ESPECIFICACIÓN FORMAL DE SISTEMAS CONCURRENTES

Dr. Mario Rossainz López

Programación Concurrente y Paralela

Otoño 2022

NRC: 60898

¿Qué se puede ejecutar Concurrentemente?

No todas las partes de un programa pueden ejecutarse de forma concurrente

```
x ← x+1;
y ← x+2;
```

La primera sentencia se tiene que ejecutar antes que la segunda

```
x ← 1;
y ← 2;
Z ← 3;
```

El orden de ejecución no interviene en el resultado final

¿Qué se puede ejecutar Concurrentemente?

x ← 1;

y ← 2;

Z ← 3;

¿Se incrementa la velocidad del sistema?

x ← x+1;

y ← x+2;

Condiciones de Bernstein

- A.J. Bernstein (1966) definió unas condiciones para determinar si dos conjuntos de instrucciones *Si y Sj* se pueden ejecutar concurrentemente.

- L(Sk)={a1, a2, . . . , an} es el **conjunto de lectura** del conjunto de instrucciones *Sk* y esta formado por todas las variables cuyos valores son referenciados (se leen) durante la ejecución de instrucciones en Sk.

- E(Sk)={b1, b2, . . . , bm} es el **conjunto de escritura** del conjunto de instrucciones *Sk que esta formado por todas las variables cuyos valores son actualizados (se escriben) durante la ejecución de las instrucciones en* Sk.

Condiciones de Bernstein

- Para que dos conjuntos de instrucciones *Si y Sj* se puedan ejecutar concurrentemente, se tiene que cumplir lo siguiente:

- $L(Si) \cap E(Sj) = \emptyset$
- $E(Si) \cap L(Sj) = \emptyset$
- $E(Si) \cap E(Sj) = \emptyset$

Condiciones de Bernstein

- EJEMPLO:

- S1: a ← x + y;
- S2: b ← z - 1;
- S3: c ← a - b;
- S4: w ←c + 1;

- Se calculan los conjuntos de lectura/escritura :

- L(S1) = { x , y }
- L(S2) = { z }
- L(S3) = { a , b }
- L(S4) = { c }

E(S1) = { a }
E(S2) = { b }
E(S3) = { c }
E(S4) = { w }

Condiciones de Bernstein

- Se aplican las condiciones de Bernstein a cada par de sentencias:

Entre S1 y S2:
L(S1) ∩ E(S2) = ∅
E(S1) ∩ L(S2) = ∅
E(S1) ∩ E(S2) = ∅

Entre S1 y S3:
L(S1) ∩ E(S3) = ∅
E(S1) ∩ L(S3) = a
E(S1) ∩ E(S3) = ∅

Entre S1 y S4:
L(S1) ∩ E(S4) = ∅
E(S1) ∩ L(S4) = ∅
E(S1) ∩ E(S4) = ∅

Entre S2 y S3:
L(S2) ∩ E(S3) = ∅
E(S2) ∩ L(S3) = b
E(S2) ∩ E(S3) = ∅

Entre S2 y S4:
L(S2) ∩ E(S4) = ∅
E(S2) ∩ L(S4) = ∅
E(S2) ∩ E(S4) = ∅

Entre S3 y S4:
L(S3) ∩ E(S4) = ∅
E(S3) ∩ L(S4) = c
E(S3) ∩ E(S4) = ∅

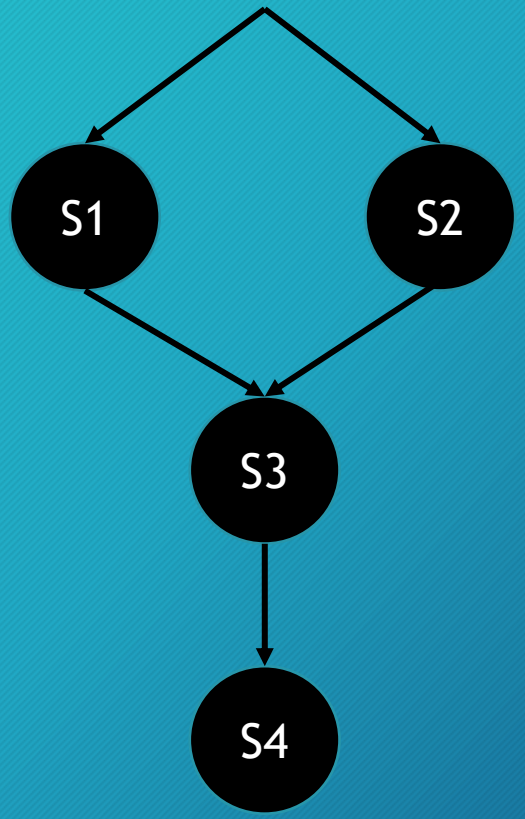
Condiciones de Bernstein

	S1	S2	S3	S4
S1	-----	1	0	1
S2	-----	-----	0	1
S3	-----	-----	-----	0
S4	-----	-----	-----	-----

Grafos de Precedencia

- Es una notación gráfica
- Es un grafo dirigido acíclico
- Cada nodo representa una parte (conjunto de instrucciones) del programa
- Una flecha desde un nodo A hasta un nodo B, representa que B sólo puede ejecutarse cuando A haya finalizado (dependencia)
- Si aparecen dos nodos en paralelo (en el mismo nivel dentro del grafo), querra decir que se pueden ejecutar concurrentemente

Grafos de Precedencia



Sentencias COBEGIN-COEND

- Todas aquellas acciones que pueden ejecutarse concurrentemente las introducimos dentro del par de instrucciones: COBEGIN/COEND

```
begin
    cobegin
        a ← x+y;
        b ← z-1;
    coend;
    c ← a-b;
    w← c+1;
end.
```

Las instrucciones dentro del par cobegin/coend pueden ejecutarse en cualquier orden, mientras que el resto se ejecuta de manera secuencial.