

Problema del puente

CÓDIGO

Monitor

```

Var mutex: Lock()
    patata, turno,
    carsNorth, carsSouth,
    pedestrian, carsNorth_waiting,
    carsSouth_waiting, Pedestrian_waiting : integrer.
    self.no_NorthP, no_SouthP,
    no_Cars : Condition

procedure wants_enter_car(direction : int)
begin
    mutex.acquire()
    if direction == 0:
        carsNorth_waiting += 1
        no_SouthP.wait_for(are_no_SouthPed) → I ✓
        self.carsNorth_waiting -= 1
        self.carsNorth += 1
    else:
        carsSouth_waiting += 1
        no_NorthP.wait_for(self.are_no_NorthPed) → I ✓
        carsSouth_waiting -= 1
        carsSouth += 1
    mutex.release()
end

procedure leaves_car(direction: int)
begin
    mutex.acquire()
    if direction==0 :
        carsNorth -= 1
        self.turno = 1
        if carsNorth ==0:
            controlturno()
            no_NorthP.notify_all()
            no_Cars.notify_all()
        else:
            carsSouth -= 1
            turno = 2
            if carsSouth==0:
                controlturno()
                no_Cars.notify_all()
                no_SouthP.notify_all()
            self.mutex.release()
    procedure wants_enter_pedestrian()
        mutex.acquire()
        Pedestrian_waiting += 1
        no_Cars.wait_for(are_no_cars) → I ✓
        Pedestrian_waiting -= 1
        pedestrian += 1
        mutex.release()

    procedure leaves_pedestrian()
        mutex.acquire()
        pedestrian -= 1
        turno = 0
        if pedestrian==0: → I ✓
            controlturno()
            no_NorthP.notify_all()
            no_SouthP.notify_all()
        mutex.release()

```

```

procedure controlturno()
if turno== 0 and carsNorth_waiting==0 and (carsSouth_waiting>0 and Pedestrian_waiting>0):
    self.turno.value = 1
elif turno== 1 and carsSouth_waiting==0 and (carsNorth_waiting>0 and Pedestrian_waiting>0):
    self.turno.value = 2
elif turno.value == 2 and Pedestrian_waiting==0 and (carsNorth_waiting>0 and carsSouth_waiting>0):
    turno= 0

fun are_no_cars() -> b : Bool
b = {(carsNorth==0 and carsSouth==0) and
      (turno==2 or (carsNorth_waiting ==0 and carsSouth_waiting ==0))}

fun are_no_SouthPed() -> b : Bool
b = {(carsSouth == 0 and pedestrian ==0) and
      (turno==0 or (carsSouth_waiting ==0 and Pedestrian_waiting ==0))}

fun are_no_NorthPed() -> b : Bool
b = {(carsNorth == 0 and pedestrian ==0) and
      (turno ==1 or (carsNorth_waiting ==0 and Pedestrian_waiting ==0))}

Begin(* monitor *)
    turno := 0
    carsNorth :=0
    carsSouth := 0
    pedestrian := 0

    carsNorth_waiting := 0
    carsSouth_waiting := 0
    Pedestrian_waiting := 0
end:

procedure car(direction: int, monitor: Monitor):
    monitor.wants_enter_car(direction)
    monitor.leaves_car(direction)
procedure pedestrian(monitor: Monitor)
    monitor.wants_enter_pedestrian()
    monitor.leaves_pedestrian()

```

INVARIANTE

Se cumple:

$$(1) \text{ CarsNorth} > 0 \Rightarrow (\text{CarsSouth} = 0 \wedge \text{pedestrian} = 0)$$

- Supongamos que se cumple (1) y que a lo largo del desarrollo se incumple por ser $\text{CarsSouth} > 0$. Si tenemos que $\text{CarsNorth} > 0$ esto supone que hay coches en dirección Norte cruzando el puente, es decir, hay procesos con dirección 0 que están ejecutando :

```
procedure car(direction: int, monitor: Monitor):
    monitor.wants_enter_car(direction)
    monitor.leaves_car(direction)
```

Ahora bien para que $\text{CarsSouth} > 0$ necesariamente se debe hacer un notify (1) y (2). Luego necesariamente Cars con dirección 0 deben haber hecho un notify (3)

```
procedure wants_enter_car(direction: int)
begin
    mutex.acquire()
    if direction == 0:
        carsNorth_waiting += 1
        no_SouthP.wait_for(are_no_SouthPed)
        self.carsNorth_waiting -= 1
        self.carsNorth += 1
    else:
        carsSouth_waiting += 1
        no_NorthP.wait_for(self.are_no_NorthPed) (1)
        carsSouth_waiting -= 1
        carsSouth += 1 (2)
    mutex.release()
end
```

pero únicamente se lleva a cabo

Si $\text{CarsNorth} = 0$. Lo que contradice el incumplimiento de (1).

Si suponemos $\text{pedestrian} > 0$ llegamos a la misma contradicción.

```
procedure leaves_car(direction: int)
begin
    mutex.acquire()
    if direction == 0:
        carsNorth -= 1
        self.turno = 1
    if carsNorth == 0:
        no_NorthP.notify_all()
    (3) { no_Cars.notify_all()
        :
    }
```

$$(2) \text{ CarsSouth} > 0 \Rightarrow (\text{CarsNorth} = 0 \wedge \text{pedestrian} = 0)$$

La demostración es análoga a la de (1).

$$(3) \text{ Pedestrian} > 0 \Rightarrow (\text{carsNorth} = 0 \wedge \text{carsSouth} = 0)$$

Demostración análoga a la de (1) por (4).

```
procedure leaves_pedestrian()
begin
    mutex.acquire()
    pedestrian -= 1
    turno = 0
    if pedestrian == 0:
        no_NorthP.notify_all()
    (4) { no_SouthP.notify_all()
        :
    }
    mutex.release()
```

Luego el invariante señala:

$$J = \{ \text{if } \text{carsNorth} > 0 \text{ then } (\text{carsSouth} = 0 \text{ and } \text{pedestrian} = 0) \}$$

$$\text{And if } \text{carsSouth} > 0 \text{ then } (\text{carsNorth} = 0 \text{ and } \text{pedestrian} = 0)$$

$$\text{And if } \text{pedestrian} > 0 \text{ then } (\text{carsNorth} = 0 \text{ and } \text{carsSouth} = 0) \}$$

El invariante nos asegura que el puente es seguro.

Ausencia de deadlock

Para ver la ausencia de deadlock es esencial tener en cuenta que los procesos terminan, es decir todo proceso que entra en el puente sale.

Supongamos ahora que nos encontramos en un deadlock, todos los procesos están esperando una señal para poder llevar a cabo sus funciones. Como todo proceso que entra en el puente sale entonces los procesos que tengan acceso en ese momento al "puente" (carsNorth, carsSouth o pedestrian) harán un notify. Estudiamos los distintos casos:

- En el momento del notify no había procesos de los otros grupos esperando.

Supongamos que los procesos que están dentro pertenecen al grupo G_1 y el resto son G_2 y G_3 . Entonces si cuando los procesos de G_1 han hecho el notify el turno le pertenece a G_2 pero como no hay procesos a la espera pueden pasar los procesos de G_1 si los hubiera. Si no hubiera ningún proceso que quisiese entrar, en el momento en el que uno quiera entrar tendrá acceso independientemente del turno. Si quisieran entrar procesos de distintos grupos el turno será de alguno de estos.

- El turno es de un grupo que no tiene procesos esperando.

Supongamos que G_1 ha terminado y ahora es el turno del grupo G_2 y no hay procesos esperando entonces si solo hay procesos a la espera en el grupo G_3 como al terminar G_1 hace un notify tanto a G_2 como a G_3 los procesos del grupo G_3 tendrán acceso.

Si suponemos que hay procesos a la espera tanto del G_1 como del G_3 y el turno es de G_2 . Entonces G_1 le ha cedido el turno a G_2 y antes de hacer el notify ha llamado a la función `Controlturno()` que habrá dado el turno a G_3 .



`procedure controlturno()`

```
if turno== 0 and carsNorth_waiting==0 and (carsSouth_waiting>0 and Pedestrian_waiting>0):
    self.turno.value = 1
elif turno== 1 and carsSouth_waiting==0 and (carsNorth_waiting>0 and Pedestrian_waiting>0):
    self.turno.value = 2
elif turno.value == 2 and Pedestrian_waiting==0 and (carsNorth_waiting>0 and carsSouth_waiting>0):
    turno= 0
```

Inanición

Para que hubiera inanición en este problema debería de ocurrir que alguno o algunos de los grupos no tuvieren acceso al puente porque un grupo de procesos no le libera.

Supongamos que los procesos del grupo 1 G_1 no liberan el puente. Sabemos que cada proceso que entra en el puente sale, luego si bloquean el puente será por una entrada masiva de procesos de este grupo ¿Es esto posible?

Supongamos P_1, P_2, \dots los procesos del grupo G_1 ordenados en el orden de entrada al puente. Entonces P_1 entra, cruza el puente y sale. Cuando P_1 sale el turno cambia y ahora es del grupo 2. Luego, una vez que ha salido P_1 , si P_k intenta entrar solo podrá hacerlo si no hay procesos de los otros grupos esperando. Luego P_k se mantiene a la espera y una vez salen los $k-1$ procesos tendrá acceso el resto de los grupos.

Lo mismo ocurriría si suponemos que cualquiera de los otros dos grupos bloquea el puente.

Obs:

Nos referimos a grupos de procesos por simplicidad. Estos representan coches en dirección norte, coches en dirección sur y peatones.

