Teresa Ballester Navarro

PRÁCTICA 2 - PROGRAMACIÓN PARALELA

(1) PRÁCTICA PUENTE (versión 1)

↳ con Peatones

cochesN : coches cruzando el puente hacia el norte
cochesS : " " " " " " sur
peatones : peatones cruzando el puente

Monitor

cochesN : int = 0
cochesS : int = 0
peatones : int = 0
pasoCoches : VC = True
pasoPeatones : VC = True

coche

loop
El coche en dirección d quiere entrar
quiereEntrar-coche(d)
El coche entra
el coche sale
salida-coche(d)

peaton

loop
El peaton quiere entrar
quiereEntrar-peaton()
peaton entra
peaton sale
salida-peaton()

$INV \equiv \{ CN \geq 0,\ CS \geq 0,\ P \geq 0,\ CN > 0 \rightarrow (CS = 0 \wedge P = 0),$
$CS > 0 \rightarrow (CN = 0 \wedge P = 0),\ P > 0 \rightarrow (CN = 0 \wedge CS = 0)$
$CN \equiv cochesN,\ CS \equiv cochesS,\ P \equiv peatones \}$

```
quiereEntrar-coche(d)
 {INV}
    if d == NORTH :
        pasoCoches.wait(cochesS == 0 ∧ peatones == 0)
        cochesN += 1
    if d == SOUTH :
        pasoCoches.wait(cochesN == 0 ∧ peatones == 0)
        cochesS += 1
 {INV ∧ (CN > 0 ∨ CS > 0)}

salida-coche(d):
{INV ∧ (CN > 0 ∨ CS > 0)}
    if d == NORTH :
       cochesN -= 1
    else :
       cochesS -= 1
    pasoCoches.notify-all()
    pasoPeatones.notify-all()
{INV}

quiereEntrar-peaton():
 {INV}
     pasoPeatones.wait(cochesN == 0 ∧ cochesS == 0)
     peatones += 1
 {INV ∧ P > 0}
```

salida-peaton()

¿INV ^ P > 0 ¿

peatones -= 1

pasoCoches.notify-all()

Esta solución cumple

↳ La exclusión mutua, pues no pueden ejecutarse simultáneamente varios procesos

↳ No hay deadlocks siempre que hay una variación en el puente se notifica a todos

↳ El invariante se conserva en todo momento.

↳ Sin embargo, hay problema con la inanición. Por ejemplo, puede ocurrir que una vez entre un peatón entren continuamente todos los peatones que son un mayor número al número de coches hacia el norte (por ejemplo) y entonces el coche esperando a ir al norte no pasará nunca.

(2) PRÁCTICA PUENTE (versión 2 sin INANICIÓN)

Para que no haya inanición introducimos una variable que indique el turno. De esta forma, siempre que algún coche/peatón ya haya cruzado el puente y haya alguien esperando a entrar, se cambia el turno. Así sucesivamente hasta que todos los coches y peatones hayan cruzado y, entonces se indicará con el turno que no hay nadie (turn = 0).

$INV \equiv \{ CN \geq 0, CS \geq 0, P \geq 0,$

$CN > 0 \rightarrow (CS == 0 \wedge P == 0 \wedge (turn == 1 \vee turn == 0),$

$CS > 0 \rightarrow (CN == 0 \wedge P == 0 \wedge (turn == 2 \vee turn == 0),$

$P > 0 \rightarrow (CN == 0 \wedge CS == 0 \wedge (turn == 3 \vee turn == 0) \}$

Monitor

```
CN : int = 0
CS : int = 0
P : int = 0
turn : int = 0
waiting-CN : int = 0
waiting-CS : int = 0
waiting-P : int = 0
pasoCoches_N : VC = True
pasoCoches-S : VC = True
pasoPeatones : VC = True

quiereEntrar-coche(d):
{INV}
  if d == NORTH:
    waiting-CN += 1
    pasoCoches_N.wait(CS == 0 ∧ P == 0 ∧ (turn == 1 ∨ turn == 0))
    waiting-CN -= 1
    CN += 1
    turn = 1
  else:
    waiting-CS += 1
    pasoCoches-S.wait(CN == 0 ∧ P == 0 ∧ (turn == 2 ∨ turn == 0))
    waiting-CS -= 1
    CS += 1
    turn = 2
{INV}
```

```
salida-coche(d):
  {INV ∧ (CN > 0 ∨ CS > 0)}
  if d == NORTH :
        SOUTH → caso else
     CN -= 1
     CS
     if waiting-CS ≠ 0 :
        -CN
        turn = 2
                1
     elif waiting-P ≠ 0 :
        turn = 3
     else :
        turn = 0
     if CN == 0 :
     CS                N
        pasoCoches-S.notify-all()
        pasoCoches-P notify-all()
  {INV}
quiereEntrar-peaton():
  {INV}
    waiting-P += 1
    pasoPeatones.wait(CS==0 ∧ CN==0 ∧ (turn==3 ∨ turn==0))
    waiting-P -= 1
    P += 1
    turn = 3
  {INV}

  salida-peaton():
  {INV ∧ P > 0}
     P -= 1
     if waiting-CN ≠ 0 :
        turn = 1
     elif waiting-CS ≠ 0 :
        turn = 2
     else :
        turn = 0
  {INV}
```

```
if P == 0 :
  pasoCoches-N.notify-all()
  pasoCoches-S.notify-all()
```