Programación Concurrente

Condiciones de Bernstein: Sirve para poder determinar si dos conjuntos de instrucciones S_i y S_j se pueden ejecutar concurrentemente

Sea S_k un conjunto de instrucciones

Se definen los siguientes conjuntos

$L(S_k) = \{a_1, a_2, \dots a_n\}$ conjunto de lectura, variables leídas en el conjunto de instrucciones

$E(S_k) = \{b_1, b_2, \dots b_n\}$ conjunto de escritura variables actualizadas en el conjunto de instrucciones

Para que dos conjuntos de instrucciones se puedan ejecutar concurrentemente se tiene que cumplir que:

$$L(S_i) \cap E(S_j) = \emptyset$$
$$E(S_i) \cap L(S_j) = \emptyset$$
$$E(S_i) \cap E(S_j) = \emptyset$$

Supongamos el siguiente conjunto de instrucciones

$S_1 \rightarrow a = x + y$

$S_2 \rightarrow b = z - 1$

$S_3 \rightarrow c = a - b$

$S_4 \mapsto w = c + 1$

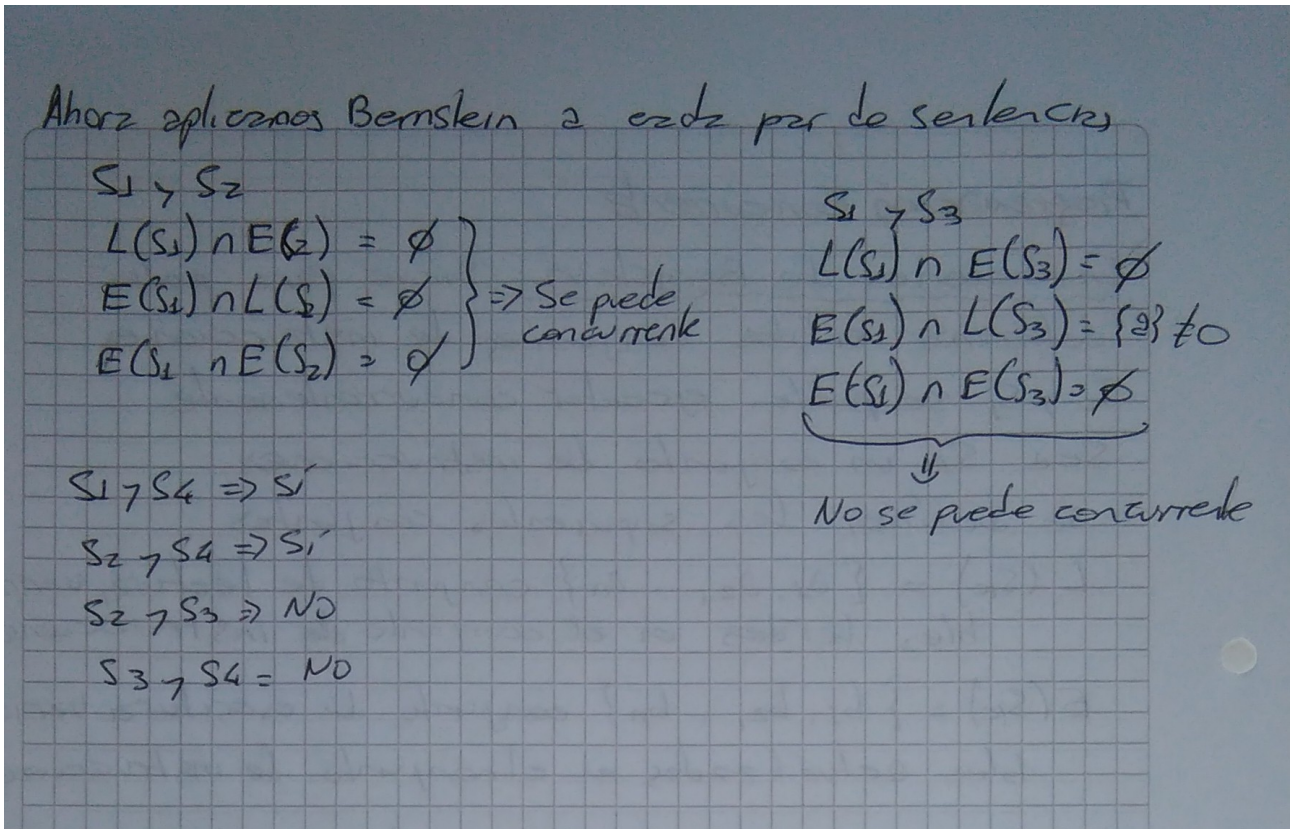
Calculamos conjuntos de lectura y escritura

$L(S_1) = \{x, y\}$ $\quad E(S_1) = \{a\}$

$L(S_2) = \{z\}$ $\quad E(S_2) = \{b\}$

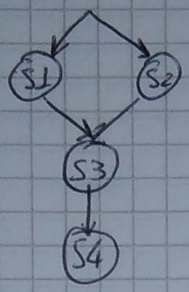
$L(S_3) = \{a, b\}$ $\quad E(S_3) = \{c\}$

$L(S_4) = \{c\}$ $\quad E(S_4) = \{w\}$


Ahora aplicamos Bernstein a cada par de sentencias
S1 ; S2
L(S1) ∩ E(S2) = ∅
E(S1) ∩ L(S2) = ∅
E(S1) ∩ E(S2) = ∅
⇒ Se puede concurrente
S1 ; S3
L(S1) ∩ E(S3) = ∅
E(S1) ∩ L(S3) = {a} ≠ ∅
E(S1) ∩ E(S3) = ∅
⇓
No se puede concurrente
S1 ; S4 ⇒ SÍ
S2 ; S4 ⇒ SÍ
S2 ; S3 ⇒ NO
S3 ; S4 = NO

Programación Concurrente

Grafos de Precedencia: Es una notación gráfica cada nodo representará un conjunto de instrucciones



Sentencias COBEGIN/COEND: Las acciones que se pueden ejecutar concurrentemente van dentro de cobegin/coend

```
begin
    cobegin
        a = x + y
        b = z - 1
    coend
    c = a - b
    w = c + 1
end
```