Práctica 2 — Monitor_CP (con posible inanición): ①

Manuel Pablo Bejarano Galeano

```
nped : int = 0
ncarNorth : int = 0
ncarSouth : int = 0
south_car : VC = False
north_car : VC = False
ped       : VC = False
```

El invariante es:

$$INV \equiv \{ nped \geq 0 \wedge ncarNorth \geq 0 \wedge ncarSouth \geq 0 \wedge$$
$$\wedge\ nped > 0 \rightarrow (ncarNorth = 0 \wedge ncarSouth = 0) \wedge$$
$$\wedge\ ncarNorth > 0 \rightarrow (nped = 0 \wedge ncarSouth = 0) \wedge$$
$$\wedge\ ncarSouth > 0 \rightarrow (nped = 0 \wedge ncarNorth = 0) \}$$

```
wants_enter_pedestrian():
    {INV}
    ped.wait( ncarNorth == 0 ∧ ncarSouth == 0)
    nped = nped + 1
    {INV ∧ nped > 0}
leaves_pedestrian():
    {INV ∧ nped > 0}
    nped = nped - 1
    {INV}
    if (nped == 0):
        north_car.notify()
        south_car.notify()
    {INV}

wants_enter_car(direction)  # direction: 1 → NORTH, 0 → SOUTH
    {INV}
    if( direction == 1 ): # NORTH
        north_car.wait( nped == 0 ∧ ncarSouth == 0)
        ncarNorth = ncarNorth + 1
        {INV ∧ ncarNorth > 0}
    else: # SOUTH
        south_car.wait (nped == 0 ∧ ncarNorth == 0)
        ncarSouth = ncarSouth + 1
        {INV ∧ ncarSouth > 0}
```

```
leaves_car (direction):
    if (direction == 1):          # NORTH
        ncarNorth = ncarNorth - 1
        if (ncarNorth == 0):
            south_car.notify()
        ped.notify()
    else:                         # SOUTH
        ncarSouth = ncarSouth - 1
        if (ncarSouth == 0):
            ped.notify()
        north_car.notify()
```

```
car (monitor, direction):
    monitor.wants_enter_car(direction)
    if (direction == 1):          # NORTH
        operaciones para pasar car NORTH
    else:                         # SOUTH
        operaciones para pasar car SOUTH
    Monitor.leaves_car(direction)
```

```
pedestrian (monitor):
    monitor.wants_enter_pedestrian()
    operaciones para pasar peatón
    monitor.leaves_pedestrian().
```

- Tenemos garantizada la seguridad ya que en el invariante se cumple que si uno de los grupos está pasando por el puente, el resto no lo hace. Además dicho invariante hemos comprobado que se cumple durante toda la ejecución.

PRÁCTICA 2 — Manuel Pablo Bejarano Galeano:

Monitor_CP: (Sin inanición)

nped : int = 0
n carNorth : int = 0
n carSouth : int = 0
nped_waiting : int = 0
ncarNorth_waiting : int = 0
ncarSouth_waiting : int = 0.
southCar : VC = False
northCar : VC = False
ped : VC = False
turn : int = -1

El invariante es:

INV ≡ { 0 ≤ nped ∧ 0 ≤ n carNorth ∧ 0 ≤ n carSouth ∧
∧ 0 ≤ nped_waiting ∧ 0 ≤ ncarNorth_waiting ∧ 0 ≤ ncarSouth_waiting ∧
∧ -1 ≤ turn ≤ 2 ∧ nped > 0 → (ncarNorth = 0 ∧ ncarSouth = 0) ∧
∧ ncarNorth > 0 → (nped = 0 ∧ ncarSouth = 0) ∧
∧ ncarSouth > 0 → (nped = 0 ∧ ncarNorth = 0) }

wantsEnterPedestrian():
{INV ∧ (turn = 0 ∨ turn = -1)}
nped_waiting = nped_waiting + 1
{INV ∧ nped_waiting > 0 ∧ (turn = 0 ∨ turn = -1)}
ped.wait((ncarNorth = = 0 ∧ ncarSouth = = 0) ∧ (((ncarSouth_waiting ≤ 5 ∧
∧ ncarNorth_waiting ≤ 5) ∨ (turn = = 0)) ∨ turn = = -1))
nped_waiting = nped_waiting - 1
{INV ∧ (turn = 0 ∨ turn = -1)}
if turn = = -1:
turn = 0
{INV ∧ nped > 0 ∧ turn = 0}
nped = nped + 1
{INV ∧ nped > 0 ∧ turn = 0}

```
leaves_pedestrian():
    {INV ∧ nped > 0}
nped = nped - 1
    {INV}
if turn == 0:
    if ncarNorth_waiting != 0:
        turn = 1
        {INV ∧ turn = 1}
    elif ncarSouth_waiting != 0:
        turn = 2
    else: {INV ∧ turn = 2}
        turn = -1
        {INV ∧ turn = -1}
if nped == 0:
    north_car.notify()
    south_car.notify()
{INV ∧ turn ≠ 0}
```

```
wants_enter_car(direction):   direction {1: SOUTH, 0: NORTH}
{INV ∧ (turn == 1 ∨ turn == 1 ∨ turn == 2)}
if direction == 0:
    {INV ∧ (turn == -1 ∨ turn == 1)}
    ncarNorth_waiting = ncarNorth_waiting + 1
    {INV ∧ (turn == -1 ∨ turn == 1) ∧ ncarNorth_waiting > 0}
    north_car.wait((nped == 0 ∧ ncarSouth == 0) ∧
        ¬(((ncarSouth_waiting <= 5 ∧ nped_waiting <= 5) ∨
            ∨ turn == 1) ∨ turn == -1)))
    ncarNorth_waiting = ncarNorth_waiting - 1
    {INV ∧ (turn == 1 ∨ turn == -1)}
    if turn == -1:
        turn = 1
else: ncarNorth = ncarNorth + 1   {INV ∧ turn = 1 ∧ ncarNorth > 0}
    {INV ∧ (turn == -1 ∨ turn == 2)}
    ncarSouth_waiting = ncarSouth_waiting + 1
    {INV ∧ (turn == -1 ∨ turn == 2) ∧ ncarSouth_waiting > 0}
    southCar.wait((nped == 0 ∧ ncarNorth == 0) ∧
        ¬(((ncarNorth_waiting ≤ 5 ∧ nped_waiting ≤ 5) ∨
            ∨ turn == 2) ∨ turn == -1)))
    ncarSouth_waiting = ncarSouth_waiting - 1
    {INV ∧ (turn == -1 ∨ turn == 2)}
    if turn == -1:
        turn = 2
{INV ∧ turn == 2}
ncarSouth = ncarSouth + 1
{INV ∧ ncarSouth > 0}
```

```
leaves_car ( direction):                    ∧ ncar North + ncar South >0}
    {INV
    if (direction == 1):  # NORTH
        {INV ∧ ncar North >0}
        ncar North = ncar North - 1
        {INV}
        if (turn == 1):
            if (n car South_waiting != 0):
                turn = 2
                {INV ∧ turn = 2}
            elif (n ped_waiting != 0):
                turn = 0
            else:  {INV ∧ turn = 0}
                turn = -1
                {INV ∧ turn = -1}
        if ncar North == 0:
            South_car . notify ()
            ped . notify()
        {INV}
    else:  # SOUTH
        {INV ∧ ncar South >0}
        n car South = ncar South - 1
        {INV}
        if turn == 2:
            if (n ped_waiting != 0):
                turn = 0
                {INV ∧ turn = 0}
            elif (n car North_waiting != 0):
                turn = 1
            else:  {INV ∧ turn = 1}
                turn = -1
                {INV ∧ turn = -1}
        if (n car South == 0):
            north_car. notify ()
            ped . notify()
        {INV}
```

```
car (direction, monitor):
    monitor. wants_enter_car (direction)
    if (direction == 1): #NORTH
        operaciones para pasar car North
    else:
        operaciones para pasar car South
    monitor. leaves_car (direction)
```

```
pedestrian(monitor):
    monitor. wants_enter_pedestrian ()
    operaciones para pasar peatón
    monitor. leaves_pedestrian()
```

- La seguridad en este programa "ampliado" se cumple ya que en el invariante se especifica nped > 0 → (ncarNorth = 0 ∧ ∧ ncarSouth = 0) ∧ ncarNorth > 0 → (nped = 0 ∧ ncarSouth = 0) ∧ ∧ ncarSouth > 0 → (nped = 0 ∧ ncarNorth = 0) y el invariante se cumple durante toda la ejecución

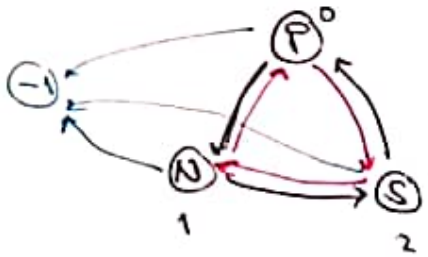
- La ausencia de deadlock se garantiza ya que si por ejemplo los peatones se encuentran en el puente y dejan de tener su turno, ya sea porque cuando sale uno de los peatones hay algún coche esperando o porque el puente queda libre, el turno pasará a ser del coche o al turno "neutro" (-1). Por lo tanto nunca habrá nadie esperando a que otro termine de esperar y por tanto no habrá deadlock.

- La ausencia de inanición se demuestra por cómo se ha definido el turno y como pasa de uno a otro:

El turno toma los valores:
- -1 → neutro
- 0 → peatón
- 1 → coche Norte
- 2 → coche Sur

Cuando el un pasante (peatón o coche) sale se verifica si sigue teniendo el turno, si es así pasa el turno a aquel grupo que tenga algún miembro esperando, si no hay nadie esperando, el turno pasa a ser -1 hasta que pase el siguiente pasante y lo cambie por el suyo.

El cambio de turno tiene prioridades, así se verifica que no haya inanición. El peatón revisa primero que no haya esperando coches N y luego coches S, los coches N revisa primero Coches S y luego peatones y, finalmente, los coches S revisan primero los peatones y luego los coches N



→ : 1ª verificación
→ : 2ª verificación
→ 3ª verificación.