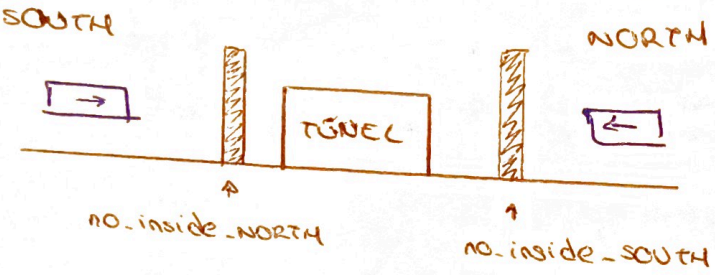
ALUMNAS: Claudia Casado Poyatos
Olga Rodriguez Acevedo
Natalia Garcia Dominguez

TÚNEL DE KIYOTAKI



Caso básico

car_NORTH : coches del norte dentro del túnel
car_SOUTH : coches del sur dentro del túnel

INVARIANTE: ¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧
car_NORTH ≥ 0 ∧ car_SOUTH ≥ 0

no_inside_NORTH : Condición (car_NORTH ?== 0)
no_inside_SOUTH : Condición (car_SOUTH ?== 0)

def: wants_enter_SOUTH :
} car_SOUTH ≥ 0 {
no_inside_NORTH.wait(car_NORTH == 0)
car_SOUTH += 1 } car_SOUTH > 0 {

Dem: car_SOUTH > 0 ∧ car_NORTH == 0 => ¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧
∧ car_NORTH ≥ 0 ∧ car_SOUTH ≥ 0 <=> ¬(falso ∧ cierto) ∧ cierto
∧ cierto <=> ¬falso ∧ cierto <=> cierto

def: leaves_tunnel_SOUTH:

{ car_SOUTH > 0 ∧ car_NORTH == 0 }

car_SOUTH -= 1

no_inside_SOUTH.signal() { car_SOUTH ≥ 0 }

Demo: car_SOUTH ≥ 0 ∧ car_NORTH == 0 ⇒

¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧ car_NORTH ≥ 0 ∧ car_SOUTH ≥ 0 ⇔

⇔ ¬(falso ∧ car_SOUTH > 0) ∧ cierto ∧ cierto ⇔

⇔ ¬(falso) ∧ cierto ⇔ cierto

def: wants_enter_NORTH:

{ car_NORTH ≥ 0 }

no_inside_SOUTH.wait(car_SOUTH == 0)

car_NORTH += 1 { car_NORTH > 0 }

Demo: car_NORTH > 0 ∧ car_SOUTH == 0 ⇒

¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧ car_NORTH ≥ 0 ∧ car_SOUTH ≥ 0 ⇔

¬(cierto ∧ falso) ∧ cierto ∧ cierto ⇔ cierto

def: leaves_tunnel_NORTH:

{ car_NORTH > 0 ∧ car_SOUTH == 0 }

car_NORTH -= 1

no_inside_NORTH.signal { car_NORTH ≥ 0 }

Demo: car_NORTH ≥ 0 ∧ car_SOUTH == 0 ⇒

¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧ car_NORTH ≥ 0 ∧ car_SOUTH ≥ 0 ⇔

¬(car_NORTH > 0 ∧ falso) ∧ cierto ∧ cierto ⇔

¬(falso) ∧ cierto ⇔ cierto.

Caso turnos

car_NORTH : coches del norte dentro del túnel
car_SOUTH : coches del sur dentro del túnel

queue_NORTH : coches del norte esperando para entrar
queue_SOUTH : coches del sur esperando para entrar

turn : variable que determina la dirección de los coches que pueden pasar

0 → si pasan los del norte
1 → si pasan los del sur

INUARIANTE : $\neg(car\_NORTH > 0 \wedge car\_SOUTH > 0) \wedge$
$car\_NORTH \geq 0 \wedge car\_SOUTH \geq 0 \wedge$
$(turn == 0 \vee turn == 1) \wedge$
$queue\_NORTH \geq 0 \wedge queue\_SOUTH \geq 0$

no_inside_NORTH : Condicion (car_NORTH == 0 ∧ (turn == 1 ∨ queue_NORTH == 0))

no_inside_SOUTH : Condicion (car_SOUTH == 0 ∧ (turn == 0 ∨ queue_SOUTH == 0))

def wants_enter_SOUTH :
{car_SOUTH ≥ 0 ∧ queue_SOUTH ≥ 0 }
queue_SOUTH += 1
no_inside_NORTH.wait (car_NORTH == 0 ∧ (turn = 1 ∨ queue_NORTH == 0))
queue_SOUTH -= 1
car_SOUTH += 1
{car_SOUTH > 0 ∧ queue_SOUTH ≥ 0 ∧ queue_NORTH ≥ 0 }

DEMO: car_NORTH == 0 ∧ (turn == 1 ∨ queue_NORTH == 0) ∧ car_SOUTH > 0 ∧
queue_SOUTH ≥ 0 ∧ queue_NORTH ≥ 0 =>

¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧ car_NORTH ≥ 0 ∧
car_SOUTH ≥ 0 ∧ (turn == 0 ∨ turn == 1) ∧
queue_NORTH ≥ 0 ∧ queue_SOUTH ≥ 0 <=>
¬(falso ∧ cierto) ∧ cierto ∧ cierto ∧ (turno == 0 ∨ turn == 1)
∧ cierto ∧ cierto <=> cierto (*)

* { • si turn == 1 → (turn == 0 ∨ turn == 1) ≡ cierto
  • si turn != 1 ∧ queue_NORTH == 0 => turn == 0 => cierto

def leaves_tunnel_SOUTH:
} car_SOUTH > 0 ∧ turn == 1 ∧ car_NORTH == 0
car_SOUTH -= 1
turn = 0
NO_inside_SOUTH.signal()
} car_SOUTH ≥ 0 ∧ turn == 0 {

DEMO: car_SOUTH ≥ 0 ∧ car_NORTH == 0 ∧ turn == 0 ∧
queue_SOUTH ≥ 0 ∧ queue_NORTH ≥ 0 =>

¬(car_NORTH > 0 ∧ car_SOUTH > 0) ∧ car_NORTH ≥ 0 ∧
car_SOUTH ≥ 0 ∧ (turn == 0 ∨ turn == 1) ∧
queue_SOUTH ≥ 0 ∧ queue_NORTH ≥ 0 <=>
¬(falso ∧ car_SOUTH > 0) ∧ cierto ∧ cierto ∧ cierto ∧
cierto ∧ cierto <=> cierto

def wants_enter_NORTH :

{ car_NORTH ≥ 0 ∧ queue_NORTH ≥ 0 }

queue_NORTH += 1

no_inside_NORTH.wait(car_SOUTH == 0 ∧ (turn = 0 ∨ queue_SOUTH == 0))

queue_NORTH -= 1

car_NORTH += 1

{ car_NORTH > 0 ∧ queue_NORTH ≥ 0 ∧ queue_SOUTH ≥ 0 }

DEMO car_SOUTH == 0 ∧ (turn == 0 ∨ queue_SOUTH == 0) ∧ car_NORTH > 0 ∧ queue_NORTH ≥ 0 ∧ queue_SOUTH ≥ 0

¬(car_NORTH > 0 ∧ car_SOUTH ≥ 0) ∧ car_NORTH ≥ 0 ∧ car_SOUTH ≥ 0 ∧ (turn == 0 ∨ turn == 1) ∧ queue_NORTH ≥ 0 ∧ ∧ queue_SOUTH ≥ 0 <=> ¬(cierto ∧ falso) ∧ cierto ∧ cierto ∧ (turn == 0 ∨ turn == 1) ∧ cierto ∧ cierto <=> cierto *

* { • si turn == 0 => (turn == 0 ∨ turn == 1) ≡ cierto
  { • si turn != 0 ∧ queue_SOUTH == 0 => turn = 1 => cierto

def leaves_tunnel_NORTH :

{ car_NORTH > 0 ∧ turn == 0 ∧ car_SOUTH == 0 }

car_NORTH -= 1

turn = 1

no_inside_NORTH.signal()

{ car_NORTH ≥ 0 ∧ turn == 1 }

Demo: car_NORTH ≥ 0 $\wedge$ car_SOUTH $== 0$ $\wedge$ turn $== 1$ $\wedge$

$\wedge$ queue_SOUTH ≥ 0 $\wedge$ queue_NORTH ≥ 0 $\Rightarrow$

$\neg$ (car_NORTH > 0 $\wedge$ car_SOUTH > 0) $\wedge$ car_NORTH ≥ 0 $\wedge$

car_SOUTH ≥ 0 $\wedge$ (turn $== 0$ $\vee$ turn $== 1$) $\wedge$

queue_SOUTH ≥ 0 $\wedge$ queue_NORTH ≥ 0 $\Leftrightarrow$

$\neg$(car_NORTH > 0 $\wedge$ falso) $\wedge$ cierto $\wedge$ cierto $\wedge$ cierto $\wedge$

cierto $\wedge$ cierto $\Leftrightarrow$ cierto.