Monitor

car_north : int = 0
car_south : int = 0
ped : int = 0

cond_north_car : VC
cond_south_car : VC
cond_ped : VC

INV = { car_north ≥ 0 ∧ car_sout ≥ 0 ∧ ped ≥ 0
car_north > 0 → car_south ~~+~~ ped == 0
car_south > 0 → car_north ~~+~~ ped == 0
ped > 0 → car_north + car_south == 0 }

def wants_enter_car(direction):
    if direction == NORTH:
        cond_north_car.wait_for(car_south + ped == 0)
        car_north ~~+=~~ += 1 → se cumple el invariante
    elif ~~direction~~ direction == SOUTH:
        cond_south_car.wait_for(car_north + ped == 0)
        car_sout += 1 → se cumple el invariante

def leaves_car(direction):
    if direction == NORTH:
        car_north -= 1
        if car_north == 0:
            cond_south_car.notify_all() {No quedan esperando
            cond_ped.notify_all()
                → {INV}
    elif direction == SOUTH:
        car_sout -= 1
        if car_south == 0:
            cond_nort_car.notify_all() {No quedan esperando
            cond_ped.notify_all()
                → {INV}

```
def wants_enter_pedestrian()
    cond_ped.wait(car_north + car_south == 0)   -> Se cumple el invariante
    ped += 1

def leaves_pedestrian():
    ped -= 1
    if ped == 0:
        cond_north_car.notify_all()   { No se quedan esperando
        cond_sout_car.notify_all()
                                      -> {INV}
```

Gracias a las variables condición, no es posible que haya coches y peatones a la vez en el puente. Además, haciendo uso de estas y los notify, no queda ningún hilo en espera.

Para la posible inanición, se propone emular un semáforo real, que de paso a todos de forma que haya paso en ambos sentidos, en función del número de coches o peatones en espera. Hubiera podido hacerse también con un control de tiempo, o número de coches que ya han pasado en un sentido determinado.

Se añadiría: car_north_waiting : int = 0
car_sout_waiting : int = 0
ped_waiting : int = 0
semaphore = E (código de variables: E=1, empty; P=2, ped; N=3, north; S=4, south;)

Al invariante anterior se añadiría: car_north_waiting ≥ 0
car_sout_waiting ≥ 0
ped_waiting ≥ 0
semaphore ∈ {E, P, N, S}

En el código:

```
def wants_enter_car:
    car_north_waiting += 1
    cond_car_north.wait_for(car_north_enter)   -> Sigue siendo seguro
    car_north_waiting -= 1
    if semaphore == E
        semaphore = N   -> pasan más coches del N.

    car_north += 1
                -> {INV}
```

(*) -> if car_north == 0
cond_car_sout.notify_all()   { No se quedan
cond_ped.notify_all()   esperando.

(*)

```
elif direction == SOUTH:
    car_sout_waiting += 1
    cond_car_sout.wait_for(car_south_enter) → sigue siendo seguro.
    car_sout_waiting -= 1

    if semaphore == E:
        semaphore = S
    car_sout += 1
                → {INV}
```

```
def leaves_car()
    Solo cambia, además de actualizar los coches que están en el ~~bar~~ puente:

    if car_sout_waiting > 0:
        semaphore = S
    elif ped_waiting > 0:
        semaphore = P
    else:
        semaphore = E
```

Dependiendo de las condiciones (descritas en INV), se repite el poder entrar gracias al emulador de semáforo.

```
(+) if car_north == 0
        cond_car_sout.notify_all() { No se quedan esperando.
        cond_ped.notify_all()
```

De igual forma sucede en cuanto a peatones se refiere.

Así, todos tienen la oportunidad de entrar de forma organizada.