Laura Henero

Monitor P:

Condition semaphore A
- in_norte int=0
- in_sur int=0
- in_peatones int=0

en mi código vienen representadas en un diccionario.
in → los que estén dentro del puente
cont → contador de los que ya han pasado.

no me interesa para la demostración:
- cont_norte int=0
- cont_sur int=0
- cont_peatones int=0

INV ≡ in_norte ≥0 ∧ int_sur ≥0 ∧ in_peatones ≥0 ∧
∧ cont_norte ≥0 ∧ cont_sur ≥0 ∧ cont_peatones ≥0 ∧
∧ [in_norte >0 => in_sur ==0 ∧ in_peatones ==0] ∧
∧ [in_sur >0 => in_norte ==0 ∧ in_peatones ==0] ∧
∧ [in_peatones >0 => in_norte ==0 ∧ in_sur ==0]

Operadores:

wants2enterP:

```
semaphoreA.wait
(in_norte==0 ∧
 in_sur==0)
 in_peatones =
  in_peatones+1
```

wants2enterN:

```
semaphoreA.wait
 (in_sur==0 ∧
  in_peatones==0)
in_norte =
  in_norte+1
```

wants2enterS:

```
semaphoreA.wait
(in_norte==0 ∧
 in_peatones==0)
in_sur =
  in_sur+1
```

* Estas tres funciones están programadas en una sola distinguiendo los casos mediante condicionales.

leaveTunelP:

```
in_peatones =
  in_peatones-1
if (in_peatones==0):
  semaphoreA.notify_all()
```

leaveTunelN:

```
in_norte =
  in_norte - 1
if(in_norte==0):
  semaphoreA.notify_all()
```

leaveTunelS:

```
in_sur =
  in_sur -1
if (in_sur==0):
  semaphoreA.notify_all()
```

Procesos:

Peatón:

```
P.wants2enterP()
passing
P.leaveTunelP()
```

Coche Norte:

```
P.wants2enterN()
passing
P.leaveTunelN()
```

Coche Sur:

```
P.wants2enterS()
passing
P.leaveTunelS()
```

⊗ Puesto que las condiciones del wait son excluyentes, solo nos hace falta un semáforo.

① Veamos que nuestro puente es seguro, i.e., no hay coches de distinto tipo a la vez ni peatones.

Si hubiera coches en sentido norte quiere decir que dentro del puente no hay peatones ni coches en sentido sur

[in_norte > 0 => in_sur == 0 ∧ in_peatones == 0 .]

Al haber ejecutado el Proceso Coche Norte, y hacer el P.wants2enterN(), este espera a la condición de que no haya coches en sentido sur ni peatones dentro del puente (lo hace mediante "semaphoreA.wait(in_sur == 0 ∧ in_peatones == 0)").

Más tarde, cuando se ejecute leaveTunnelN se garantizará que se notifique a los procesos sobre cuando in_norte == 0, i.e, no haya coches en sentido norte dentro del puente.

Los otros casos, in_peatones > 0 y in_sur > 0, son análogos.

② Veamos la ausencia de deadlocks.

Si nuestros procesos estuviesen bloqueados, estarían esperando en el wait del wants2enter. Si esto pasara, se tendría que cumplir que in_norte ≠ 0 ∧ in_sur ≠ 0 ∧ in_peatón ≠ 0, pero esto no puede ocurrir por ① (!!)

③ Veamos ausencia de inanición

Supongamos que Coche Norte tiene inanición. Eso implicaría que esté bloqueado en el wait del wants2enterN. Esto se puede deber a dos motivos:

1) semaphoreA.notify_all() no se ejecuta
2) (in_sur == 0 ∧ in_peatones == 0) nunca es cierto.

- Si in_sur > 0 => se ejecutará leaveTunnelS => in_sur disminuirá, en algún momento se notificará semaphoreA. y la condición del wait será cierta.
- Si in_peatones > 0 es análogo.

Para Coche Sur y Peatón es análogo.