Práctica 2 - PRPA

- Primero desarrollamos una solución sencilla, y después la mejoraremos para que no tenga deadlocks, inanición y que el puente sea seguro (no haya choques ni atropellos)

Sol sencilla:

- **Monitor**

```
Inv{
num_coches_norte: int=0
num_coches_sur: int=0
num_peatones: int=0
norte_coches_esperando: int=0
sur_coches_esperando: int=0
peatones_esperando: int=0
coches_sur: VC
coches_norte: VC
peatones: VC
turn: int=0 # 0→esta vacio el puente, 1→turno coches norte, 2→turno coches sur, 3→turno peatones
Inv}
```

- El invariante se seguirá manteniendo pues en ningún momento tendremos dos coches en distintos sentidos en el puente ni ningún peatón con coches al mismo tiempo.

- El invariante será:

$$Inv \equiv \begin{cases} num\_coches\_norte \geq 0,\ num\_coches\_sur \geq 0,\ num\_peatones \geq 0 \\ norte\_coches\_esperando \geq 0,\ sur\_coches\_esperando \geq 0,\ peatones\_esperando \geq 0 \\ num\_coches\_norte > 0 \rightarrow num\_coches\_sur = 0 \wedge num\_peatones = 0 \\ num\_coches\_sur > 0 \rightarrow num\_coches\_norte = 0 \wedge num\_peatones = 0 \\ num\_peates > 0 \rightarrow num\_coches\_norte = 0 \wedge num\_coches\_sur = 0 \\ turn \in \{0,1,2,3\} \end{cases}$$

- **Wants_enter_car(direction)**

```
if direction == North:
    norte_coches_esperando += 1
    coches_norte.wait((num_coches_sur==0 ∧ num_peates==0) ∧ (turn==1 ∨ turn==0))
    norte_coches_esperando -= 1
    turn = 1
    num_coches_norte += 1
```

*

Continuación función wants_enter_car

* elif direction == South:
    sur_coches_esperando += 1
    coches_sur.wait((num_coches_norte == 0 ∧ num_peatones == 0) ∧ (turn == 2 ∨ turn == 0))
    sur_coches_esperando -= 1
    turn = 2
    num_coches_sur += 1

- leaves_car(direction)
    if direction == North: → {Inv ∧ num_coches_norte > 0}
        num_coches_norte -= 1
        if num_coches_norte == 0:
            coches_sur.notify_all()
            peatones.notify_all()
            turn = 0
    elif direction == South: → {Inv ∧ num_coches_sur > 0}
        num_coches_sur -= 1
        if num_coches_sur == 0:
            coches_norte.notify_all()
            peatones.notify_all()
            turn = 0

wants_enter_pedestrian()
    peatones_esperando += 1
    peatones.wait(num_coches_norte == 0 ∧ num_coches_sur == 0) ∧ (turn == 3 ∨ turn == 0)
    peatones_esperando -= 1
    turn = 3
    num_peatones += 1

leaves_pedestrian() → {Inv ∧ num_peatones > 0}
    num_peatones -= 1
    if num_peatones == 0:
        coches_norte.notify_all()
        coches_sur.notify_all()
        turn = 0

car(direction)
loop
monitor.wants_enter_car(direction)
monitor.leaves_car(direction)

pedestrian()
loop
monitor.wants_enter_pedestrian()
monitor.leaves_pedestrian()

- Esta solución será segura en cuanto a circulación y no habrá deadlock, pues un coche un una dirección no podrá entrar si hay coches en otra dirección dentro, o peatones, y enálogo para los peatones si hay coches en el puente. Esto se consigue gracias a los wait del want_enter_car y wants_enter_pedestrian()

- Pero, esta solución tiene inanición, pues, por ejemplo, supongamos que tenemos 50 coches que quieran pasar al norte y 1 coche que quiera pasar al sur:

Supongamos además que ya ha pasado un coche hacia el norte, entonces podrán pasar el resto detrás suya, y puede pasar que no paren de llegar coches que quieran ir al norte y entonces el coche que quiera ir al sur no podrá pasar nunca.

Solución sin inanición:

Para resolver el problema de la inanición, cambiaremos la función leaves_car(direction) y leaves_pedestrian(). Para ello, cuando un coche entre en una dirección (o peatón), podrán pasar coches en esa dirección hasta que el primero salga del puente, y entonces cambiamos el turno de forma rotativa (siempre que haya alguien esperando en el puente). Cambiamos dichas funciones. (No habrá inanición, pues iremos cambiando turnos, y tarde o temprano irán pasando todos los coches y peatones)

leaves_car:

```
if direction == North: → {Inv ∧ num_coches_norte>0}
    num_coches_norte -= 1 → no cambia Inv
    if sur_coches_esperando != 0:
        turn = 2 → no cambia Inv
    elif peatones_esperando != 0:
        turn = 3 → no cambia Inv
    else:
        turn = 0 → no cambia Inv
    if num_coches_norte == 0:
        coches_sur.notify_all()
        peatones.notify_all()
    {Inv}
elif direction == South: → {Inv ∧ num_coches_sur>0}
    num_coches_sur -= 1 → no cambia Inv
    if peatones_esperando != 0:
        turn = 3 → no cambia Inv
    elif norte_coches_esperando != 0:
        turn = 1 → no cambio Inv
    else:
        turn = 0 → no cambia Inv
    if num_coches_sur == 0:
        coches_norte.notify_all()
        peatones.notify_all()
    {Inv} ✓
```

leaves_pedestrian → {Inv ∧ num_peatones>0}

```
num_peatones -= 1 → no cambia Inv
if norte_coches_esperando != 0:
    turn = 1 → no cambia Inv
elif sur_coches_esperando != 0:
    turn = 2 → no cambia Inv
else:
    turn = 0 → no cambia Inv
if num_peatones == 0:
    coches_norte.notify_all()
    coches_sur.notify_all()
{Inv} ✓
```

No habrá inanición pues si vamos rotando los turnos de paso de forma arbitraria, todos pasarán tarde o temprano, pues se esa cambiando la prioridad de paso continuamente.