419 Compilers II

Tutorial exercise 1

If you did course 332, Parallel Architectures, you should remember the examples in Question 1 — look over them quickly and move onto Question 2.

Make sure that you have a look at Question 3.

Question 1: Basic dependence analysis

For each of the following loops, draw the dependence graph, and if possible write down a vectorised version with the same input-output behaviour:

1. ```
   for i := 1 to N do
       A[i+1] := A[i] + 1;
   ```

2. ```
   for i := 1 to 10 do
       A[10*i] := A[11*i]
   ```

3. ```
   for i := 1 to N do
       A[i] := B[i];
       C[i] := A[i] + B[i];
       E[i] := C[i+1];
   ```

Question 2: Wavefronts

This question is intended to prepare for the section of the course on using unimodular matrices to capture loop transformations. It concerns the following example:

```
for i1 := 0 to 3 do
  for i2 := 0 to 3 do
    A[i1,i2] := A[i1-1,i2] + A[i1,i2-1]
```

1. (a) Draw the iteration space for this loop and mark on it the dependences.

   (b) Write down the dependence direction vector for each dependence.
```

(c) Write down the dependence *distance* vector for each dependence. Recall that the dependence distance is the number of iterations which separate a pair of references to the same location, while the dependence directions is its sign.

(d) Can this loop nest be interchanged?

(e) Can this loop be vectorised or parallelised?

(f) Can this loop be tiled (i.e. executed block-wise to improve cache behaviour)?

2. Consider this loop:

```
for k1 := 0 to 3 do
  for k2 := k1 to k1+3 do
    A[k1,k2-k1] := A[k1-1,k2-k1] + A[k1,k2-k1-1]
```

(a) Draw the iteration space for this loop and mark on it the dependences.

(b) Write down the dependence direction vector for each dependence.

(c) This loop has the same effect as the earlier loop, but has been *skewed*. What does this mean — what is the relationship between the two loops?

(d) Can this loop nest be interchanged?

(e) Can this loop be vectorised or parallelised?

(f) Can this loop be tiled (i.e. executed block-wise to improve cache behaviour)?

Question 3: Less basic dependence analysis

For each of the following loops, draw the dependence graph, and if possible write down a vectorised version with the same input-output behaviour:

(a)
```
for i := 1 to 10 do
    A[i*2-1] := A[i*2-N*4]
```
where N is not known at compile-time.

(b)
```
for i := 1 to 10 do
    for j := 1 to 11 do
      A[i*10+j] := A[i*10+j-1]
```

(c)
```
for i := 1 to 100 do
    for j := 1 to i-1 do
      A[i*100+j] := A[j*100+i]
```
(Note that this loop is "triangular").

Paul Kelly, Imperial College, February 1997.

419 Compilers II Tutorial 1

Sample Solutions

Question 2: Wavefronts

This question is intended to prepare for the section of the course on using unimodular matrices to capture loop transformations. It concerns the following example:

```
for i1 := 0 to 3 do
  for i2 := 0 to 3 do
    A[i1,i2] := A[i1-1,i2] + A[i1,i2-1]
```

1. (a) Draw the iteration space for this loop and mark on it the dependences.

$$
\begin{array}{ccccccc}
S^{00} & \underset{\delta}{\longrightarrow} & S^{01} & \underset{\delta}{\longrightarrow} & S^{02} & \underset{\delta}{\longrightarrow} & S^{03} \\
\delta\downarrow & & \delta\downarrow & & \delta\downarrow & & \delta\downarrow \\
S^{10} & \underset{\delta}{\longrightarrow} & S^{11} & \underset{\delta}{\longrightarrow} & S^{12} & \underset{\delta}{\longrightarrow} & S^{13} \\
\delta\downarrow & & \delta\downarrow & & \delta\downarrow & & \delta\downarrow \\
S^{20} & \underset{\delta}{\longrightarrow} & S^{21} & \underset{\delta}{\longrightarrow} & S^{22} & \underset{\delta}{\longrightarrow} & S^{23} \\
\delta\downarrow & & \delta\downarrow & & \delta\downarrow & & \delta\downarrow \\
S^{30} & \underset{\delta}{\longrightarrow} & S^{31} & \underset{\delta}{\longrightarrow} & S^{32} & \underset{\delta}{\longrightarrow} & S^{33}
\end{array}
$$

   (b) Write down the dependence direction vector for each dependence.

   - S $\delta_{<,.}$ S due to the first reference to `A`.
   - S $\delta_{.,<}$ S due to the second reference to `A`.

   (c) Write down the dependence *distance* vector for each dependence. Recall that the dependence *distance* is the number of iterations which separate a pair of references to the same location, while the dependence directions is its sign.

   - (1,0) for the first reference to `A`.
   - (0,1) for the second reference to `A`.

   (d) Can this loop nest be interchanged?

   Interchange is clearly valid since the execution order of the interchanged loop nest would respect the dependences of the original loop.

   (e) Can this loop be vectorised or parallelised?

   There is a loop carried data dependence in the inner loop which blocks vectorisation. Similarly, the loop-carried data dependence in the outer loop prevents parallel execution of `i1` iterations. Interchanging the loop nest does not help.

(f) Can this loop be tiled (i.e. executed block-wise to improve cache behaviour)?

Yes. To see this consider larger loop limits, strip-mine then interchange:

```
for i1 := 0 to 29 do
  for i2 := 0 to 29 do
    A[i1,i2] := A[i1-1,i2] + A[i1,i2-1]
```

```
for ii1 := 0 to 29 step 10 do
  for i1 := ii1 to ii1+10 do
    for ii2 := 0 to 29 step 10 do
      for i2 := ii2 to ii2+10 do
        A[i1,i2] := A[i1-1,i2] + A[i1,i2-1]
```

```
for ii1 := 0 to 29 step 10 do
  for ii2 := 0 to 29 step 10 do
    for i1 := ii1 to ii1+10 do
      for i2 := ii2 to ii2+10 do
        A[i1,i2] := A[i1-1,i2] + A[i1,i2-1]
```

To see how this works, think about the order in which the 30×30 iteration space is traversed.

2. Consider this loop:

```
for k1 := 0 to 3 do
  for k2 := k1 to k1+3 do
    A[k1,k2-k1] := A[k1-1,k2-k1] + A[k1,k2-k1-1]
```

Solutions for the remaining questions are not available but should become apparent from the lecture notes. See also Banerjee's paper "Unimodular Transformations of Double Loops".

Paul Kelly, Imperial College, February 1997.