Hay que asegurar que no crucen coches en direcciones contrarias, además de que no crucen peatones mientras haya coches en el puente.
Si lo hacemos solo con semáforos, se puede dar el caso en el cual, con un buen número de peatones, o coches en una dirección se impida que pase cualquier otra cosa por el puente. Para ello introducimos un límite de la cantidad de procesos del mismo tipo que pueden cruzar el puente a la vez, para así evitar la inanición.

I = { cars_south >= 0

1. cars_north >= 0
2. pedestrians >= 0
3. cars_south_w >= 0
4. cars_north_w >= 0
5. pedestrians_w >= 0
6. south_counter >= 0
7. north_counter >= 0
8. pedestrians_counter >= 0
9. cars_north > 0 => cars_south = 0 & pedestrians = 0
10. cars_south > 0 => cars_north = 0 & pedestrians = 0
11. pedestrians > 0 => cars_north = 0 & cars_south = 0
12. cars_south_w > 0 => north_counter < max_waitlist & pedestrians_counter < max_waitlist
13. cars_north_w > 0 => south_counter < max_waitlist & pedestrians_counter < max_waitlist
14. pedestrians_w > 0 => south_counter < max_waitlist & north_counter < max_waitlist

}
car_wants_to_cross()
pedestrian_wants_to_cross()
car_crossed()
pedestrian_crossed()

Variables del invariante:

- cars_south: La cantidad de coches que están en el puente cruzando hacia el sur
- cars_south_w: La cantidad de coches que están esperando para cruzar al sur
- counter_south: La cantidad de coches en dirección sur que ha entrado al puente sin dejar pasar a peatones o coches en direccion norte

Análogo para el cars_north

- pedestrians: La cantidad de peatones en ambas direcciones cruzando el puente
- pedestrians_w: La cantidad de peatones esperando para cruzar
- pedestrians_counter: La cantidad de peatones seguidos que ha entrado al puente sin dejar pasar coches

Condiciones 1-8: Condiciones de existencia, las variables son enteras y mayores o iguales que cero
Condiciones 9-11: Si están cruzando peatones no puede haber coches cruzando, si hay coches cruzando en cierta dirección, no puede haber coches cruzando en la contraria ni peatones.

Condiciones 12-14: Si se está esperando a que cruce un coche en cierta dirección, no pueden cruzar más de "*max_waitlist*" peatones o coches en dirección contraria seguidos, igual para el caso en el que estén esperando peatones, esto evita la inanición.
Las condiciones 1-11 se cumplen en todo momento, comprobemos que las condiciones 12-14 se cumplen a través del monitor.

CAR_WANTS_TO_CROSS()

```
    def car_wants_to_cross(self, direction):
{Cuando entramos, el invariante se cumple}
        self.mutex.acquire()
        if direction == NORTH:
            self.cars_north_w.value += 1
cars_north_w sigue siendo mayor que cero, le estamos sumando 1
            self.sem_north.wait_for(self.cars_can_go_north)
Cuando el monitor permite continuar se verifica la siguiente condición:
(self.cars_south.value == 0 and self.pedestrians.value == 0) and \
(self.counter_north.value < max_waitlist or (self.cars_south_w.value == 0\
  and self.pedestrians_w.value == 0))
Es decir, cars_south = 0, pedestrians = 0 y o bien counter_north < max_waitlist
o bien cars_south_w = 0 y pedestrians_w = 0.
Por lo tanto el coche puede empezar a cruzar el puente

            self.cars_north.value += 1
Podemos aumentar cars_north ya que cars_south = 0 y pedestrians = 0
            self.cars_north_w.value -= 1
Podemos restar 1 a cars_north_w ya que era mayor que 0
            self.counter_north.value += 1
            self.mutex.release()
```

Por tanto, al salir de car_wants_to_cross cuando el coche va en direccion norte se verifican todas las condiciones 1-8, que cars_south = 0, pedestrians = 0 y cars_north > 0, y que o bien counter_north < max_waitlist o bien cars_south_w = 0 y pedestrians_w = 0 ya que el monitor se ha abierto.
Es análogo para el sur y para los peatones.

CAR_CROSSED()

```
def car_crossed(self, direction):
{El invariante se cumple cuando entramos para las condiciones 1-8, que cars_north > 0 y
que pedestrians = 0 y cars_south = 0}
        self.mutex.acquire()
        if direction == NORTH:
            self.cars_north.value -= 1
```

Podemos disminuir cars_north porque cuando pasamos por car_crossed, cars_north > 0, ya que entramos siempre después de un car_wants_to_cross.

```
            if self.cars_north.value == 0:
                self.pedestrians_counter.value = 0
```

north_counter >= 0 se verifica, además, dejamos paso a los peatones en primera instancia

```
                if self.pedestrians_w.value == 0:
                    self.counter_south.value = 0
```

En caso de que no haya peatones esperando, dejamos pasar coches del sur, evitando el deadlock

```
            self.sem_north.notify_all()
            self.sem_south.notify_all()
            self.sem_ped.notify_all()
```

Si cars_north = 0, entonces pueden seguir pasando coches siempre y cuando pedestrians_w = 0 y cars_south = 0 o counter_north < max_waitlist, una vez acabe, deja pasar a peatones, que a su vez dejan pasar a coches del sur en primera instancia, en caso de que no haya peatones esperando, deja pasar a coches del sur. Notificamos a todos los monitores para que comprueben si cars_south_w == 0 y si pedestrians_w == 0 además de counter_north < max_waitlist.

Es análogo en dirección contraria y en el caso de pedestrian_wants_to_cross y pedestrian_crossed.