CS/ECE 560 Foundations of Fine-Grained Parallelism — Spring 2012 HW #4
Representing Dependences and Transformations in Frameworks Due February 22, 2012 at 11:59pm

ALL of the homeworks and project writeups in this course must be written using latex. See the template latex file (cs560-template.tex) at http://www.cs.colostate.edu/~cs560/Spring2012/assignments.php. Homework assignments are to be completed individually. Homeworks need to be submitted electronically via RamCT AND by email to cs560@cs.colostate.edu by 11:59pm on the due date. For this homework submit a tar ball with all of the latex source and a Makefile to build the writeup pdf. Total points: 100

1. [70 points] Data dependence analysis and unimodular transformations

```
for (i=0; i<N; i++) {
  for (j=3; j<M; j++) {
    A[j] = B[ j ][ i ] - 99;
    B[ j-2][ i+1 ]  = sin(i * j *3);
  }
}
```

(a) For the above program, what is the direction vector for the output dependences between writes to A[i]? (Hint: Recall that $(*, <), (*, =)$, and $(*, >)$ are not legal dependence vectors.) Show the setup for the data dependence problem and how you derive the direction vector. (Hint: you might want to use an eqnarray in latex to list the constraints.)

(b) For the above program, what is the distance vector for the flow dependence? Show the setup for the data dependence problem and how you derive the distance vector.

(c) Can we parallelize the i loop or the j loop. Please explain why or why not for each loop.

(d) Is the problem 1 loop fully permutable? Why or why not? What is the unimodular transformation matrix that specifies a permutation of the `i` and `j` loops in the program for problem 1? Describe the legality or illegality of permutation by applying the unimodular transformation matrix to the direction vectors.

2. [30 points] Loop Fission and the Kelly and Pugh Transformation Framework

(a) Show whether loop fission is legal or illegal for the following program using the K&P transformation framework notation. To do this, set up the dependence analysis problem, find the lexicographically non-negative dependence relation, show the fission representation using a mapping, and then show how you check the legality.

```
for (i=0; i<N; i++) {
    A[ i ] = ... ;
    ... = A[ i - 1 ];
}
```

(b) Show whether loop fission is legal or illegal for the following program using the K&P transformation framework. To do this, set up the dependence analysis problem, find the lexicographically non-negative dependence relation, show the fission representation using a mapping, and then show how you check the legality.

```
for (i=0; i<N; i++) {
    A[ i ] = ... ;
    ... = A[ i + 2 ];
}
```

(c) For both part (a) and part (b) indicate whether the data dependence relation represents a flow, anti, or output dependence.