Practica 2 : Problema Puente.

Versión 1:

Mon Im - Puente 1

Ncoches: int = 0 → Nº coches norte en el puente
Scoches: int = 0 → " " sur " "
np: int = 0 → " peatones " "

INV ≡ { Ncoches, Scoches, np ≥ 0
Ncoches ≥ 0 → Scoches = 0 ∧ np = 0
Scoches ≥ 0 → Ncoches = 0 ∧ np = 0
np ≥ 0 → Scoches = 0 ∧ Ncoches = 0 }

puede_pasar_sur: VC = True
puede_pasar_norte: VC = True
puede_pasar_peaton: VC = True.

entra_coche (direccion)
{INV}
if (direccion = SUR):
puede_pasar_sur.wait(Ncoches = 0 ∧ np = 0)
{INV ∧ Ncoches = 0 ∧ np = 0}
Scoches += 1.
if (direccion = NORTE):
puede_pasar_norte.wait(Scoches = 0 ∧ np = 0)
{INV ∧ Scoches = 0 ∧ np = 0}
Ncoches += 1
{INV}

sale_peaton ()
{INV ∧ np > 0}
np -= 1
puede_pasar_sur.notify_all()
puede_pasar_norte.notify_all()
{INV}

entra_peaton ()
{INV}
puede_pasar_peaton.wait
(Ncoches = 0 ∧ Scoches = 0)
{INV ∧ Ncoches = 0 ∧ Scoches = 0}
np += 1
{INV}.

sale_coche (direccion):
if (direccion = SUR)
{INV ∧ Scoches > 0}
Scoches -= 1
puede_pasar_norte.notify_all()
puede_pasar_peaton.notify_all()
if (direccion = NORTE)
{INV ∧ Ncoches > 0}
Ncoches -= 1
puede_pasar_sur.notify_all()
puede_pasar_peaton.notify_all()

En esta primera versión se garantiza la seguridad del puente, debido a las variables condición y que se mantiene el invariante, no puede haber más de 2 tipos en el puente.

Sin embargo, se produce inanición, ya que mientras un tipo esté en el puente, sólo podrán entrar de ese mismo tipo hasta que salga, y todos los demás esperando a que no haya ninguno.

Para solucionar esto, podemos añadir turnos de la forma siguiente: Cuando un tipo sale del puente le cede el turno a otro tipo (en un orden circular), que espera a que salgan todos los que hay en el puente del anterior y comienza a entrar.

Sería añadir la variable turno: int t=0 (por empezar por algún). Asignar a cada tipo un turno: peatones → 0, Norte → 1, Sur → 2. Añadir a la variable condición que sea su turno al pasar. Por ejemplo:

puede_pasar_peatones (Nnoches=0 ∧ Snoches=0 ∧ turno=1).

Y al salir, después de hacer np-1, hacemos la instrucción turno=(turno+1) mod 3. Así para todos los tipos.

Con esto solucionamos la inanición, pero surgen varios problemas/cuestiones a mejorar: ① El problema puede sufrir de deadlock en los procesos, ya que se puede ceder el turno a un tipo del cual no van a pasar más y quedarse ahí pillados. ② No es óptimo ceder el turno a un tipo el cuál no tiene ningún coche esperando si en otros sí hay ahí o puede seguir pasando del mismo.

¿Cómo mejorarlo? Podemos llevar la cuenta de los coches que están esperando para entrar de cada tipo. Así, solo cederemos el turno al siguiente si hay algún coche de ese tipo esperando, si no al siguiente, y si tampoco tiene ninguno esperando, el turno se mantiene en el que estaba. Así hasta que vuelva a salir otro coche del puente y vuelva a comprobar si es preciso ceder el turno. Además, si lo implementamos así, en el momento en el que no estén esperando ninguno, solo podrán entrar los últimos que han entrado y no el primero que vuelva a llegar. Para solucionar esto, añadimos a el vector la posibilidad de entrar si no hay ninguno de los otros 2 tipos esperando.

Además cabe destacar que el turno se cambiará cuando un coche de un tipo salga del puente, dándole prioridad a este tipo mientras el primer coche ha entrado y está en el puente, lo que sentido para optimizar los tiempos de espera, y que tampoco esté continuamente cambiando el turno, lo que no tendría mucho sentido, podríamos añadir un delay si queremos hacer la espera más grande hasta cambiar de turno.

Otra opción sería ceder el turno si y solo si hay alguno de los tipos que tenga más esperando que el que acaba de salir. Esta opción es óptima para casos generales, pero podría provocar inanición en otros. Imaginemos que hay unos esperando de un tipo y se han agolpado un millón en el otro lado del puente, por ejemplo M. Este que estaba esperando tendría que esperar a los otros M para pasar.

Veamos cómo sería el esquema de la versión anterior.

Practica 2 : Problema puente

Version 2:

Monitor_Puente.

```
Ncoches : int = 0      * N: coches norte en el puente
Scoches : int = 0      * V: coches sur en el puente
np : int = 0           * N: peatones puente.

Ncoches_waiting : int = 0 }
Scoches_waiting : int = 0 } los que estan esperando a pasar.
Peatones_waiting : int = 0 }

puede_pasar_sentido : VC = True
puede_pasar_norte : VC = True
puede_pasar_peaton : VC = True

turno : int = 0   (* 0 peatones, 1 norte, 2 sur.)
```

INV ≡

- Ncoches, Scoches, np ≥ 0
- Ncoches_waiting, Scoches_waiting, Peatones_waiting ≥ 0
- Ncoches > 0 → Scoches = 0 ∧ np = 0
- Scoches > 0 → Ncoches = 0 ∧ np = 0
- np > 0 → Scoches = 0 ∧ Ncoches = 0.
- 0 ≤ turno ≤ 2.

```
entra_coche (direccion)
{INV}
if (direccion = SUR):
    Scoches_waiting + 1
    puede_pasar_sentido.wait(*)
      {INV ∧ Ncoches = 0 ∧ np = 0 ∧ ...}
    Scoches_waiting - 1
    Scoches + 1
if (direccion = NORTE):
    Ncoches_waiting + 1
    puede_pasar_norte.wait(**)
    {INV ∧ Scoches = 0 ∧ np = 0 ∧ ...}
    Ncoches_waiting - 1
    Ncoches + 1
{INV}
```

```
entra_peaton()
{INV}
Peatones_waiting + 1
puede_pasar_peaton.wait(***)
{INV ∧ Ncoches = 0 ∧ Scoches = 0 ∧ ...}
Peatones_waiting - 1
np + 1
{INV}
```

{INV}

(*) Ncoches = 0 ∧ np = 0 ∧ (turno = 2 ∨ (Peatones_waiting = 0 ∧ Scoches_waiting = 0))

(**) Scoches = 0 ∧ np = 0 ∧ (turno = 1 ∨ (Peatones_waiting = 0 ∧ Ncoches_waiting = 0))

(***) Scoches = 0 ∧ Ncoches = 0 ∧ (turno = 0 ∨ (Ncoches_waiting = 0 ∧ Scoches_waiting = 0))

sale_coche(direccion):

{INV}

if (direccion = SUR):

{INV ∧ Scoches >0}

Scoches -= 1

if (Nwaiting > 0):

turno = 1

elif (Pwaiting > 0):

turno = 0

* Esto depende de cómo quieras ir combinando la prioridad. Otra opción es darselo al que más tiene esperando.

puede_pasar_norte.notify_all()

puede_pasar_peaton.notify_all()

if (direccion = NORTE):

{INV ∧ Ncoches >0}

Ncoches -= 1

if (Pwaiting > 0):

turno = 0

elif (Swaiting > 0):

turno = 2

puede_pasar_sur.notify_all()

puede_pasar_peaton.notify_all()

{INV}

sale_peaton():

{INV ∧ np > 0}

np -= 1

if (Swaiting > 0):

turno = 2

elif (Nwaiting > 0):

turno = 1

puede_pasar_sur.notify_all()

puede_pasar_norte.notify_all()

{INV}

En esta versión 2, por los desarrollados antes, no se da inanición y se respeta la seguridad del puente. ¿Puede haber bloqueos? La única forma que haya bloqueo es que sea el turno de un tipo el cual no tenga nadie esperando, y haya esperando en alguno de los otros 2 tipos. Esto no puede ocurrir: Al iniciar el proceso, puede entrar cualquiera ya que NorteWaiting = SurWaiting = PuebloWaiting = 0. Supongamos que entra uno del norte, al salir sólo cederá si hay alguno de los demás esperando, si hay alguno esperando, se repite el proceso con otro coche del sur o pueblo (cuando salgan todos los del norte). Si no, entonces SurWaiting = PuebloWaiting = 0 y no se cambia el turno. Si NorteWaiting > 0, o quiere entrar uno del norte, entra sin problema y se repite el proceso, si no, es decir, NorteWaiting = 0, entonces puede entrar cualquiera de los otros 2 que quiera entrar, por la condición que hemos puesto (por ejemplo para sur) de NorteWaiting = 0 ∧ PuebloWaiting = 0

Como cada vez que sale un coche vuelve a comprobar los turnos, estamos suponiendo de que es el último que entra de un tipo, es decir cuando sale, no quedan del norte en el puente. Como vemos no se produce deadlock.

Adjunto en el github esta última versión (práctica 2) y la versión de ceder el turno al que más tenga esperando, introducida brevemente, y mejor para algunos casos. (práctica 2-2).