Pablo Sierra Erice

El problema consiste en diseñar una solución sencilla para el paso de vehículos y peatones por el puente de ambito donde:

- No se permite el paso de vehículos en ambos sentidos
- Los vehículos y los peatones no pueden compartir el puente
- Pueden pasar peatones en sentido contrario
- El puente va del Norte al Sur y es de doble sentido

Vamos a crear el monitor (Utilizado clases en Python).

integer array [0..2] numbe ← [0..0] → Número de coches del norte, sur y peatones cruzando
integer array [0..2] waiting ← [0..0] → Número de coches del norte, sur y peatones esperando
Condition array [0..2] Oktogoss

Creamos tres listas donde van los coches del norte (índice 0), los coches del sur (índice 1) y los peatones (índice 2) especificados arriba.

Ahora creamos funciones auxiliares:

Oktogoss [0] es number[2]=0 y number[1]=0
Oktogoss [1] es number[2]=0 y number[0]=0
Oktogoss [2] es number[0]=0 y number[1]=0

```
def wants_enter_car(direction)
    wait(mutex)
    if direction = 0: (Es decir, dirección North)
        waiting[0] ← waiters[0] + 1
        wait( Oktogoss[0])
        waiting[0] ← waiting[0] - 1
        turn ← 0
        number[0] ← number[0] + 1
    if direction = 1: (Es decir, dirección South)
        waiting[1] ← waiting[1] + 1
        wait( Oktogoss[1])
        waiting[1] ← waiting[1] - 1
        turn ← 1
        number[1] ← number[1] + 1
    signal(mutex)

def leaves_car(direction)
    mutex.acquire()
    number[direction] ← number[direction] - 1
    if direction = 0: (North)
```

```
        if waiting[1] ≠ 0:
            turn ← 1
        if waiting[2] ≠ 0:
            turn ← 2
        if numbe[0] = 0:
            signal C (oktopass[1])
            signal C (oktopass[2])
    if direction = 1. (Sud)
        if waiting[2] ≠ 0:
            turn ← 2
        if waiting[0] ≠ 0:
            turn ← 0
        if numbe[1] = 0:
            signal C [oktopass[2])
            signal C (oktopass[0])
    signal (mutex)

operation wants_ate_pedestrian:
    wait (mutex)
    waiting[0] ← waiting[0] + 1
    wait C (oktopass[2])
    waiting[0] ← waiting[0] - 1
    turn ← 2
    numbe[0] ← number[0] + 1
    signal (mutex)

operation leaves_pedestrian:
    wait (mutex)
    number[0] ← numbe[0] - 1
    if waiting[1] ≠ 0:
        turn ← 0
    if waiting[2] ≠ 0:
        turn ← 1
```

```
if number[0] = 0:
    if waiting[1] ≠ 0:
        signal(OKtopass[1])
    else:
        signal(OKtopass[2])
signal(mutex)
```

También tenemos que los procesos son:

ca(direction)

```
loop forever
    monitor.wats_after_car(direction)
    monitor.leaves_car(direction)
```

pedestrian()

```
loop forever
    monitor.wats_after_pedestrian()
    monitor.leaves_pedestrian()
```

¿Cuál es el invariante de la solución?

number[i] ≥ 0

waiting[i] ≥ 0

¬ empty(OKtopass[0]) → (number[1]=0 and number[2]=0)
¬ empty(OKtopass[1]) → (number[0]=0 and number[2]=0)
¬ empty(OKtopass[2]) → (number[0]=0 and number[1]=0)

¿Es el puente seguro?

Cumple la segunda parte del invariante:

- Si pasan coches el norte ⇒ no pasan peatones ni coches del sur
- Si pasan coches al sur ⇒ no pasan peatones ni coches al norte
- Si pasan peatones ⇒ no pasan coches

Se puede ver el error que utilizando la solución wait(c) que garantiza esperar a que un peatón pase si no hay coches y que un coche pase si no hay peatones o coches de la otra dirección. Igualmente signal(c) avisa de que un coche o peatón puede pasar.

- La solución tiene ausencia de Deadlock?

Gracias a turn se especifica quién va a pasar, que le q-e si varios procesos (nodos al norte, al sur y peatones) intentan pasar, solo puede pasar el q-e tiene el turno y pasa si sigue.

¿Ausencia de Inanición?

Ningún proceso se q-edó infinitamente atascado pues los turnos se cambia todo el rato.