 \leq max(h(I, J)) \tag{30}$$

By combining (4),(5) and (29) we obtain:

$$h(I,J) = \sum_{k=1,\theta_k='='}^{m} (a_k - b_k)I_k + \sum_{k=1,\theta_k\neq'='}^{m} a_k I_k - \sum_{k=1,\Theta_k\neq'='}^{m} b_k J_k + a_0 - b_0 \tag{31}$$

Let we take a case $\theta_i = <$:

We need minimum and maximum of function:

$$f = a_i I_i - b_i J_i \tag{32}$$

Next inequality is derived from our assumption (25):

$$0 \leq I_i - T_i(\theta) \leq J_i - T_i(\theta) - 1 \leq U_i(\theta) - T_i(\theta) \tag{33}$$

Because of that we will transform the function f (32) on follows:

$$a_i I_i - b_i J_i = -b_i(J_i - T_i(\theta) - 1) + a_i(I_i - T_i(\theta)) + (a_i - b_i)T_i(\theta) - b_i \tag{34}$$

Now, by using of Lemma 1. we obtain:

for $\theta_i = <$:

$$LC_i = -(a_i^- + b_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) - b_i \tag{35}$$

$$UC_i = (a_i^+ - b_i)(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) - b_i \tag{36}$$

Second correction of Banerjee's test was done for $\theta_i = >$. In that case, we, first, must transform our assumption (26) as follows:

$$0 \leq J_i - T_i(\theta) + 1 \leq I_i - T_i(\theta) \leq U_i(\theta) - T_i(\theta) \tag{37}$$

The function f (32) can be written as follows, too:

$$a_i I_i - b_i J_i = a_i(I_i - T_i(\theta)) - b_i(J_i - T_i(\theta) + 1) + (a_i - b_i)T_i(\theta) + b_i \tag{38}$$

Now, by using Lemma 1. we obtain:
for $\theta_i = >$:

$$LC_i = -(b_i^+ + a_i)^+(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) + b_i \tag{39}$$

$$UC_i = (a_i^+ + b_i^)(U_i(\theta) - T_i(\theta)) + (a_i - b_i)T_i(\theta) + b_i \tag{40}$$

This completes proof of our modification of Banerjee's test.

6. Conclusion

In designe process of a FORTRAN parallelizing compiler appeared a problem: how do we test the dependence on loops which bounds are not constant. In our testing examples, the most frequent cases were the loops in which inner loop bounds are linear function of outer loop indices. For dependence testing on loops with constant loop bounds we used Banerjee's test. It makes a question: is it possible to replace these functions with constants? We solwe this problem at next way: we change the lower bounds functions with their minimums, and we also change the upper bounds functions with their maximums. The expressions for computing these bounds values, when corresponding function are linear, are given in chapter 3 of this paper. We show that the loop bound values, in our case, are dependent on data dependence direction vector, too. We had to do some modification of Banerjee's test, because we take in the direction vector influence on loop bounds. Modificated Banerjee's test was presented in chapter 5.

References

[1] Z.Li, P.C.Yew, *An Eficient Interprocedural Analysis for Program Parallelization and Restructuring* (1988), ACM Press, New York.

[2] M. Walf, *Optimizing Supercompiler for Supercomputers*, Pitman, London, 1989.

[3] M. Walf, *The power test for data dependence*, IEEE Transactions on Parallel and Distributed Systems **3, No. 5** (September 1992), 591–601.

[4] H. Zima, B. Chapman, *Supercompilers for Parallel and Vector Computers*, ACM Press, New York, 1988.

[5] T. M. O'Keefe, H. G. Deitz, *Loop Coalescing and Scheduling for Barier MIMD Architectures*, IEEE Transactions on Parallel and Distributed Systems **4, No. 9** (September 1993), 1060–1064.

[6] T. H. Tzen, L. M. Ni, *Dependence Uniformization: A Loop Parallelization Technique*, IEEE Transactions on Parallel and Distributed Systems **4, No. 5** (May 1993.), 547–558.

Suzana Stojković, Faculty of Electronic Engeneering, Computeer Science departmetnt, Beogradska 14, 18000 Niš