Problema del puente de Ambite

LUCÍA ROLDÁN RODRÍGUEZ

March 2023

1 Invariante del Monitor

1. $cn \geq 0, cs \geq 0, ped \geq 0, cn\_waiting \geq 0, cs\_waiting \geq 0, ped\_waiting \geq 0, 0 \leq turno \leq 2$
2. $cn > 0 \Rightarrow cs = 0$ and $ped = 0$
3. $cs > 0 \Rightarrow cn = 0$ and $ped = 0$
4. $ped > 0 \Rightarrow cn = 0$ and $cs = 0$
5. $turno = 0$ and $cn = 0$ and $cs = 0$ and $ped = 0 \Rightarrow cn\_waiting > 0$ or (cs_waiting = 0 and ped_waiting = 0)
6. $turno = 1$ and $cn = 0$ and $cs = 0$ and $ped = 0 \Rightarrow cs\_waiting > 0$ or (cn_waiting = 0 and ped_waiting = 0)
7. $turno = 2$ and $cn = 0$ and $cs = 0$ and $ped = 0 \Rightarrow ped\_waiting > 0$ or (cn_waiting = 0 and cs_waiting = 0)

2 Demostración del Invariante

Veamos que las condiciones anteriores son invariantes del monitor.

Al inicializar el mointor cn = 0, cs = 0, ped = 0, cn_waiting = 0, cd_waiting = 0, ped_waiting = 0 y turno = 0 $\Rightarrow$ se cumplen las 7 condiciones del invariante. Las variables se encuentran en los rangos marcados por 1,por otro lado, 2,3 y 4 se cumplen trivialmente puesto que no se cumplen los

condicionales y 5,6 y 7 se cumple puesto que cn_waiting = 0, cd_waiting = 0, ped_waiting = 0.

Ahora debemos comprobar, suponiendo cierto que cuando una función del monitor se empieza a ejecutar se cumple el invariante, que se sigue cumpliendo cuando la función acaba o cuando se encuentra bloqueada puesto que abandonará el monitor dando paso a que otro proceso ejecute otra función del mismo. Además se debe verificar que se cumple el invariante antes de llamar a algún notify. Recorramos todas las funciones comprobándolo:

- wants_enter_car(0) cumple al salir INV y $cn > 0$. Como la condición para salir del wait cs == 0 and ped == 0, la única condición del invariante que se podría violar que es la 2, nos aseguramos que se cumpla.
  Por otra parte veamos cuándo se puede bloquear la función en el wait y que se cumple el invariante en el caso de bloquearse. Al entrar en el wait $cn\_waiting > 0$ por lo que las únicas condiciones que podría no estar cumpliendo el invariante son la 5,6 y 7.
  - Si se bloquea porque $cs > 0$ o $ped > 0$ no estaría incumpliendo ninguna de ellas.
  - Si cs = 0 y ped = 0 :
    - si turno = 1 y $cs\_waiting > 0$ se cumplen las condiciones
    - si turno = 1 y cs_waiting =0 no se bloquea el wait puesto que al entrar en la función el invariante asegura ped_waiting = 0 y esa variable no ha sido modificada.
    - si turno = 2 y $ped\_waiting > 0$ se cumplen las condiciones
    - si turno = 2 y ped_waiting =0 no se bloquea el wait puesto que al entrar en la función el invariante asegura cs_waiting = 0 y esa variable no ha sido modificada.
    - si turno = 0 no se bloquea el wait.
- leaves_car(0) siempre se ejecuta justo a continuación de wants_enter_car(0) por tanto al empezarse a ejecutar cumple INV y $cn > 0$. Con ello nos aseguramos que al reducir en uno cn, este siga estando dentro del rango de la primera condición del invariante. Por otra parte podría darse cn = 0, cs = 0 y ped = 0 al salir de esta función por tanto veamos que ocurre con las condiciones 5,6,y 7 del invariante. La función comprueba

primeramente si $cs\_waiting > 0$ de serlo asigna turno = 1 con lo que cumpliría las condiciones. De darse el caso $cs\_waiting = 0$ se comprueba si $ped\_waiting > 0$ en cuyo caso se actualiza el turno = 2 con lo que cumpliría las condiciones. Finalmente si $cs\_waiting = 0$ y ped_waiting = 0 se actualiza turno = 0 con lo que se cumplen las condiciones. Podemos asegurar de esta forma que se cumple el invariante al salir de leaves_car(0) y por tanto también antes de ejecutar cualquier notify dentro de esta función pues es la última orden que se ejecuta.

- wants_enter_car(1) cumple al salir INV y $cs > 0$. Como la condición para salir del wait cn == 0 and ped == 0 , la única condición del invariante que se podría violar, que es la 3, nos aseguramos que se cumpla.
  Por otra parte veamos cuándo se puede bloquear la función en el wait y que se cumple el invariante en el caso de bloquearse. Al entrar en el wait $cs\_waiting > 0$ por lo que las únicas condiciones que podría no estar cumpliendo el invariante son la 5,6 y 7.
  - Si se bloquea porque $cs > 0$ o $ped > 0$ no estaría incumpliendo ninguna de ellas.
  - Si cn = 0 y ped = 0:
    - si turno = 0 y $cn\_waiting > 0$ se cumplen las condiciones.
    - si turno = 0 y cn_waiting = 0 no se bloquea el wait puesto que al entrar en la función, el invariante asegura ped_waiting = 0 y esa variable no ha sido modificada.
    - si turno = 2 y $ped\_waiting > 0$ se cumplen las condiciones.
    - si turno = 2 y ped_waiting = 0 no se bloquea el wait puesto que al entrar en la función el invariante asegura cn_waiting = 0 y esa variable no ha sido modificada.
    - si turno = 1 no se bloquea el wait.

- leaves_car(1) siempre se ejecuta justo a continuación de wants_enter_car(1) por tanto al empezarse a ejecutar cumple INV y $cs > 0$. Con ello nos aseguramos que al reducir en uno cs, este siga estando dentro del rango de la primera condición del invariante. Por otra parte podría darse cn = 0, cs = 0 y ped = 0 al salir de esta función por tanto veamos que ocurre con las condiciones 5,6,y 7 del invariante. La función comprueba primeramente si $ped\_waiting > 0$ de serlo asigna turno = 2 con

lo que cumpliría las condiciones. De darse el caso $ped\_waiting = 0$ se comprueba si $cn\_waiting > 0$ en cuyo caso se actualiza el turno = 0 con lo que cumpliría las condiciones. Finalmente si $cn\_waiting = 0$ y ped_waiting = 0 se actualiza turno = 1 con lo que se cumplen las condiciones. Podemos asegurar de esta forma que se cumple el invariante al salir de leaves_car(1) y por tanto también antes de ejecutar cualquier notify dentro de esta función pues es la última orden que se ejecuta.

- wants_enter_pedestrian() cumple al salir INV y $ped > 0$. Como la condición para salir del wait cs == 0 and cn== 0, la única condición del invariante que se podría violar, que es la 2, nos aseguramos que se cumpla.
  Por otra parte veamos cuándo se puede bloquear la función en el wait y que se cumple el invariante en el caso de bloquearse. Al entrar en el wait $ped\_waiting > 0$ por lo que las únicas condiciones que podría no estar cumpliendo el invariante son la 5,6 y 7.
  - Si se bloquea porque $cs > 0$ o $cn > 0$ no estaría incumpliendo ninguna de ellas.
  - Si cs = 0 y cn = 0:
    - si turno = 1 y $cs\_waiting > 0$ se cumplen las condiciones
    - si turno = 1 y cs_waiting = 0 no se bloquea el wait puesto que al entrar en la función el invariante asegura cn_waiting = 0 y esa variable no ha sido modificada.
    - si turno = 0 y $cn\_waiting > 0$ se cumplen las condiciones
    - si turno = 0 y cn_waiting = 0 no se bloquea el wait puesto que al entrar en la función el invariante asegura cs_waiting = 0 y esa variable no ha sido modificada.
    - si turno = 2 no se bloquea el wait.

- leaves_pedestrian() siempre se ejecuta justo a continuación de wants_enter _pedestrian() por tanto al empezarse a ejecutar cumple INV y $ped > 0$. Con ello nos aseguramos que al reducir en uno ped, este siga estando dentro del rango de la primera condición del invariante. Por otra parte podría darse cn = 0, cs = 0 y ped = 0 al salir de esta función por tanto veamos que ocurre con las condiciones 5,6,y 7 del invariante. La función comprueba primeramente si $cn\_waiting > 0$ de serlo asigna turno = 0

con lo que cumpliría las condiciones. De darse el caso $cn\_waiting = 0$ se comprueba si $cs\_waiting > 0$ en cuyo caso se actualiza el turno = 1 con lo que cumpliría las condiciones. Finalmente si $cn\_waiting = 0$ y cs_waiting = 0 se actualiza turno = 2 con lo que se cumplen las condiciones. Podemos asegurar de esta forma que se cumple el invariante al salir de leaves_pedestrian y por tanto también antes de ejecutar cualquier notify dentro de esta función pues es la última orden que se ejecuta.

- carnorth_mayenter(), carsouth_mayenter(), pedestrian_mayenter() son funciones que no modifican las variables, ni tienen funciones wait ni signify por tanto se cumplirá el invariante.

3 Seguridad del Puente

Para demostrar la seguridad del puente comprobaremos que no pueden pasar peatones a la vez que coches y que no pueden pasar simultáneamente coches de sentidos opuestos:

- Si un coche quiere entrar en dirección norte antes de aumentar en uno el número de coches en sentido norte y darle paso a cruzar, la función wants_enter_car(0) comprueba entre otras cosas que cs == 0 y ped == 0 $\Rightarrow$ Un coche en dirección norte sólo puede entrar si el puente está vacío o hay coches pasando en la misma dirección.

- Si un coche quiere entrar en dirección sur antes de aumentar en uno el número de coches en sentido sur y darle paso a cruzar, la función wants_enter_car(1) comprueba entre otras cosas que cn == 0 y ped == 0 $\Rightarrow$ Un coche en dirección sur sólo puede entrar si el puente está vacío o hay coches pasando en la misma dirección.

- Si un peatón quiere entrar en el puente, antes de aumentar en uno el número de peatones y darle paso a cruzar, la función wants_enter_ped comprueba entre otras cosas que cn == 0 y cs == 0 $\Rightarrow$ Un peaton sólo puede entrar si en el puente no hay coches, es decir , si el puente está vacío o hay otros peatones cruzando.

Concluyendo así que nunca hay coches y peatones a la vez en el puente, así como no hay simultáneamente coches de sentidos opuestos.

4 Ausencia de deadlocks

Veamos que no se pueden encontrar bloqueados a la vez los coches en sentido norte, los coches en sentido sur y los peatones

Supongamos que los peatones al ejecutar wants_enter_pedestrian se encuentran bloqueados en nocars_waitfor(pedestrian_mayenter) $\Rightarrow$ $ped\_waiting > 0$ and ($cn > 0$ or $cs > 0$ or ($turno \neq 2$ and ($cn\_waiting > 0$ or $cs\_waiting > 0$). Analicemos los diferentes casos en los que se puede bloquear nocars_waitfor(pedestrian_mayenter):

- $cn > 0$ or $cs > 0$ alguno de los sentidos se encuentra activo.
- $cn = 0$ and $cs = 0$ and ($turno \neq 2$ and ($cn\_waiting > 0$ or $cs\_waiting > 0$):
  - ped $\neq 0$ $\Rightarrow$ se consumirán en un tiempo finito sin poder entrar más peatones puesto que se encontrarían bloqueados por hipótesis hasta que se actualiza $ped = 0$ derivándose en el siguiente caso:
  - ped = 0
    - $turno = 1$ $\Rightarrow$ $cs\_waiting > 0$ puesto que (cs = 0 y cn = 0 y ped = 0 y $ped\_waiting > 0$) y por el invariante no puede bloquearse un proceso en este estado si cs_waiting = 0 . Concluyendo que si $turno = 1$ y $cs\_waiting > 0$ entonces la segunda rama de wants_enter_car no se encuentra bloqueada puesto que ped = 0, $cn = 0$, $turno = 1$ y $cs\_waiting > 0$ dando paso a los coches en dirección sur.
    - $turno = 0$ $\Rightarrow$ $cn\_waiting > 0$ puesto que (cs = 0 y cn = 0 y ped = 0 y $ped\_waiting > 0$) y por el invariante no puede bloquearse un proceso en este estado si cn_waiting = 0 . Concluyendo que si $turno = 0$ y $cn\_waiting > 0$ entonces la primera rama de wants_enter_car no se encuentra bloqueada puesto que ped = 0, $cn = 0$, $turno = 0$ y $cn\_waiting > 0$ dando paso a los coches en dirección norte.

Supongamos que los coches en sentido norte al ejecutar wants_enter_car(0) se encuentran bloqueados en nocs_noped_waitfor(carnorth_mayenter) $\Rightarrow$ $cn\_waiting > 0$ and ($ped > 0$ or $cs > 0$ or ($turno \neq 0$ and ($cs\_waiting > 0$ or $ped\_waiting > 0$). Analicemos los diferentes casos en los que se puede bloquear nocs_noped_waitfor (carnorth_mayenter):

- $ped > 0$ or $cs > 0$ alguno de las dos opciones se encuentra activa.
- $ped = 0$ and $cs = 0$ and ($turno \neq 0$ and ($ped\_waiting > 0$ or $cs\_waiting > 0$):
  - cn $\neq 0 \Rightarrow$ se consumirán en un tiempo finito sin poder entrar más coches en sentido norte puesto que se encontrarían bloqueados por hipótesis hasta que se actualiza $cn = 0$ derivándose en el siguiente caso:
  - cn = 0
    - $turno = 1 \Rightarrow cs\_waiting > 0$ puesto que (cs = 0, cn = 0, ped = 0 y $cn\_waiting > 0$) y por el invariante no puede bloquearse un proceso en este estado si cs_waiting = 0 . Concluyendo que si $turno = 1$ y $cs\_waiting > 0$ entonces la segunda rama de wants_enter_car no se encuentra bloqueada puesto que ped = 0, $cn = 0$, $turno = 1$ y $cs\_waiting > 0$ dando paso a los coches en dirección sur.
    - $turno = 2 \Rightarrow ped\_waiting > 0$ puesto que (cs = 0 y cn = 0, ped = 0 y $cn\_waiting > 0$) y por el invariante no puede bloquearse un proceso en este estado si ped_waiting = 0 . Concluyendo que si $turno = 2$ y $ped\_waiting > 0$ entonces de wants_enter_ped no se encuentra bloqueada puesto que ped = 0, $cn = 0$, $turno = 2$ y $cn\_waiting > 0$ dando paso a peatones.

Supongamos que los coche en sentido sur al ejecutar wants_enter_car(1) se encuentran bloqueados en nocs_noped_waitfor(carsouth_mayenter) $\Rightarrow cs\_waiting > 0$ and ($ped > 0$ or $cn > 0$ or ($turno \neq 1$ and ($cn\_waiting > 0$ or $ped\_waiting > 0$). Analicemos los diferentes casos en los que se puede bloquear nocs_noped_waitfor (carsouth_may_enter):

- $ped > 0$ or $cn > 0$ alguno de las dos opciones se encuentra activa.
- $ped = 0$ and $cn = 0$ and ($turno \neq 1$ and ($ped\_waiting > 0$ or $cs\_waiting > 0$):
  - cs $\neq 0 \Rightarrow$ se consumirán en un tiempo finito sin poder entrar más coches en sentido norte puesto que se encontrarían bloqueados por hipótesis hasta que se actualiza $cs = 0$ derivándose en el siguiente caso:

- cs = 0
  - $turno = 0 \Rightarrow cn\_waiting > 0$ puesto que (cs = 0 y cn = 0 y ped = 0 y $cs\_waiting > 0$) y por el invariante no puede bloquearse un proceso en este estado si cn_waiting = 0 . Concluyendo que si $turno = 0$ y $cn\_waiting > 0$ entonces la primera rama de wants_enter_car no se encuentra bloqueada puesto que ped = 0, $cs = 0$, $turno = 0$ y $cn\_waiting > 0$ dando paso a los coches en dirección norte.
  - $turno = 2 \Rightarrow ped\_waiting > 0$ puesto que (cs = 0 y cn = 0 y ped = 0 y $cn\_waiting > 0$) y por el invariante no puede bloquearse un proceso en este estado si ped_waiting = 0 . Concluyendo que si $turno = 2$ y $ped\_waiting > 0$ entonces de wants_enter_ped no se encuentra bloqueada puesto que ped = 0, $cn = 0$, $turno = 2$ y $cn\_waiting > 0$ dando paso a peatones.

5 Ausencia de inanición

- Ausencia de inanición en los peatones: Veamos que el peaton no se queda indefinidamente bloqueado en nocars.wait_for. Supongamos que un peaton se encuentra bloqueado veamos que eventualmente se llama a un nocars.notify_all() que será capaz de dar paso a un peaton esperando y por hipótesis de justicia dará paso a todos los peatones en algún momento.

  - si $ped > 0$ or $cn > 0$ or $cs > 0$ se terminarán de consumir los coches o peatones existentes fijando el turno correspondiente 0,1 o 2 y actualizando,tras el paso del último coche o peatón ped = 0 and cn = 0 and cs = 0 lo que llevará al siguiente subapartado.
  - si ped = 0 and cn = 0 and cs = 0
    - si turno = 2, hay dos opciones o el proceso no se encontraba bloqueado o tras actualizar el turno a 2 y consumirse todos los coches que había en la carretera se ha hecho un nocar_notify_all(), en cualquier caso dará paso a algún peatón esperando (puesto que cs = 0, cn = 0 y turno = 2).
    - si el $turno = 1 \Rightarrow cs\_waiting > 0$(puesto que $ped\_waiting > 0$), hay dos opciones o el proceso no se encontraba bloqueado

o tras actualizar el turno a 1 y consumirse todos los coches que había en la carretera se ha hecho un nocn_noped_notify_all(),en cualquier caso algunos de los coches que se encontraban esperando en sentido sur pasarán y tras ello actualizarán el turno a 2 y cs= 0 haciendo un nocar_notify_all() dando paso a algún peatón esperando(puesto que cs = 0, cn = 0 y turno = 2).

* si el $turno = 0 \Rightarrow cn\_waiting > 0$(puesto que $ped\_waiting > 0$) algunos de los cuales pasarán y tras ello actualizarán cn = 0 y el turno a 1, en caso de existir coches esperando en sentido sur y haciendo un nocs_noped_notify_all(), se llevará a cabo el proceso del apartado anterior. En caso de no haber coches esperando en sentido sur, se actualizará el turno a 2 y haciendo un nocar_notify_all() dará paso a algún peatón esperando (puesto que cn = 0, cs = 0 y turno = 2) .

Cabe destacar que en este apartado siempre que se da paso a coches en alguno de los dos sentidos no se quedan indefinidamente dentro de la carretera coches de un mismo sentido:

- si $cs > 0$ en cuanto pase el primero (que lo hará en un tiempo finito) el turno se actualizará a turno = 2 puesto que $ped\_waiting > 0$, lo que evitará que más coches entren en sentido sur pues nocn_noped se encontrará bloqueado puesto que el turno no es 1 y $ped\_waiting > 0$.
- si $cn > 0$ en cuanto pase el primero (que lo hará en un tiempo finito) el turno se actualizará a turno = 1, si hay coches en sentido sur esperando, lo que evitará que más coches entren en sentido norte pues nocs_noped se encontrará bloqueado puesto que el turno no es 0 y $cs\_waiting > 0$. En caso de no existir coches en sentido sur esperando el turno se actualizará a turno = 2 puesto que $ped\_waiting > 0$, lo que evitará que más coches entren en sentido norte pues nocs_noped se encontrará bloqueado puesto que el turno no es 0 y $ped\_waiting > 0$.

- Ausencia de inanición en coches con sentido norte : Veamos que un coche en sentido norte no se queda indefinidamente bloqueado en nocs_noped.wait_for. Supongamos que un coche en sentido norte se encuentra bloqueado

veamos que eventualmente se llama a un nocs_noped.notify_all() que será capaz de dar paso a un coche esperando en sentido norte y por hipótesis de justicia dará paso a todos en algún momento.

- si $ped > 0$ or $cn > 0$ or $cs > 0$ se terminarán de consumir los coches o peatones existentes fijando el turno correspondiente 0,1 o 2 y actualizando,tras el paso del último coche o peatón ped = 0 and cn = 0 and cs = 0 lo que llevará al siguiente subapartado.
- si ped = 0 and cn = 0 and cs = 0
  - si turno = 0, hay dos opciones o el proceso no se encontraba bloqueado o tras actualizar el turno a 0 y consumirse todos los coches o peatones que había en eel puente se ejecuta un nocs_noped_notify_all(), en cualquier caso dará paso a algún coche esperando en sentido norte (puesto que cs = 0, ped = 0 y turno = 0).
  - si el $turno = 2 \Rightarrow ped\_waiting > 0$(puesto que $cn\_waiting > 0$), hay dos opciones o el proceso no se encontraba bloqueado o tras actualizar el turno a 2 y consumirse todos los coches o peatones que había en el puente se ha hecho un nocars_notify_all(), en cualquier caso algunos de los peatones que se encontraban esperando pasarán y tras ello actualizarán el turno a 0 y ped= 0, se ejecuta un nocs_noped_notify_all() dando paso a algún coche esperando en sentido norte(puesto que cs = 0, ped = 0 y turno = 0).
  - si el $turno = 1 \Rightarrow cs\_waiting > 0$(puesto que $cn\_waiting > 0$) algunos de los cuales pasarán y tras ello actualizarán cs = 0 y el turno a 2, en caso de existir peatones esperando y haciendo un nocars_notify_all() se llevará a cabo el proceso del apartado anterior. En caso de no haber peatones esperando se actualizará el turno a 0 y haciendo un nocs_noped_notify_all() dará paso a algún coche esperando en sentido norte(puesto que ped = 0, cs = 0 y turno = 0) .

Cabe destacar que en este apartado siempre que se da paso a coches en sentido sur o peatones no se quedan indefinidamente dentro :

- si $ped > 0$ en cuanto pase el primero (que lo hará en un tiempo

finito) el turno se actualizará a turno = 0 puesto que $cn\_waiting > 0$, lo que evitará que más peatones entren nocars.wait se encontrará bloqueado puesto que el turno no es 2 y $cn\_waiting > 0$.

- si $cs > 0$en cuanto pase el primero (que lo hará en un tiempo finito) el turno se actualizará a turno = 2, si hay peatones esperando, lo que evitará que más coches entren en sentido sur pues nocn_noped.wait no permitirá el paso puesto que el turno no es 1 y $ped\_waiting > 0$. En caso de no existir peatones esperando el turno se actualizará a turno = 0 puesto que $cn\_waiting > 0$, lo que evitará que más coches entren en sentido sur pues nocn_noped.wait no dejará paso puesto que el turno no es 1 y $cn\_waiting > 0$.

- Ausencia de inanición en coches con sentido sur : Veamos que un coche en sentido sur no se queda indefinidamente bloqueado en nocn_noped.wait_for. Supongamos que un coche en sentido sur se encuentra bloqueado veamos que eventualmente se llama a un nocn_noped.notify_all() que será capaz de dar paso a un coche esperando en sentido sur y por hipótesis de justicia dará paso a todos en algún momento.

  - si $ped > 0$ or $cn > 0$ or $cs > 0$ se terminarán de consumir los coches o peatones existentes fijando el turno correspondiente 0,1 o 2 y actualizando,tras el paso del último coche o peatón ped = 0 and cn = 0 and cs = 0 lo que llevará al siguiente subapartado.
  - si ped = 0 and cn = 0 and cs = 0
    - si turno = 1, hay dos opciones o el proceso no se encontraba bloqueado o tras actualizar el turno a 1 y consumirse todos los coches o peatones que había en el puente se ejecuta un nocn_noped_notify_all(), en cualquier caso dará paso a algún coche esperando en sentido sur (puesto que ped = 0, cn = 0 y turno = 1).
    - si el $turno = 0 \Rightarrow cn\_waiting > 0$(puesto que $cs\_waiting > 0$), hay dos opciones o el proceso no se encontraba bloqueado o tras actualizar el turno a 0 y consumirse todos los coches o peatones que había en el puente se ha hecho un nocs_noped_notify_all(),en cualquier caso algunos de los coches que se encontraban esperando en sentido norte pasarán y tras ello actualizarán el

turno a 1 y cn= 0, se ejecuta un nocn_noped_notify_all() dando paso a algún coche esperando en sentido sur(puesto que cn = 0, ped = 0 y turno = 1).

* si el $turno = 2 \Rightarrow ped\_waiting > 0$(puesto que $cs\_waiting > 0$) algunos de los cuales pasarán y tras ello actualizarán ped = 0 y el turno a 0, en caso de existir coches esperando en sentido norte y haciendo un nocs_noped_notify_all() se llevará a cabo el proceso del apartado anterior. En caso de no haber coches esperando en sentido norte se actualizará el turno a 1 y haciendo un nocn_noped_notify_all() dará paso a algún coche esperando en sentido sur(puesto que ped = 0, cn = 0 y turno = 1) .

Cabe destacar que en este apartado siempre que se da paso a coches en sentido norte o peatones no se quedan indefinidamente dentro :

- si $cn > 0$ en cuanto pase el primero (que lo hará en un tiempo finito) el turno se actualizará a turno = 1 puesto que $cs\_waiting > 0$, lo que evitará que más coches en sentido norte entren ya que nocs_noped.wait no permitirá el paso puesto que el turno no es 0 y $cs\_waiting > 0$.
- si $ped > 0$ en cuanto pase el primero (que lo hará en un tiempo finito) el turno se actualizará a turno = 0, si hay coches esperando en sentido norte, lo que evitará que más peatones entren pues nocars.wait no permitirá el paso puesto que el turno no es 2 y $cn\_waiting > 0$. En caso de no existir coches en sentido norte esperando el turno se actualizará a turno = 1 puesto que $cs\_waiting > 0$, lo que evitará que más peatones entren ya que nocars.wait no les dejará paso puesto que el turno no es 2 y $cs\_waiting > 0$.