

## Loop Transformations for Parallelism & Locality

### Last time

- Unimodular transformation framework
- Loop permutation
- Loop reversal
- Loop skewing

### Today

- Kelly & Pugh transformation framework
- Loop transformations
  - Loop fusion
  - Loop fission

## Loop Fusion Example

### What are the dependences?

```

do i = 1, n
s1   A(i) = B(i) + 1
enddo

do i = 1, n
s2   C(i) = A(i) / 2
enddo

do i = 1, n
s3   D(i) = 1 / C(i+1)
enddo
    
```

Dependence arrows:  $s_1 \delta^f s_2$  (blue),  $s_2 \delta^f s_3$  (blue)

### What are the dependences?

```

do i = 1, n
s1   A(i) = B(i) + 1
s2   C(i) = A(i) / 2
s3   D(i) = 1 / C(i+1)
enddo
    
```

Dependence arrows:  $s_1 \delta^f s_2$  (blue),  $s_3 \delta^a s_2$  (red)

Fusion changes the dependence between  $s_2$  and  $s_3$ , so fusion is illegal

## Kelly and Pugh Transformation Framework

### Specify iteration space as a set of integer tuples

$$\{[i, j] \mid 1 \leq i, j \leq n\}$$

### Specify data dependences as mappings between integer tuples (i.e., data dependence relations)

$$\{[i, j] \rightarrow [i', j'] \mid (i = i' - 1) \wedge (j = j' - 1) \wedge (1 \leq i, j, i', j' \leq n)\}$$

### Specify transformations as mappings between integer tuples

$$\{[i, j] \rightarrow [i', j'] \mid (i' = j) \wedge (j' = i)\}$$

### Execute iterations in transformed iteration space in lexicographic order

## Specifying Loop Fusion in Kelly and Pugh Framework

### Specify iteration space as a set of integer tuples

$$IS_1 = \{[1, i_1, 1] \mid 1 \leq i_1 \leq n\}$$

$$IS_2 = \{[2, i_2, 1] \mid 1 \leq i_2 \leq n\}$$

$$IS_3 = \{[3, i_3, 1] \mid 1 \leq i_3 \leq n\}$$

$$IS = IS_1 \cup IS_2 \cup IS_3$$

### Specify data dependences as mappings between integer tuples (i.e., data dependence relations)

$$D_{12} = \{[1, i_1, 1] \rightarrow [2, i_2, 1] \mid i_1 = i_2\}$$

$$D_{23} = \{[2, i_2, 1] \rightarrow [3, i_3, 1] \mid i_2 = i_3 + 1\}$$

$$D = D_{12} \cup D_{23}$$

### Specify transformations as mappings between integer tuples

$$T_1 = \{[1, i_1, 1] \rightarrow [1, i'_1, 1] \mid i'_1 = i_1\}$$

$$T_2 = \{[2, i_2, 1] \rightarrow [1, i'_2, 2] \mid i'_2 = i_2\}$$

$$T_3 = \{[3, i_3, 1] \rightarrow [1, i'_3, 3] \mid i'_3 = i_3\}$$

$$T = T_1 \cup T_2 \cup T_3$$

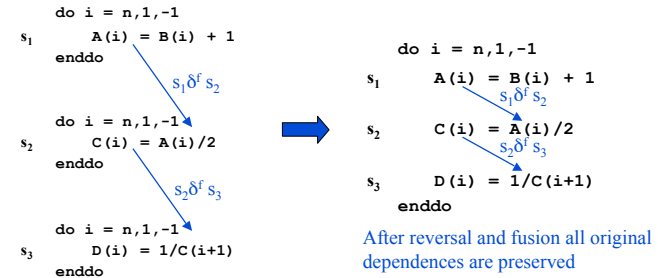
## Checking Legality in Kelly & Pugh Framework

For each dependence,  $[I] \rightarrow [J]$  the transformed  $I$  iteration must be executed after the transformed  $J$  iteration.

## Loop Fusion Example (cont)

Loop reversal is legal for the original loops

- Does not change the direction of any dep in the original code
- Will reverse the direction in the fused loop:  $s_3 \delta^a s_2$  will become  $s_2 \delta^f s_3$

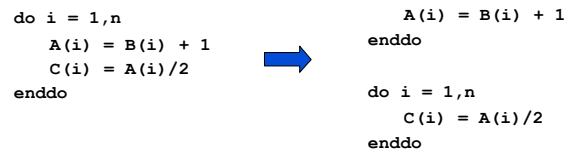


## Loop Fission (Loop Distribution)

**Idea**

- Split a loop nest into multiple loop nests (the inverse of fusion)

**Example**



**Motivation?**

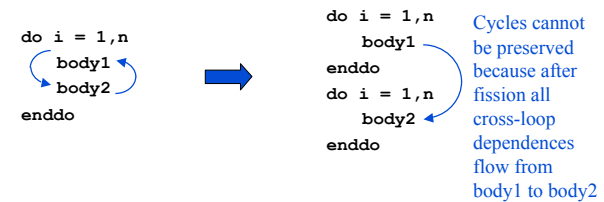
- Produces multiple (potentially) less constrained loops
- May improve locality
- Enable other transformations, such as interchange

**Legality?**

## Loop Fission (cont)

**Legality**

- Fission is legal when the loop body contains no cycles in the dependence graph



## Loop Fission Example

Recall our fusion example

```

do i = 1, n
s1   A(i) = B(i) + 1
enddo

do i = 1, n
s2   C(i) = A(i) / 2
enddo

do i = 1, n
s3   D(i) = 1 / C(i+1)
enddo
    
```

Can we perform fission on this loop?

```

do i = 1, n
s1   A(i) = B(i) + 1
enddo

do i = 1, n
s2   C(i) = A(i) / 2
enddo

do i = 1, n
s3   D(i) = 1 / C(i+1)
enddo
    
```

## Loop Fission Example (cont)

If there are no cycles, we can reorder the loops with a topological sort

```

do i = 1, n
s1   A(i) = B(i) + 1
enddo

do i = 1, n
s3   D(i) = 1 / C(i+1)
enddo

do i = 1, n
s2   C(i) = A(i) / 2
enddo
    
```

Can we perform fission on this loop?

```

do i = 1, n
s1   A(i) = B(i) + 1
enddo

do i = 1, n
s2   C(i) = A(i) / 2
enddo

do i = 1, n
s3   D(i) = 1 / C(i+1)
enddo
    
```

## Concepts

### Loop transformation

- Loop fusion
- Loop fission

### Kelly & Pugh Transformation Framework

- iteration spaces as constrained sets of integer tuples
- data dependences as mappings between integer tuples
- transformations as mappings between integer tuples

## Next Time

### Lecture

- Tiling