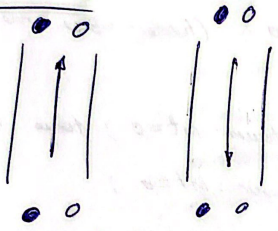
Práctica Puente PRPA

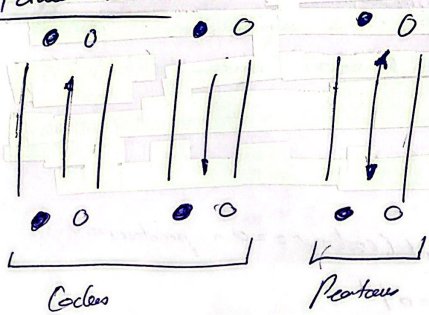
En el puente de Acubite existen diferentes estados de paso entre coches y peatones. La práctica consiste en hacer una sincronización de los coches con los peatones de 3 diferentes formas:

Forma 1



En esta forma no hay peatones y los coches pueden pasar de un lado a otro pero solo en 1 dirección, es decir, no puede haber coches circulando en sentidos opuestos.

Forma 2



En esta forma hay peatones y la restricción que le ponemos es que no pueden entrar el puente cuando hay coches circulando mientras, pero sí que le pueden hacer en ambas direcciones a diferencia de los coches, en la que mantenemos la restricción de que no pueden circular por el puente en doble sentido como en la forma 1.

Para asegurar una buena sincronización entre coches nos hemos asegurado de que cada vez que un flujo de coches empieza a entrar en el puente, este está previamente vacío.

Despertamos con el notifyAll() a todos los coches para que no haya retrasos ni inanición.

Scanned with CamScanner

Los problemas de interbloqueo (deadlocks) los solucionamos estableciendo turnos. Sin turnos podría ocurrir que dos coches se quedaran parados y no cruzaran el puente.

invariante ≡ { coches norte ≥ 0, coches sur ≥ 0, Peatones ≥ 0,
coches norte > 0 ⇒ (coches norte == 0 ∧ peatones == 0 ∧ (turno == 1 ∨ turno == 0))
coches sur > 0 ⇒ (coches norte == 0 ∧ peatones == 0 ∧ (turno == 2 ∨ turno == 0))
peatones > 0 ⇒ (coches norte == 0 ∧ coches sur == 0 ∧ (turno == 3 ∨ turno == 0)) }

monitor

coches norte : int = 0 ; coches sur : int = 0 , peatones : int = 0 ; turno : int = 0

espera coches_norte = 0 : int = 0 ; espera coches_sur : int = 0 ;

espera peatones : int = 0 ; entrada coches_norte : VC = true;

entrada coches_sur : VC = true ; entrada peatones : VC = True;

def solicita entrar_coche (dirección)

{Invariante}

si dirección == NORTH ;

espera coches_norte += 1

entrada coches_norte . wait (coches sur == 0 ∧ peatones == 0 ∧

(turno == 1 ∨ turno == 0))

espera coches_norte -= 1

coches norte += 1

turno = 1

else:

espera coches_sur += 1

entrada coches_sur . wait (coches norte == 0 ∧ peatones == 0 ∧

∧ turno == 0)

espera coches_sur -= 1

coches sur += 1

turno = 2

{Invariante}

Scanned with CamScanner

```
def salida_coches(dirección):
    {Invariante}
    if dirección == NORTH:
        coches_norte -= 1
        if espera_coches_sur ≠ 0:
            turno = 2
        elif espera_peatones ≠ 0:
            turno = 3
        else:
            turno = 0
        if coches_norte == 0
            entrada_coches_sur.notify_all()
            entrada_coches_norte.notify_all()
    {Invariante}
```

```
    else:
        coches_sur -= 1
        if espera_coches_norte ≠ 0:
            turno = 1
        elif espera_peatones ≠ 0:
            turno = 3
        else:
            turno = 0
        if coches_sur == 0:
            entrada_coches_sur.notify_all()
            entrada_peatones.notify_all()
```

```
def solicita_cruzar_peatones():
    {Invariante ∧ peatones > 0}
        peatones -= 1
        if espera_coches_norte ≠ 0:
            turno = 1
        elif espera_coches_sur ≠ 0:
            turno = 2
        else:
            turno = 0
        if peatones == 0:
            entrada_coches_norte.notify_all()
            entrada_coches_sur.notify_all()
    {Invariante}
```

El puente es seguro dado que establecimos turnos entre coches y peatones se asegura que no coincidan dentro del puente.

Scanned with CamScanner