Práctica 2

Invariante

Sean p := Peatones cruzando, c0 := coches cruzando en dirección 0 y c1 := coches cruzando en dirección 1.

El invariante del monitor es

$$(c0>0 \Rightarrow p=0 \wedge c1=0) \wedge (c1>0 \Rightarrow p=0 \wedge c0=0) = I$$

De ambas condiciones se infiere $p>0 \Rightarrow c0=0 \wedge c1=0$

La fórmula es inicialmente cierta, pues p, c0 y c1 se inicializan a 0.

Consideramos las ejecuciones de las operaciones del monitor de principio a fin

- are_no_cars, are_no_cars0_and_pedestrians y are_no_cars1_and_pedestrians no alteran c0, p ni c1 y por tanto I sigue siendo cierto
- wants_enter_car: podría incrementar c0, pero en ese caso el programa se asegura (por la condición) de que p=c1=0. Análogo con c1.
- leaves_car: Se podrían reducir c0 o c1 haciéndolas cero, pero esto no falsifica I.
- wants_enter_pedestrian: Se incrementa p, pero el programa se asegura (por la condición) de que c0=c1=0.
- leaves_pedestrian: Se reduce p haciéndose, quizás, cero, pero I se seguiría cumpliendo.

Analizamos los notify

- notify en leaves_car: Podría desbloquear un peatón (p>0) o un coche en el otro sentido, pero se asegura de que esto ocurre solo si los coches en este sentido son cero.
- notify en leaves_pedestrian: Podría desbloquear un coche (c0 o c1>0) pero el programa se asegura de que esto ocurre solo si p=0 y como un peatón ha cruzado c0 y c1 también eran cero.

Puente seguro.

Es consecuencia del invariante.

Los casos no seguros son $c0>0 \wedge c1>0$ y $(p>0 \wedge c0>0) \vee (p>0 \wedge c1>0)$, pero ambos contradicen el invariante

Deadlocks

Habría un deadlock si todos los peatones están esperando en no-cars, todos los c0 en no-cars1-no-pedestrians y todos los c1 en no-cars0-no-pedestrians. Sin embargo, si nadie está cruzando p=c0=c1=0 y self.turn.value $\in \{0,1,2\}$, por tanto, una de las 3 condiciones se cumple y algún proceso se desbloquea.

Inanición

Si un peatón sufre inanición es porque se queda esperando a no-cars. Suponga-mos que es porque c0>0. Por progreso en los coches todos acabarán ejecutando leaves_car, cuando el primero de ellos lo haga self.turn.value será 1 y cuando el último de ellos lo haga habrá notify(no_cars) y notify(no_cars0_no_pedestrians). Si no hay c1 esperando esto desbloquea a un peatón (y por inducción acabará desbloqueando a todos), en caso contrario desbloquea a un c1. Repitiendo el razonamiento llegamos a la ejecución de notify(no_cars) y self.turn.value=2, lo cual desbloquea un (y eventualmente todo) peatón. Si suponemos c1>0 estamos en el 2º caso que acabamos de demostrar.

Si un coche c0 sufre inanición es porque se queda esperando a no_cars1_no_pedestrians. Sea c1>0, por progreso en los coches todos acaban ejecutando leaves_car, cuando el primero lo haga self.turn.value será 2 y cuando el último lo haga habrá notify(no_cars) y notify(no_cars1_no_pedestrians). Si no hay peatones esperando esto desbloquea un coche c0 (y eventualmente todos), en caso contrario desbloquea un c1. Repitiendo el razonamiento para p>0 llegamos a self.turn.value=0, notify(no_cars) y notify(no_cars0_no_pedestrians) y notify(no_cars1_no_pedestrians), lo cual esta vez sí desbloquea un c0 (y even-tualmente todos). Si p>0 estamos en el caso recién demostrado.

La demostración es análoga para la inanición de coches c1.