2. ESPECIFICACIÓN FORMAL DE SISTEMAS CONCURRENTES

Dado un programa secuencial, existe un modelo formal para saber que partes de dicho programa pueden ser ejecutadas concurrentemente, es decir, en paralelo y qué partes no. Obviamente no todas las partes de un programa secuencial pueden ser paralelizadas, es decir, no podemos tener un programa secuencial paralelizado al 100 por ciento. Consideremos por ejemplo las siguientes instrucciones de un programa:

x ← x+1;

y ← x+2;

Esta claro que la primera instrucción se tiene que ejecutar antes que la segunda, por tanto, éstas dos instrucciones no pueden ejecutarse concurrentemente (al mismo tiempo). Ahora consideremos las siguientes instrucciones:

x ← 1;

y ← 2;

z ← 3;

Queda claro que el orden en el que se ejecuten las instrucciones anteriores es irrelevante para el resultado final de sus ejecuciones, es decir, las tres instrucciones pueden paralelizarse sin ningún problema. Si dispusiéramos de tres procesadores, podríamos ejecutar cada una de las instrucciones en uno de ellos consiguiendo incrementar la velocidad del sistema.

Con estos ejemplos hemos utilizado el sentido común para saber qué se podía y que no se podía paralelizar, sin embargo como se mencionó al inicio del presente tema existe un modelo formal para saberlo llamado *"Las Condiciones de Bernstein"* las cuales se utilizan para determinar si dos conjuntos de instrucciones en un programa secuencial, digamos *Si y Sj* se pueden ejecutar concurrentemente. Dichas condiciones deben su nombre a la persona que las ideó: A. J. Bernstein en 1966.

2.1 Condiciones de Bernstein

Para poder determinar si dos conjuntos de instrucciones se pueden ejecutar concurrentemente, se definen en primer lugar los siguientes conjuntos:

$L(S_k)=\{a_1, a_2, \ldots a_n\}$ como el **conjunto de lectura** del conjunto de instrucciones S_k y que esta formado por todas las variables cuyos valores son referenciados (*se leen*) durante la ejecución de las instrucciones en S_k.

$E(S_k)=\{b_1, b_2, \ldots b_n\}$ como el **conjunto de escritura** del conjunto de instrucciones S_k y que está formado por todas las variables cuyos valores son actualizados (*se escriben*) durante la ejecución de las instrucciones en S_k.

Para que dos conjuntos de instrucciones S_i y S_j se puedan ejecutar concurrentemente, se tiene que cumplir lo siguiente:

a) $L(Si) \cap E(Sj) = \emptyset$
b) $E(Si) \cap L(Sj) = \emptyset$
c) $E(Si) \cap E(Sj) = \emptyset$

Ejemplo: Supongamos que tenemos lo siguiente:

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

Utilizando las condiciones de Bernstein mostraremos qué instrucciones pueden ejecutarse concurrentemente y cuáles no. Para ello en primer lugar calculamos los conjuntos de lectura y escritura de cada una de las instrucciones:

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

Ahora aplicamos las condiciones de Bernstein a cada par de instrucciones:

Entre S1 y S2:

$L(S1) \cap E(S2) = \emptyset$

$E(S1) \cap L(S2) = \emptyset$

$E(S1) \cap E(S2) = \emptyset$

Entre S1 y S3:

$L(S1) \cap E(S3) = \emptyset$

$\mathbf{E(S1) \cap L(S3) = a}$

$E(S1) \cap E(S3) = \emptyset$

Entre S1 y S4:

$L(S1) \cap E(S4) = \emptyset$

$E(S1) \cap L(S4) = \emptyset$

$E(S1) \cap E(S4) = \emptyset$

Entre S2 y S3:

$L(S2) \cap E(S3) = \emptyset$

$\mathbf{E(S2) \cap L(S3) = b}$

$E(S2) \cap E(S3) = \emptyset$

Entre S2 y S4:

$L(S2) \cap E(S4) = \emptyset$

$E(S2) \cap L(S4) = \emptyset$

$E(S2) \cap E(S4) = \emptyset$

Entre S3 y S4:

$L(S3) \cap E(S4) = \emptyset$

$\mathbf{E(S3) \cap L(S4) = c}$

$E(S3) \cap E(S4) = \emptyset$

De todo esto se deduce la siguiente matriz en la que se muestran los pares de instrucciones que pueden ejecutarse de forma concurrente:

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

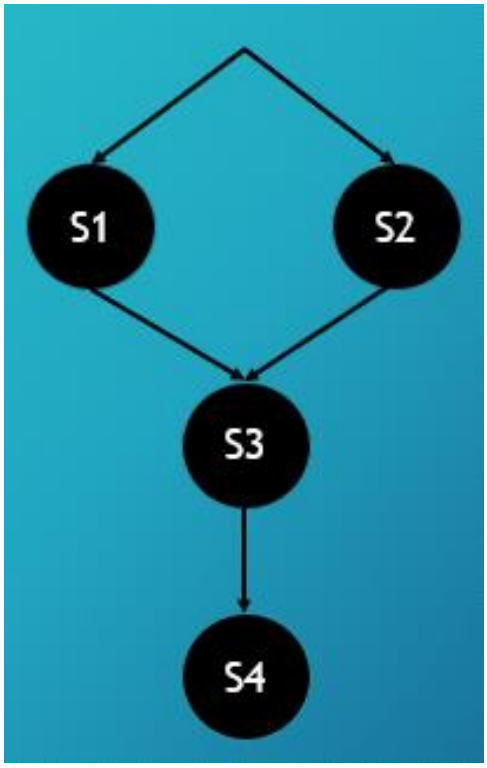
La matriz anterior es llamada Matriz de Precedencia. Es una matríz cuadrada cuyo tamaño es el número de instrucciones que se están comparando. Es una matríz booleana donde un 1 indica que el par de instrucciones que se comparan si pueden ejecutarse concurrentemente y un 0 indica que dicho par no se puede ejecutar en paralelo.

Una vez que se sabe qué se puede y que no se puede ejecutar concurrentemente, se hace necesario algún tipo de notación para especificar qué partes de un programa pueden ejecutarse en paralelo y qué partes no.

2.2. Especificación de la ejecución concurrente usando Grafos de Precedencia

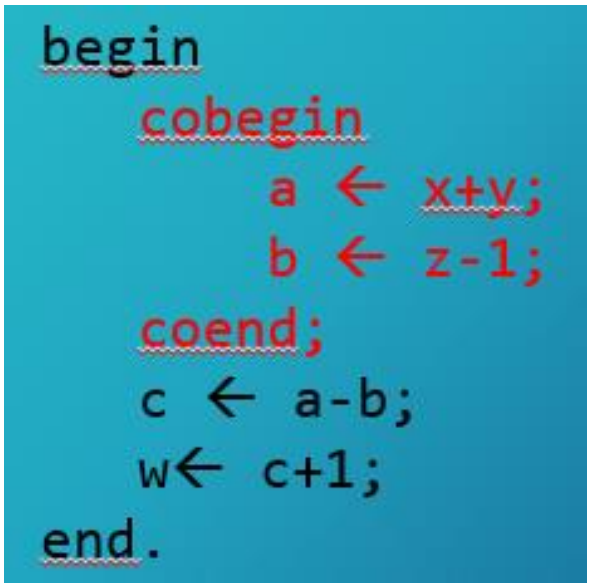
Un grafo de precedencia es una notación gráfica. El grafo es dirigido y acíclico. Cada nodo representa una parte (conjunto de instrucciones) del sistema. Una flecha desde un nodo A hasta un nodo B representa que B sólo puede ejecutarse cuando A haya finalizado (ejecución secuencial). Si aparecen dos nodos en paralelo, querrá decir que éstos se pueden ejecutar concurrentemente (al mismo tiempo).

Para el ejemplo anterior, el grafo de precedencia será el de la figura siguiente:



2.3. Especificación de la ejecución concurrente usando el par COBEGIN-COEND

Todas aquellas instrucciones que pueden ejecutarse concurrentemente se escriben dentro del par de sentencias COBEGIN-COEND. El ejemplo anterior quedaría de la siguiente manera:



Las instrucciones dentro del par COBEGIN-COEND pueden ejecutarse en paralelo, es decir, en cualquier orden, mientras que el resto se ejecuta de manera secuencial.