30

Extending the Banerjee-Wolfe Test to Handle Execution Conditions

D. Klappholz
Stevens Institute of Technology, Hoboken, New Jersey

X. Kong
Sun Microsystems, Inc., Mountain View, California

Abstract
The Banerjee-Wolfe test is one of the major data dependence tests used in automatic parallelization of sequential code. Though it is only an approximate test, its relatively high accuracy and its relatively low cost account for its great popularity. Being an approximate test, the Banerjee-Wolfe test does, however, sometimes result in a loss of parallelism. One of its potential sources of failure is the fact it does not traditionally take execution conditions into account. The purpose of the present paper is to show that the Banerjee-Wolfe test may be extended to handle simple execution conditions without significant additional cost.

1. Introduction
The Banerjee test [1] and its extension, the Banerjee-Wolfe test [6] are relatively inexpensive, and extremely accurate [8]-[10], though ultimately only approximate, data dependence tests. Although they usually find safe parallelism, being approximate tests they sometimes fail in disappointingly simple situations. Consider, for example, the Gaussian elimination code of Figure 1. It is easy to see that both the I-loop and the J-loop may be converted into for all loops, but that this would not be the case if the execution condition attached to the J-loop were removed/ignored.

If we take the Banerjee-Wolfe Test to mean the way the test is traditionally applied, then it indeed fails to find the indicated parallelism. If, on the other hand, the Banerjee-Wolfe Test refers to the test's underlying mathematics, then matters are different. In Section 3 we
show how the mathematics underlying the Banerjee-Wolfe test, appropriately applied, finds the parallelism in the code of Figure 1. In Section 4 we show how the Banerjee-Wolfe test may be extended to handle simple execution conditions in general -- with no more than a moderate increase in cost.

```
for K = 1 to 100
    for I = 1 to 100
        if I ≠ K then
            for J = 1 to 100
                A(I,J) = A(I,J) - A(I,J)*(A(K,J)/A(K,K))
            end
        end
end
```

Gaussian Elimination Code

Figure 1

In a code of any significant size there are, typically, thousands, tens of thousands, or even more pairs of array references which must be tested for dependence. The Banerjee-Wolfe test must, typically, be applied multiple times for each such pair -- once for each direction vector of interest. As a consequence, some writers of production compilers, considering even the repeated application of the Banerjee-wolfe test to be excessively time consuming, test only for a subset of the potentially useful direction vectors. No extension of the Banerjee-Wolfe test is likely to be adopted by commercial compiler writers unless its cost is only insignificantly greater than that of the Banerjee-Wolfe test. More expensive tests can be useful to handle special cases, most likely, as Wolfe and Tseng suggest [14], in parallelization tools, and under programmer control. The extended test presented here, since it handles only simple execution conditions, certainly does not eliminate the need for such tests, but does suggest that such tests need not be resorted to as frequently as might have been expected.

2. The Banerjee-Wolfe Test

The data dependence problem is that of determining whether a linear equation with integer coefficients has an integer solution within some region of a multi-dimensional Euclidian space. The region is defined by a specification of the extreme values assumed by each of the variables together with a set of equalities/inequalities (direction vector constraints) among some pairs of variables.