Sinhué García G.1 PRPA - Práctica 2

Solución sencilla que cumple seguridad

Sea $I = \{0, 1, 2\}$

Monitor:

Inv: { inside ≥ 0;
$\forall i \in I$, who_is_inside[i] $\in \{0,1\}$;
$0 \leq \sum_{i \in I}$ who_is_inside[i] ≤ 1 }

```
int inside = 0
who_is_inside = [0]*3
VC free_bridge

empty_bridge (direction:int) -> bool:
    return (0 == who_is_inside [direction ⊕ 1]) and
           (0 == who_is_inside [direction ⊖ 1])

wants_enter_car (dir :int):
    mutex.wait()
    free_bridge.wait_for(empty_bridge (dir))
    who_is_inside [dir] = 1
    inside += 1
    mutex.signal()
leaves_car (dir :int):
    mutex.wait()
    inside -= 1
    if inside == 0:
        who_is_inside [dir] = 0
        free_bridge.notify_all()
    mutex.signal()
```

Análogo para peatones con dir = 2.

El puente es seguro, ya que cualquier vehículo que quiera entrar, va a tener que esperar a que los otros dos no estén en el puente. A hacer esto, una vez finaliza wants_enter_car tenemos {Inv ∧ who_is_inside[dir]=1 ∧ inside > 0}
Porque, ninguna otra dirección va a poder entrar al puente, hasta que ya no quede ningún vehículo dentro (inside == 0) y who_is_inside[dir] cambie a 0.

Problema de inanición: Si tuviéramos un sentido por el que pudieran pasar un nº no finito de vehículos, este podría hacerse con el control del puente, ya que (inside == 0) no tendría porque darse => Los otros dos esperan indefinidamente.

Por lo que vamos a modificar el monitor para que tenga este problema en cuenta. Añadimos una variable binaria que controle la espera:

$$Inv := \{ inside \geq 0 \;;\; \forall i \in I \; who\_is\_inside[i] \in \{0,1\} \wedge who\_is\_waiting[i] \in \{0,1\} \;;\; 0 \leq \sum_{i \in I} who\_is\_inside[i] \leq 1 \;;\; 0 \leq \sum_{i \in I} who\_is\_waiting[i] \leq 1 \}$$

Monitor:

... (lo que ya teníamos)

who_is_waiting = [0]*3

VC waiting

...

is_anyone_waiting (dir:int) -> bool:

return ((0 == who_is_waiting [dir ⊕ 1]) and (0 == who_is_waiting [dir ⊖ 1]))

wants_enter (dir:int):

mutex.wait()

waiting.wait_for (is_anyone_waiting (dir))

who_is_waiting [dir] = 1

free_bridge.wait_for (empty_bridge (dir))

who_is_inside [dir] = 1

inside += 1

who_is_waiting [dir] = 0

waiting.notify_all ()

mutex.signal()

leaves (dir:int):

mutex.wait()

inside -= 1

if (inside == 0):

who_is_inside[dir] = 0

free_bridge.notify_all()

mutex.signal()

Seguimos conservando la seguridad por el mismo razonamiento.

Vemos que no hay deadlocks:

Para que ocurriera, se tendría que dar que; el vehículo que ya está esperando a entrar al puente, no tuviera nunca acceso al mismo. (Ambos wait_for bloquearían todos procesos). Pero esto nunca puede pasar, ya que una vez pasa a esperar, no va a dejar entrar a ningún coche en la dirección actual.

Por lo que hay un nº finito en el puente, que acabarán saliendo y dejando el puente libre, una vez (inside == 0).

Veamos ahora que ya no puede haber inanición:

Si un proceso se quedara parado en free_bridge, acabará entrando por lo explicado anteriormente (deadlock), por lo que no se puede dar esta situación.

Si un proceso se quedara parado en waiting, hay que ver que siempre va a acabar entrando.

1) Ya hay alguien esperando para entrar al puente, y otra dirección dentro del puente.

Cuando la dirección actual del puente se quede vacía, este hará la llamada free_bridge.notify_all(), que hará que la dirección esperando entre al puente.

Solo hay una dirección esperando por el invariante.

Una vez dicha dirección esté dentro del puente, efectuará la llamada waiting.notify_all() (En este momento ya no espera nadie en su dirección).

Entonces, por la hipótesis de justicia, cualquier proceso esperando acabará entrando. ⇒ Nuestro proceso entra y por lo tanto no hay inanición.

Cualquier otra posibilidad es un caso particular de este, por lo que no tendremos inanición. Ya que siempre se efectuara la llamada waiting.notify_all().