Basic Loop Parallelization

- Data dependences
- Loop parallelization
- How to extract data dependence information

Parallelization Goal

- DoAll loops: Loops whose iterations can execute in parallel
- Example

```
for (i = 0; i < n; i++)
    A[i] = B[i] + C[i]
```

  - Statically assign each iteration to a processor

Data Dependence of Scalars

  - True dependence
  - Anti dependence
  - Output dependence
- Data dependence exists from a dynamic instance i to i’ iff
  - either i or i’ is a write operation
  - i and i’ refer to the same variable
  - i executes before i’

Lack of Data Flow Information

- Example
  (1) x = f()
  (2) x = g()
  (3)   = x

- Standard data dependences are flow-insensitive
  - Don't care if there is no intervening write between a write and a read; (3) is dependent on (1) in the above
- Standard data dependence
  - Alias analysis
  - Memory disambiguation

Data Dependences for Arrays

```
for (i = 2; i < 5; i++)
    A[i] = A[i-2] + 1

for (i = 0; i < 3; i++)
    A[i] = A[i] + 1
```

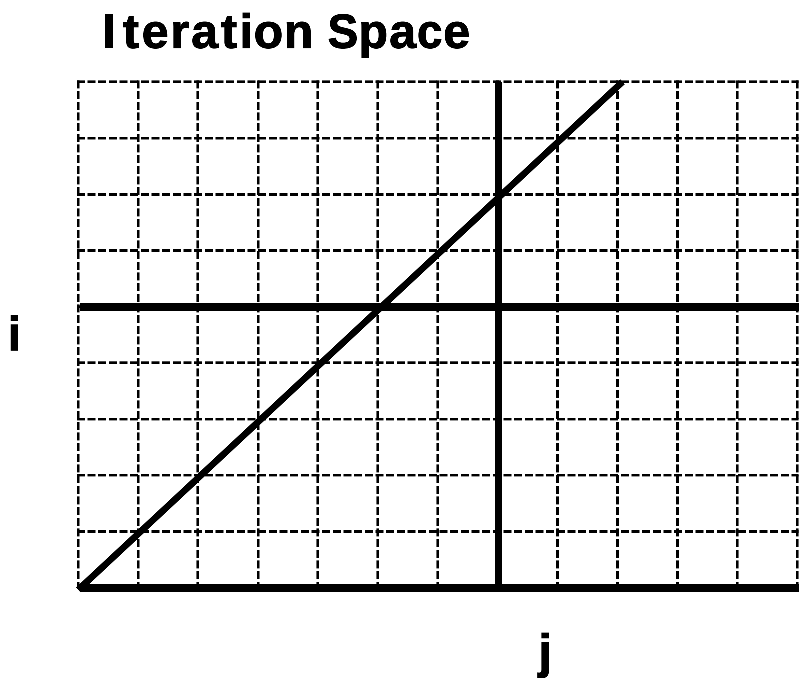
- Recognize DOALL loops (intuitively)
  - Find data dependences in loops
  - No dependences crossing iteration boundaries
    $\Rightarrow$ parallelizable

Iteration Space

- *n*-dimensional discrete space for *n*-deep loops

Program

```
FOR i = 0 TO 5
   FOR j = i TO 7
      A[i,j] = A[i,j-1] + 1
```



Iteration Space

- Iteration is represented as coordinates in iteration space
- Sequential execution order of iteration: Lexicographic order

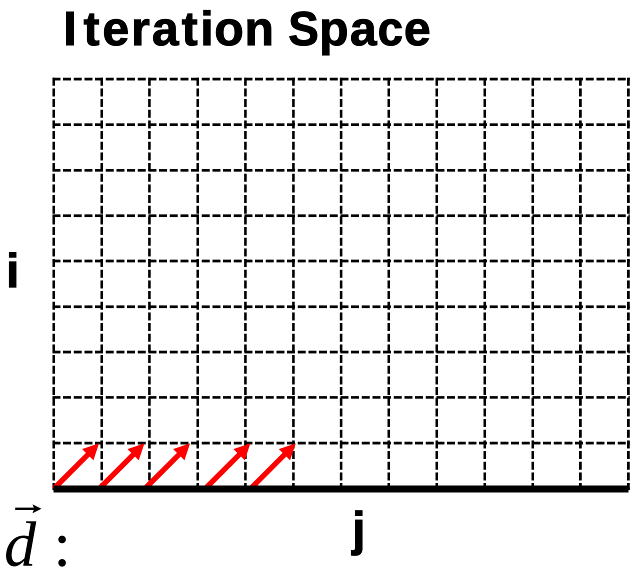
  $$[0,0], [0,1], .. ,[0,7], [1,1], \; ... \;\; [1,6]$$

- Iteration $\vec{i}$ is lexicographically less than $\vec{i}'$, iff

  $$\exists c \; s.t. (i_1, .. i_{c-1}) = (i'_1, .. i'_{c-1}) \; and \; i_c < i'_c$$

Distance Vectors

```
FOR (i = 0; i < n; i++)
   FOR (j = 0; j < n; j++)
      A[i,j] = A[i-1,j-1] + 1
```

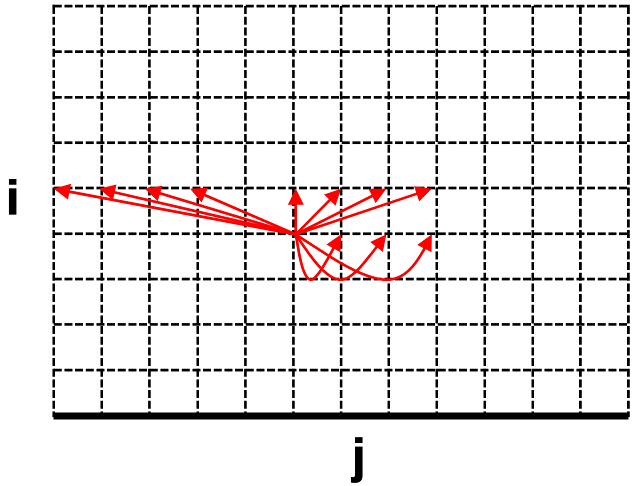


- Distance vector=[1,1]
- A loop has a distance vector $\vec{d}$:
  if there exists data dependence from a node $\vec{i}$ to a later node $\vec{i}'$ and $\vec{d} = \vec{i}' - \vec{i}$
- Since $\vec{i} \leq \vec{i}', \vec{d} \geq 0$
  $\vec{d}$ is lexicographically greater than or equal to 0

Distance Vectors

```
FOR (i = 0; i < n; i++)
   FOR (j = 0; j < n; j++)
      a = a + a;
```

Iteration Space



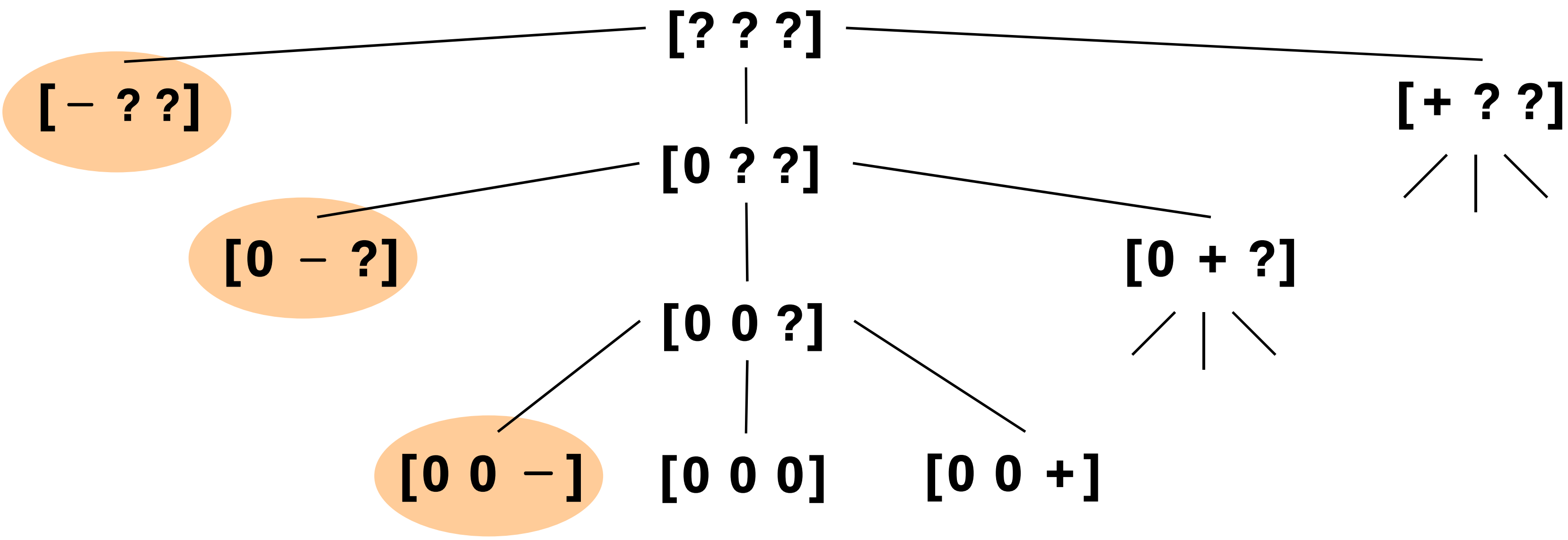
- Distance vector(infinitely large set)
  ([0,0][0,1],..[0,n])([1,-n],..,[1,0],..[1,n])..([n,-n],..[n,0],..[n,n])
- Summarized as direction vector (0 or lexicographically positive)
  ([0,0][0,+][+,-][+,0][+,+])

Direction Vectors

- Common notations

  (0 =),(+ <),(− >),(+/− *)

- What can the direction vectors for 3-D loops be like?



Test for Parallelization

- Example

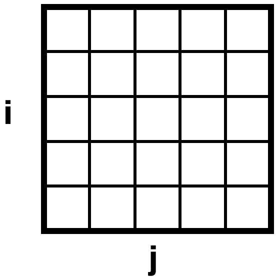
```
for (j = 0; j < n; j++)
    A[j] = A[j] + 1;
```

- Distance Vector: [0]
- Test for parallelization:
  - A loop with data dependence $\vec{d} \in D$ is parallelizable if for all $\vec{d}$ = (0)

Parallelization of Loops

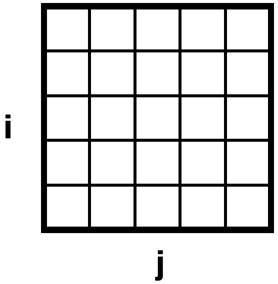
```
FOR (i = 0; i < n; i++)
   FOR (j = 0; j < n; j++)
      A[i,j] = A[i,j-1] + 1
```

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



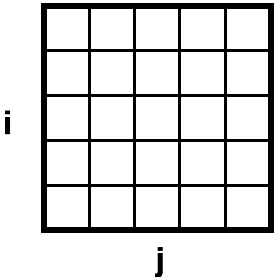

```
FOR (i = 0; i < n; i++)
   FOR (j = 0; j < n; j++)
      A[i,j] = A[i-1,j] + 1
```

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$




```
FOR (i = 0; i < n; i++)
   FOR (j = 0; j < n; j++)
      A[i,j] = A[i-1,j-1] + 1
```

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



- $\vec{d} = (d_1, d_2, .., d_n)$ is loop-carried at level *i* if d_i is the first non-zero element ⇒ does not affect the parallelizability of inner loop (= 0: no dependence; ≠ 0: second-order effect)

Test for Parallelization

- The *i* th loop of an *n*-dimensional loop is parallelizable if there does not exist any level *i* dependences
- The *i* th loop of an *n*-dimensional loop is parallelizable for all dependences $\vec{d} = (d_1, .., d_n)$, either
  - $(d_1, .., d_{i-1}) > 0$
  - $(d_1, .., d_i) = 0$

Improving Parallelizability

- Scalar Privatization

```
float t;
for (i = 0; i < n; i++) {
    t    = A[i];
    b[i] = t*t;
}
```

```
for (i = 0; i < n; i++)
{
    float t;
    t    = A[i];
    b[i] = t*t;
}
```

Summary

- Data Dependence
  - read/write
  - same variable
  - in direction of sequential ordering
- Representation
  - iteration space
  - dependence vectors: distance vectors, direction vectors
- Parallelization Testing