Problema del Tunel unidireccional sin inanición

1 de abril de 2022

1. Objetivo y propuesta

El objetivo de nuestro código es evitar la **inanición** producida en la primera propuesta en la que puede ocurrir que tengamos algún coche esperando eternamente sin poder pasar porque llegan constantemente coches en la otra dirección de tal forma que nunca se vacia el tunel.

Para evitarlo nuestra propuesta es provocar una especie de 'turnos'. Cada vez que empieza un turno actualizamos la variable permiso que permite pasar un número fijo de coches (los coches que están esperando para entrar con la dirección), cuando estos coches han pasado cambiamos la dirección del tunel y repetimos lo anterior.

2. Invariante

En este caso el invariante es más complicado de expresar, nos limitamos a comprobar que es imposible que dos coches en dirección contraria esten a la vez en el tunel.

3. Inicialización

```
29  class Monitor():
30      def __init__(self):
31          self.manager = Manager()
32          self.mutex = Lock()
33          self.direction = Value('i',-1)
34          self.waiting = self.manager.list([0,0])
35          self.permission = Value('i',1)
36          self.semaphore= Condition(self.mutex)
37          self.carDir= None
38
39      def car_dir(self, car_direction):
40          self.carDir = car_direction
41
42
43      def validTunnel(self):
44          return (self.direction.value == self.carDir and self.permission.value > 0)
```

Supongamos que el primer coche 1 quiere entrar en el tunel, tenemos c1.direction = -1 (la dirección del tunel no está definida). Entramos en el condicional de la línea 52 y fija su dirección como la del tunel. Entra en el tunel y deja sin permiso al resto de coches (iniciamos con self.permission = 1, solo

pasa un coche). Hasta que este coche no salga del tunel el resto de coches no pueden entrar. Luego en la inicialización el invariante se mantiene.

4. Caso general

¿Puede haber 2 coches en el tunel con direcciones contrarias al mismo tiempo?

Supongamos que tenemos dos coches c1 y c2 al mismo tiempo en el tunel, llegaremos a la conclusión que los dos coches tienen la misma dirección. Ambos coches han entrado en el tunel por lo tanto han tenido que cumplir *ValidTunnel* y tenemos:

$$c1.carDir == c1.direction \qquad and \qquad c2.carDir == c2.direction \tag{1}$$

pero ¿puede cambiar la dirección del tunel con coches dentro?. Para ello analicemos cuando cambia la dirección:

- si $c2.direction == -1$ este caso ya se ha visto que no altera la invariante en la inicialización. También se puede dar en el caso de que al salir del tunel un coche no hay ninguno más esperando pero ese caso es idéntico a la inicialización. (además no se puede dar el caso puesto que tendríamos el coche c2 esperando)
- si $self.permission == 0$ y notificamos el cambio: Este caso solo se produce cuando ha salido del tunel el último coche con permiso. Es imposible que esto ocurra mientras que el coche c1 este en el tunel y por lo tanto es imposible que se haya cambiado la dirección entre c1 y c2 y por lo tanto

$$c1.carDir = c1.direction = c2.direction = c2.carDir \tag{2}$$

Concluimos que en cualquier caso se cumple nuestro invariante

```
46      def wants_enter(self, car_direction):
47          self.mutex.acquire()
48          self.waiting[car_direction] += 1
49          print('coches esperando' + str(self.waiting))
50
51          self.car_dir(car_direction)
52          if self.direction.value == -1:
53              self.direction.value = car_direction
54
55          #if (self.waiting[(car_direction+1)%2]==0) and self.inTunnel == 0:
56              #self.direction.value = car_direction
57
58          self.semaphore.wait_for(self.validTunnel)
59          self.waiting[car_direction] -=1
60          print('ha entrado con este permiso: ' + str(self.permission.value))
61          self.permission.value -=1
62          self.mutex.release()
```

```
64      def leaves_tunnel(self, car_direction):
65          self.mutex.acquire()
66          if self.permission.value == 0:
67              print('self.permission.value = ' + str(self.permission.value))
68              #self.empty.wait_for(self.emptyTunnel)
69              if self.waiting[(car_direction+1)%2]!=0: #hay coches en el otro lado esperando
70                  print('habia coches esperando en el otro lado, cambiamos direccíon')
71                  self.direction.value = (car_direction +1 )%2
72                  print('cambio de dirección a: ' + str(self.direction.value))
73                  self.permission.value = self.waiting[self.direction.value]
74                  print('el permiso ahora es:' + str(self.permission.value))
75              elif self.waiting[self.direction.value] == 0 and self.waiting[(self.direction.value+1)%2] == 0:
76                  self.permission.value = 1
77                  self.direction.value = -1
78                  print('no queda ningún coche esperando')
79                  print('el permiso ahora es:' + str(self.permission.value))
80              else:
81                  self.permission.value = self.waiting[self.direction.value]
82                  print('no hay coches esperando en dirección contraria, no cambiamos dirección')
83                  print('el permiso ahora es:' + str(self.permission.value))
84          self.semaphore.notify_all()
85          self.mutex.release()
```