CS314 Fall 2018

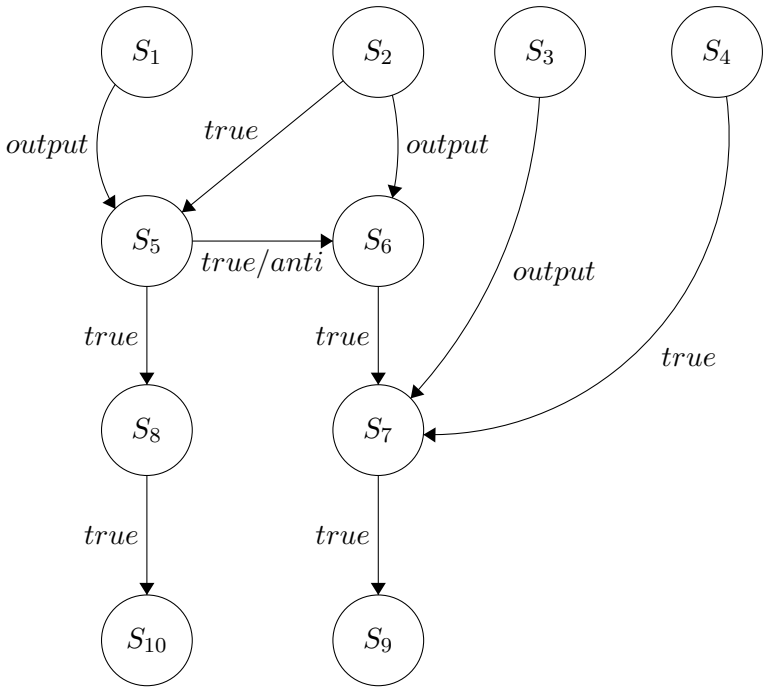
Assignment 8 Solutions

Problem 1 - Dependencies

Recall the following program (a sequence of instructions), with each instruction labeled as S<num>:

```
S1:   a := 4;
S2:   b := 2;
S3:   c := 5;
S4:   read(d);
S5:   a := b + 3;
S6:   b := a - 3;
S7:   c := d * b;
S8:   e := a + 6;
S9:   print(c);
S10:  print(e);
```

(a) The statement-level dependence graph is as follows:



(b)

The sequential execution time is 10 cycles (1 per instruction). For parallel execution time, note that a statement can be executed once all statements it depends on have been executed. You can schedule statements based on the topological sort of their dependences. The critical path in this graph is $S_1 \rightarrow S_5 \rightarrow S_6 \rightarrow S_7 \rightarrow S_9$ for a total of 5 cycles. Another critical path is $S_2 \rightarrow S_5 \rightarrow S_6 \rightarrow S_7 \rightarrow S_9$. One way to separate statements into groups that can be executed parallel is:

Cycle 1: {S_1, S_2, S_3, S_4}
Cycle 2: {S_5}
Cycle 3: {S_6, S_8}
Cycle 4: {S_7, S_{10}}
Cycle 5: {S_9}

Problem 2 - Dependence Analysis

Give the source and sink references, the type (whether a dependence is **true**, **anti**, or **output**), and the distance vectors for all dependences in the following loops.

(a)
```
do i = 3, 100
    a(i) = a(i - 1) + a(i + 1) - a(i - 2)
enddo
```

Source Reference	Sink reference	Type	Distance vector
a(i)	a(i - 1)	true	(1)
a(i)	a(i - 2)	true	(2)
a(i + 1)	a(i)	anti	(1)

(b)
```
do i = 2, 100
    a(3 * i) = a(3 * i - 3) + a(3 * i + 3)
enddo
```

Source Reference	Sink reference	Type	Distance vector
a(3i + 3)	a(3i)	anti	(1)
a(3i)	a(3i - 3)	true	(1)

(c)
```
do i = 1, 10
    a(i) = a(5) + a(i)
enddo
```

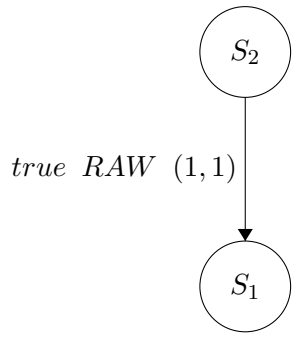
Source Reference	Sink reference	Type	Distance vector
a_W(i)	a(5)	true	
a(5)	a_W(i)	anti	

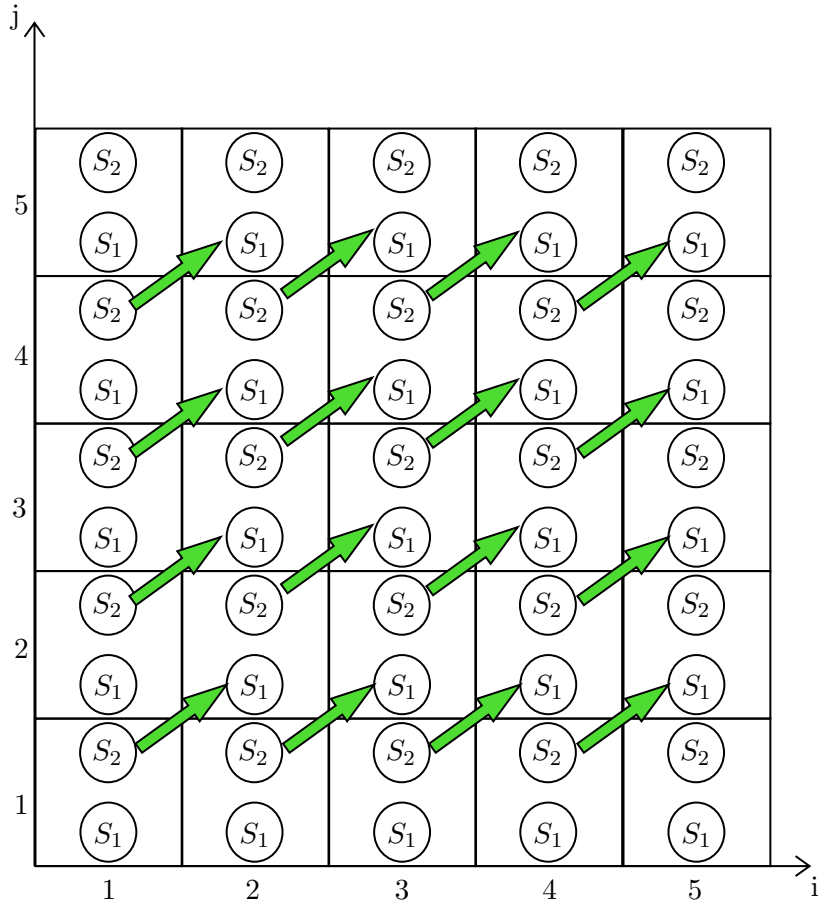
Problem 3 - Loop Parallelization

Given the following nested loop:

```
do i = 2, 100
    do j = 2, 100
S1:     a(i, j) = b(i - 1, j - 1) + 2
S2:     b(i, j) = i + j - 1
    enddo
enddo
```

(a) The statement-level dependence graph with distance vectors, along with the iteration space graph, are given below:





(b) The given loop above, in its current form, is parallelizable by fixing i. By fixing i, the j-loop can be parallelizable, as shown in the dependence graph for statement instances.

(c) First, note that for statements S_1 and S_2, using the distance vector found in 3(a):

$$i_2 = i_1 + 1 \quad , \quad j_2 = j_1 + 1$$

Following Lecture 19, we wish to find $C_{11}, C_{12}, C_{21}, C_{22}, d_1$, and d_2 such that:

$$p(S_1) = C_{11} * i + C_{12} * j + d_1$$

$$p(S_2) = C_{21} * i + C_{22} * j + d_2$$

and

$$C_{11} * i_1 + C_{12} * j_1 + d_1 = C_{21} * i_2 + C_{22} * j_2 + d_2$$

Substitution gives the following:

$$C_{11} * i_1 + C_{12} * j_1 + d_1 = C_{21} * (i_1 + 1) + C_{22} * (j_1 + 1) + d_2$$

In matrix form, this is:

$$\begin{pmatrix} C_{11} & C_{12} \end{pmatrix} * \begin{pmatrix} i_1 \\ j_1 \end{pmatrix} + d_1 = \begin{pmatrix} C_{21} & C_{22} \end{pmatrix} * \begin{pmatrix} i_1 + 1 \\ j_1 + 1 \end{pmatrix} + d_2$$

Simplifying this results in the following:

$$\begin{pmatrix} C_{11} - C_{21} & C_{12} - C_{22} \end{pmatrix} * \begin{pmatrix} i_1 \\ j_1 \end{pmatrix} + (d_1 - d_2 - (C_{21} + C_{22})) = 0$$

Solving for this gives:

$$C_{11} = C_{21}, \quad C_{12} = C_{22}, \quad (d_1 - d_2) = C_{21} + C_{22}$$

Any values satisfying this relation will be correct. Since the hint given suggested $d_1 = d_2 = 0$, set $C_{11} = C_{21} = 1$ and $C_{12} = C_{22} = -1$. Thus,

$$p(S_1) = i - j$$

$$p(S_2) = i - j$$

(d) We now generate the two-level loop for the affine schedule given above. First, we find the range for p. Since i ranges from 2 to 100, and j ranges from 2 to 100, p ranges from -98 to 98. We then have the following:

```
do p = -98, 98
    do i = 2, 100
        do j = 2, 100
            if (i - j == p)
                a(i, j) = b(i - 1, j - 1) + 2
                b(i, j) = i + j - 1
            endif
        enddo
    enddo
enddo
```

Now we can eliminate j by doing Fourier-Motzkin elimination: notice $i - j = p$. So, since j ranges from 2 to 100,

$$2 \leq i - p \leq 100$$

Solving for this gives: $i \leq p + 100$ and $i \geq p + 2$. Moreover, note that i ranges from 2 to 100 as well. Therefore, we have:

```
do p = -98, 98
    do i = max(2, p + 2), min(100, p + 100)
        a(i, i - p) = b(i - 1, i - p - 1) + 2
        b(i, i - p) = i + (i - p) - 1
    enddo
enddo
```

which is the necessary two-level loop code.