Práctica del túnel

Autores: Miguel Ángel Díaz Pérez, Sergio López González y Ana Midón García

El túnel de Kiyotaki es de una única dirección, pueden trascurrir por él coches que vengan de ambas direcciones. La práctica consiste en crear un sistema de semáforos que controle el tránsito del túnel, para así evitar choques entre coches que entren en direcciones contrarias.

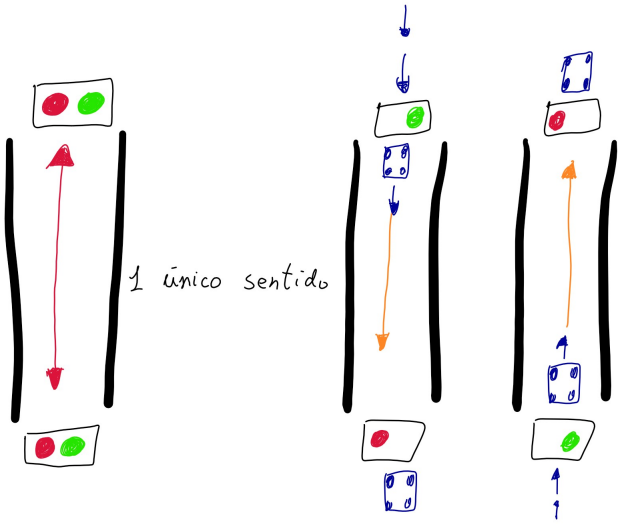

Figura 1: Idea de uso del túnel

Para conseguirlo, hemos puesto como condición para que puedan pasar de norte a sur que no haya coches pasando al norte y que no haya coches esperando a pasar desde el sur o que sea el turno de los coches provenientes del norte, y viceversa.

```
def no_cars_north(self):
    return self.inside_north.value == 0 \
           and (self.cars_north.value == 0 or self.semaphore.value == 1)

def no_cars_south(self):
    return self.inside_south.value == 0 \
           and (self.cars_south.value == 0 or self.semaphore.value == 0)
```

Figura 2: Invariante

El invariante que definimos es I={no_cars_north= 1 o no_cars_south= 1}. Este invariante se cumple en todo momento, viendo la imagen 2. *self.inside_north* se

refiere a los coches que ya están pasando de norte a sur, *self.cars_north* los coches esperando a pasar de norte a sur, y *self.semaphore* vale 1 si les toca pasar a los coches del sur, y 0 si les toca pasar a los del norte. Los valores *self.inside_south* y *self.cars_south* son análogos a los del norte. Por tanto, la función no_cars_north es verdadera si no hay coches pasando de norte a sur y no hay coches esperando o bien, es el turno de los coches en el sur.

```
def wants_enter(self, direction):
    self.mutex.acquire()
    if direction == "north":
        self.cars_north.value += 1
        self.open_north.wait_for(self.no_cars_south)
        self.inside_north.value += 1
        self.cars_north.value -= 1
    elif direction == "south":
        self.cars_south.value += 1
        self.open_south.wait_for(self.no_cars_north)
        self.inside_south.value += 1
        self.cars_south.value -= 1
    self.mutex.release()
```

Figura 3: Función *wants_enter*

Durante la realización de la práctica hemos encontrado varios inconvenientes:

- En primer lugar, nos dimos cuenta de que presentaba problemas de interbloqueo. Sin turnos, podía ocurrir que los coches a ambos lados se quedasen parados, esperando a que los del otro lado pasasen, por tanto, nunca llegaban a cruzar el túnel. Solucionamos esto implementando los turnos.
- Más tarde, observamos que existía un problema de inanición cuando los coches no son notificados de que pueden pasar, por lo que esperan indefinidamente; esto lo hemos resuelto utilizando *notify_all*.