PRPA

Práctica 2 Monitores

```
Monitor P
    integer turn          <- 0
    integer pedestrian    <- 0
    integer carsN         <- 0
    integer carsS         <- 0
    condition OK_pedestrian
    condition OK_carsN
    condition OK_carsS

operation entra_pedestrian
    if carsN ≠ 0 or carsS ≠ 0 or (¬empty OK_carsN and turn ≠ 0) or (¬empty OK_carsS and turn ≠ 0):
            waitC(OK_pedestrian)
    pedestrian <- pedestrian + 1

operation sale_pedestrian
    pedestrian <- pedestrian - 1
    if ¬ empty (OK_carsN):
            turn = 1
    elif ¬ empty (OK_carsS):
            turn = 2
    if pedestrian == 0:
        if ¬ empty (OK_carsN):
                signalC(OK_carsN)  (notify_all)
        else:
                signalC(OK_carsS)  (notify_all)

operation entra_carN:
    if pedestrian ≠ 0 or carSS ≠ 0 or (¬empty(OK_pedestrian) and turn ≠ 1) or (¬empty(OK_carsS) and turn ≠ 1)
            waitC(OK_carsN)
    carsN <- carsN + 1

operation sale_carN:
    carsN <- carsN - 1
    if ¬empty (OK_carsS):
        turn = 2
    elif ¬ empty (OK_pedestrian):
        turn = 0
    if carsN == 0:
        if ¬empty (OK_carsS):
            signalC(OK_carsS)  (notify_all)
        else:
            signalC(OK_pedestrian)  (notify_all)
```

operation entra_carS:

```
if pedestrian ≠ 0 or carsN ≠ 0 or (¬empty(OK_pedestrian) and turn ≠ 2) or (¬empty(OK_carsN) and turn ≠ 2)
        waitC(OK_carsN)
    carsS <- carsS + 1

operation sale_carS:
    carS <- carS - 1
    if ¬empty(OK_pedestrian):
        turn = 0
    elif ¬empty(OK_carsN):
        turn = 1
    if carsS == 0:
        if ¬empty(OK_pedestrian):
            signalC(OK_pedestrian) (notify_all)
        else:
            signalC(OK_pedestrian) (notify_all)
```

- Escribe el invariante del monitor: Usando la notación $\begin{cases} P \equiv \text{pedestrian} \\ CN \equiv \text{CarsN} \\ CS \equiv \text{CarsS} \end{cases}$

  * Invariantes (asociados a la no-negatividad): $\boxed{P \geq 0 \quad CN \geq 0 \quad CS \geq 0}$

  * Invariantes (de la seguridad): $\boxed{\begin{array}{l} (P > 0 \rightarrow CN = 0 \land CS = 0) \land \\ \land (CN > 0 \rightarrow P = 0 \land CS = 0) \land \\ \land (CS > 0 \rightarrow P = 0 \land CN = 0) \end{array}}$

  * Invariantes (garantizan la ausencia de inanición, empleado para la demostración)

$$\boxed{\begin{array}{l} \neg empty(OK\_\cancel{peatón}\ pedestrian) \Rightarrow CN \neq 0 \lor CS \neq 0 \lor (\neg empty(OK\_carsN) \land turn \neq 0) \lor (\neg empty(OK\_carsS) \land turn \neq 0) \\ \neg empty(OK\_carsN) \Rightarrow P \neq 0 \lor CS \neq 0 \lor (\neg empty(OK\_pedestrian) \land turn \neq 1) \lor (\neg empty(OK\_CarsS) \land turn \neq 1) \\ \neg empty(OK\_carsS) \Rightarrow P \neq 0 \lor CN \neq 0 \lor (\neg empty(OK\_pedestrian) \land turn \neq 2) \lor (\neg empty(OK\_CarsN) \land turn \neq 2) \end{array}}$$

PRPA

- **Demuestra que el puente es seguro:**

Para estructurar la demostración, observamos que el monitor presenta 6 operaciones, y cada operador de salida (que son 3) presenta dos signal C. Por tanto, este monitor tiene 12 estados atómicos de ejecución. Veamos si a partir de estos se verifica el invariante de la seguridad:

- Para los 6 primeros estados, los asociados a los operadores, las operaciones "entra" para los 3 objetos {carN, carS, pedestrian} presentan un if que garantizan el invariante impidiendo que coincidan dos objetos contemporáneamente en el puente. Las 3 operaciones "salen" se ejecutan después de una operación "entra" respectiva que no ha modificado la seguridad, por tanto también la respetan.
- Los otros 6 estados, asociados a los procesos waiting, se encuentran situados en una operación "salen", con un if previo que garantiza que se ha vaciado el puente, luego al liberar el proceso waiting pertinente que por el IRR se ejecutará de forma inmediata, se continua con la ejecución del correspondiente proceso de entrada, verificando la seguridad, pues el puente está vacío, y por la estructura del operador sale, solo libera los procesos waiting de un único operador "entra".

- **Demuestra la ausencia de deadlock:**

El deadlock ocasionado porque todos los objetos se ceden el paso, se produce al salir un objeto {carN, carS, pedestrian} que vacía el puente, y hay elementos waiting para cada objeto, luego ninguno pasa al puente.

Para evitarlo, definimos una variable turn $\in \{0,1,2\}$ que siempre da preferencia a uno, así, si es su turno no cede el paso.

En su implementación, cuando el último elemento de un objeto sale del puente vaciándolo, cede el turno al siguiente objeto, las cesiones del turno siguen en preferencias una estructura triangular:

$$\begin{tikzcd} CS \arrow[dr] & & CN \arrow[ll] \\ & P \arrow[ur] & \end{tikzcd}$$

Los peatones dan prioridad a ceder el turno a CN, CN a CS y CS a P.

De forma que, el nuevo objeto al ser liberado no producirá deadlock pues aunque haya elementos esperando de los otros dos objetos, el turno le permite seguir su ejecución sin "dejar paso".

Una demostración intuitiva; el deadlock se produce en los wait C, si todos están bloqueados simultáneamente y dependen de los otros para ser liberados. Examinando la forma de los if:

if $\underbrace{\text{objeto1} \neq 0 \text{ or objeto2} \neq 0}$ or $\underbrace{(\neg\text{empty(OK.objeto1) and turn} \neq x) \text{ or } (\neg\text{empty(OK.objeto2) and turn} \neq x)}$

Si se bloquea por esto es que hay un elemento pasando luego no hay deadlock

No se pueden bloquear los 3 a la vez pues turn $\in \{0,1,2\}$, así $x \in \{0,1,2\}$ turn $= x$ para alguno de los operadores de salida

- **Demuestra la ausencia de inanición:**

Sin perdida de generalidad pues la definición de los operadores para los 3 objetos sigue la misma estructura. Vamos a probar que no hay inanición de los peatones siendo enteramente similar la demo. para carS y carN.

En primer lugar, el bloqueo se produciría en OK-pedestrian, bastaría comprobar la siguiente implicación: $\neg$ empty(OK-pedestrian) $\rightarrow$ signalC(OK-pedestrian)

es decir, si hay peatones esperando entonces se produce un signalC de estos en algún momento.

Para ello, argumentamos por reducción al absurdo supongamos $\neg$empty(OK-pedestrian) $\wedge$ $\neg$signalC(OK-pedestrian)

En primera instancia tenemos el siguiente invariante:

$\neg$ empty(OK-pedestrian) $\Rightarrow$ CN $\neq 0$ $\vee$ CS $\neq 0$ $\vee$ ($\neg$ empty(OK-carsN) $\wedge$ turn $\neq 0$) $\vee$ ($\neg$empty(OK-carsS) $\wedge$ turn $\neq 0$)

Comprobaremos que todos ellos inducen a signalC(OK-pedestrian):

- CN $\neq 0$: Están pasando coches por el norte, eventualmente llegará un último coche que pase (pues al haber peatones esperando se detiene la entrada de coches norte, habiendo cambiado el turno al salir el primer coche) Los coches norte dan prioridad a los coches sur luego:
  - si $\neg$ empty(OK-carS)., hay coches sur esperando estos pasarán hasta que finalmente pase un último (se detiene la entrada de coches sur pues hay peatones esperando que cambiarán el turno al salir el primer coche sur) y estos dan prioridad a los peatones esperando, luego acaba en signalC(OK-pedestrian)
  - si no hay coches sur esperando: directamente ejecuta signalC(OK-pedestrian)
- CS $\neq 0$: Por el mismo argumento, habrá un último coche sur pues cuando pase el primero se cambia el turno y al haber peatones esperando dejan de entrar coches sur y los coches sur dan prioridad a los peatones luego acaba en signalC(OK-pedestrian)

- $\neg$empty(OK-carN) and turn $\neq 0$:
  Hemos probado CN $\neq 0$ y CS $\neq 0$ conducen a signalC(OK-pedestrian)
  Supongamos CN $= 0$ y CS $= 0$. Tenemos dos posibilidades:
  - P $\neq 0$: argumento similar a CN $\neq 0$ y CS $\neq 0$
  - P $= 0$. En este caso, sabemos turn $\neq 0$, luego haya o no peatones esperando no van a entrar pues hay carN esperando y no es el turno del peatón. Así o bien entra carN o bien carS en cualquiera de los casos volvemos a CS $\neq 0$ y CN $\neq 0$, que llevan a signalC(OK-pedestrian)

- $\neg$ empty(OK-carS) and turn $\neq 0$. De forma simétrica al argumento anterior.

Finalmente, llegamos a una contradicción lo que verifica:

$\neg$empty(OK-pedestrian) $\rightarrow$ signalC(OK-pedestrian)