Lección 3: Concurrencia y sincronización

- Composición concurrente de acciones
- Acciones atómicas y sincronización
- Atomicidad de una expresión
- Atomicidad de una acción de asignación
- Sincronización de acciones

Composición concurrente de acciones

- **Ejecución secuencial**: un flujo de control
- **Ejecución concurrente**: varios flujos de control, uno por cada acción concurrente o proceso
- Notación:

comp. concurrente

```
PCo A1 || A2 ||...|| An  FCo
```

- Ejemplo:

```
Vars x:Ent :=0
     y:Ent :=0
PCo  <x:=x+1>  ||  <y:=y+1> FCo
z:=x+y
```

Composición concurrente de acciones

- Declaración implícita de procesos:

```
Consts n=... {lo que sea}
Tipos matriz=
      vector(1..n,1..n) de Real
Vars a,b,c:matriz

PCo i:=1..n,j:=1..n
  Vars  sum: Real:=0

  Pt  k:=1..n
    sum:=sum+a(i,k)*b(k,j)
  FPt
  c(i,j):=sum
FCo
```

- *n^2 procesos que se ejecutan en paralelo*
- *i, j: constantes locales de cada acción*
- *sum, k: variables locales de cada acción*

Composición concurrente de acciones

- *Mayor es un proceso*
- *Suma es un vector de "n" procesos Suma(i)*
- *La notación anterior equivale a:*

```
PCo
    Mayor
||  PCo  i:=1..n
       Suma(i)
    FCo
FCo
```

```
Consts n=... {lo que sea}
Tipos vectN=
        vector(1..n) de Ent
Vars a,b:vectN
Mayor::
   Vars max:Ent
   max:=a(1)
   Pt j:=2..n tq max<a(j)
      max := a(j)
   FPt
Suma(i:1..n)::
   b(i):=a(1)
   Pt j:=2..i
     b(i):=b(i)+a(j)
   FPt
```

Acciones atómicas y sincronización

- **Razonar** sobre un programa concurrente necesita tener en cuenta los problemas de interferencia
  - ya sea para asegurar propiedades de vivacidad o de seguridad
- Formas de enfrentarse a la interferencia
  - estudiar los casos en que no la hay
  - imponerla en los casos en que puede haberla
- ¿Cuándo se puede considerar una acción como atómica?
  - es una **acción atómica de grano fino**
    - directamente implementada sobre el hardware
  - se dan determinadas condiciones suficientes

Acciones atómicas y sincronización

- ¿Cómo imponer atomicidad?
  - sincronizando
- Formas de sincronizar:
  - formar **acciones de atómicas de grano grueso** mediante la composición de acciones atómicas de grano fino
  - **detener** la ejecución de algunas acciones hasta que se satisfagan determinadas condiciones
- Nuestra "arquitectura":
  - acceso a un escalar es atómico
  - cada proceso tiene su propio conjunto de registros
  - resultados intermedios de una expresión, en memoria local al proceso que la ejecuta

sincronización por ***exclusión mutua***

sincronización por ***condición***

Atomicidad de una asignación

*propiedad "CMU": "**como mucho una**"*

- ¿Cuándo se puede considerar atómica `x := E`
  - "`E`" no referencia variables modificadas por otro proceso
  - se cumple la propiedad CMU

- Una expresión "`E`" cumple la propiedad CMU cuando:

  (C1) no interviene en `E` ninguna variable modificable por otro proceso concurrente

  (C2) en `E` interviene únicamente una variable modificable por otro proceso concurrente y el número de ocurrencias de dicha variable se limita a una

- Una asignación "`x := E`" la cumple cuando:
  - `x` es una variable "local" y `E` cumple la propiedad
  - `x` es compartida (escalar, claro) y `E` no es modificable por otro proceso

Atomicidad de una asignación

- Ejemplo: ¿Cumplen CMU?

```
Vars x,y,z,w:Ent
...
PCo
   x:=y+z
|| w:=10
FCo
```

```
Vars x,y:Ent
...
PCo
   x:=y+1
|| y:=x+1
FCo
```

```
Vars x,y:Ent
...
PCo
   x:=y+1
|| y:=y+1
FCo
```

Sincronización

- ¿Qué pasa cuando necesitamos una ejecución atómica, pero la instrucción no lo es?
- Necesitamos establecer una sincronización "conceptual"
  - no tiene por qué estar implementada por hardware
- Instrucción de sincronización:

```
<await B
     S
>
```

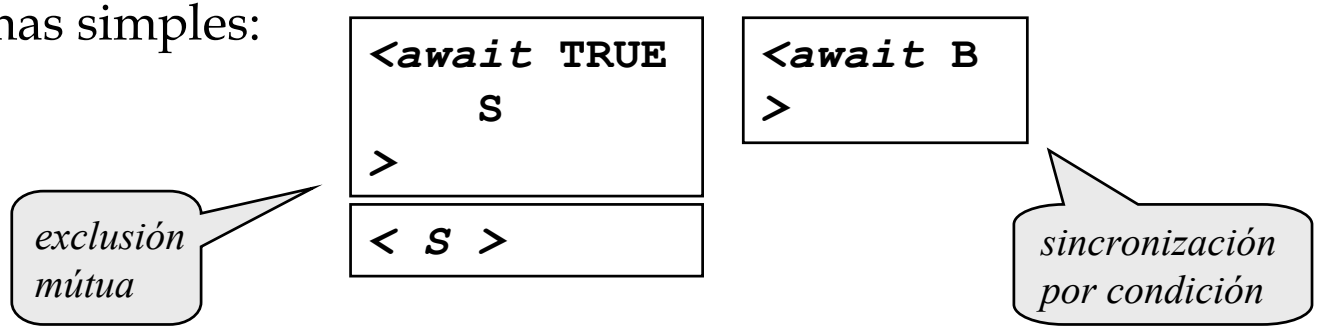
¡Todo se ejecuta atómicamente!

exclusión mutua + sincronización por condición

- Semántica informal:
  - B es una guarda booleana
  - la ejecución se detiene hasta que se cumpla B
  - después, se ejecuta S (hay que asegurar que S termine)

Sincronización

- Ejemplo: `<await (s>0)  s:=s-1 >`
- Significado:
  - el proceso se detiene hasta que **s>0** (puede que no ocurra nunca)
  - entonces se ejecuta la asignación
- Es **LA** instrucción de sincronización
  - muy potente y general
  - muy costosa de implementar
- Formas simples:



Sincronización

- Ejemplos:
  - `< x:=x+1; y:=y+1 >` se ejecutan atómicamente
  - `<await (buf>0) >` esperar a que **`buf>0`**

- Si resulta que B cumple la CMU, entonces son equivalentes

```
<await B
>
```

```
Mq ¬B
   seguir
FMq
```

¡¡Ya tenemos una implementación!!

Un ejemplo

- *f : Z ---> N* con al menos un cero. Buscamos uno de ellos

```
            Vars encont: Booleano
Posit::                          |  Negat::
  Vars p:Ent := 0                |    Vars n:Ent := 1
                                 |
  encont := FALSE                |    encont := FALSE
  Mq ¬encont                     |    Mq ¬encont
    p := p+1                     |      n := n-1
    encont := (f(p) = 0)         |      encont := (f(n) = 0)
  FMq                            |    FMq
```

Un ejemplo

```
Vars encont: Booleano := FALSE
```

```
Posit::
  Vars p:Ent := 0

  Mq ¬encont
    p := p+1
    encont := (f(p) = 0)
  FMq
```

```
Negat::
  Vars n:Ent := 1

  Mq ¬encont
    n := n-1
    encont := (f(n) = 0)
  FMq
```

Un ejemplo

```
            Vars encont: Booleano := FALSE
Posit::                              Negat::
  Vars p:Ent := 0                      Vars n:Ent := 1

  Mq ¬encont                           Mq ¬encont
    p := p+1                             n := n-1
    Si f(p)=0 Ent                        Si f(n)=0 Ent
       encont := TRUE                       encont := TRUE
    FSi                                  FSi
  FMq                                  FMq
```

Un ejemplo

"cableamos" el scheduler

```
Vars encont: Booleano := FALSE
     turno: Ent := 1

Posit::                          Negat::
  Vars p:Ent := 0                  Vars n:Ent := 1

  Mq ¬encont                       Mq ¬encont
    <await turno=1                   <await turno=2
      turno := 2                       turno := 1
    >                                >
    p := p+1                         n := n-1
    Si f(p)=0 Ent                    Si f(n)=0 Ent
       encont := TRUE                   encont := TRUE
    FSi                              FSi
  FMq                              FMq
```

Un ejemplo

```
            Vars encont: Booleano := FALSE
                 turno: Ent := 1
Posit::                                  Negat::
  Vars p:Ent := 0                          Vars n:Ent := 1

  Mq ¬encont                               Mq ¬encont
    <await turno=1                           <await turno=2
      turno := 2                               turno := 1
    >                                        >
    p := p+1                                 n := n-1
    Si f(p)=0 Ent                            Si f(n)=0 Ent
       encont := TRUE                           encont := TRUE
    FSi                                      FSi
  FMq                                      FMq
  turno := 2                               turno := 1
```

Un ejemplo

```
            Vars encont: Booleano := FALSE
                 turno: Ent := 1
Posit::                                  Negat::
  Vars p:Ent := 0                          Vars n:Ent := 1

  Mq ¬encont                               Mq ¬encont
    <await turno=1>                          <await turno=2>
    turno := 2                               turno := 1
    p := p+1                                 n := n-1
    Si f(p)=0 Ent                            Si f(n)=0 Ent
       encont := TRUE                           encont := TRUE
    FSi                                      FSi
  FMq                                      FMq
  turno := 2                               turno := 1
```