TÚNEL SOLUCIÓN BÁSICA

Alejandro Miranda Cortés

Héctor Rodríguez Peco

Marta Impuesto Rico

TUNNEL SOLUCIÓN BÁSICA: demostración

Alejandro Miranda Cortés
Héctor Rodríguez Pecq
Marta Impuesto Rico

```
car : cierto
  Loop
    direction:= random(NORTH, SOUTH)
    monitor. wants_enter(direction)
    monitor. leaves_tunnel(direction)
```

monitor:

cars_north : Int ↳ # coches yendo hacia el norte en el túnel.

cars_south : Int ↳ # coches yendo hacia el sur en el túnel.

someone_north: condition

someone_south: condition

{INV: cars_north ≥ 0 ∧ cars_south ≥ 0 ∧ (¬(cars_noth > 0 ∧ cos_south > 0)}

```
def wants_eat(direction):
  {INV}
  if direction == NORTH
      someone_south.wait(cars_south==0)
      {INV ∧ cars_south==0}
      carS_noth:= cos_noth +1
      {INV}
  else : ↳ (direction == South)
      someone_north.wait (cars_noth ==0)
      {INV∧ cos_noth==0}
      cars_south:= cos_South+1
      {INV}
  {INV}
```

- INV ∧ cars_south == 0 ⟺ cos_north ≥ 0 ∧ cos_south == 0 ∧

∧ (¬(cars_north > 0 ∧ cos_south > 0)) ⟺ cos_north ≥ 0 ∧ cos_south == 0 ⇒

(cos_south > 0: foso; cars_north > 0 ∧ cos_south > 0: falso; ¬(cars_north > 0 ∧ cos_south > 0): cierto)

[Los primeros dos términos de la conjunción se siguen mutuamente y el último porque cars_south == 0]

⇒ INV ✓

⇒ cos_north ≥ 1 ∧ cos_south == 0 ⇒ cos_north > 0 ∧ cos_south == 0

↓

cars_noth:= cos_noth +1

análogo al otro

• INV ∧ cars_north == 0 <=> cars_north == 0 ∧ cars_south ≥ 0 ⇒ cars_north == 0 ∧

∧ cars_south ≥ 1 ⇒ cars_noth == 0 ∧ cars_south > 0 ⇒ INV ✓ → cars_south := cars_south + 1

↓

pero con cars_north == 0

def leave_tunnel(direction):

+ INV ∧ ((cars_north > 0 ∧ cars_south == 0 ∧ direction == NORTH) ⊕

⊕ (cars_south > 0 ∧ cars_north == 0 ∧ direction == SOUTH) +

if direction == NORTH:

+ INV ∧ cars_north > 0 ∧ cars_south == 0 +

cars_north := cars_north - 1

someone_north.signal()

+ INV +

else: ↳ (direction == south)

+ INV ∧ cars_south > 0 ∧ cars_north == 0 +

cars_south := cars_south - 1

someone_south.signal()

+ INV +

+ INV +

• INV ∧ cars_north > 0 ∧ cars_south == 0 <=> cars_north ≥ 0 ∧ cars_south ≥ 0 ∧

∧ (¬(cars_north > 0 ∧ cars_south > 0)) ∧ cars_north > 0 ∧ cars_south == 0 <=>

<=> cars_north > 0 ∧ cars_south == 0 ⇒ cars_north + 1 > 0 ∧ cars_south == 0 <=>

↓

cars_north := cars_north - 1

<=> cars_north ≥ 0 ∧ cars_south == 0 ⇒ INV ✓

análogo al otro caso

• INV ∧ cars_south > 0 ∧ cars_north == 0 <=> cars_south > 0 ∧ cars_north == 0 ⇒

=> cars_south + 1 > 0 ∧ cars_north == 0 (=> cars_north ≥ 0 ∧ cars_south == 0 ⇒ INV ✓

↓

cars_south := cars_south - 1

↓

pero cars_north == 0