

① Pseudocódigo con la solución al problema del puente.

Monitor Puente

N: int = 0 ;

S: int = 0 ;

P: int = 0 ;

EspN: int = 0 ;

EspS: int = 0 ;

EspP: int = 0 ;

hayDentroN: Condition

hayDentroS: Condition

hayDentroP: Condition

medenPasonN: Condition

medenPasonS: Condition

medenPasonP: Condition

ColaN: int

ColaS: int

ColaP: int

- `Wants_Anter_Cars(dir)`

`If dir = North:`

`EspN = EspN + 1;`

`medenPasonN.Wait(EspS <= ColaS ^ EspP <= ColaP);`

`hayDentroN.Wait(S == 0 ^ P == 0);`

`EspN = EspN - 1;`

`N = N + 1;`

`If EspN <= ColaN:`

`hayDentroS.Notify();`

`hayDentroP.Notify();`

`If dir = South:`

`(analog)`

- `Leaves_Car(dir)`

`If dir = North:`

`N = N - 1;`

`If N == 0:`

`hayDentroS.Notify();`

`hayDentroP.Notify();`

`else:`

`(analog).`

- `Wants_antia_Nedestrism()`:

`EspP = Esp + 1`

`medenPasonP.Wait(EspN <= ColaN ^ EspS <= ColaS);`

`hayDentroP.Wait(N == 0 ^ S == 0)`

`EspP = EspP - 1;`

`P = P + 1;`

`If EspP <= ColaP`

`hayDentroN.Notify();`

`hayDentroS.Notify();`

- `Leaves_Nedestrism()`:

`P = P - 1`

`If P == 0:`

`hayDentroN.Notify();`

`hayDentroS.Notify();`

② Invariante del puente:

El puente es compartido por coches y peatones $\Rightarrow \begin{cases} N \geq 0 & \wedge \text{Esp}N \geq 0 \\ S \geq 0 & \wedge \text{Esp}S \geq 0 \\ P \geq 0 & \wedge \text{Esp}P \geq 0 \end{cases}$

- No pueden pasar cochilos en ambos sentidos a la vez
- y los peatones No pueden compartir puente con los coches, \Rightarrow

$\Rightarrow \begin{cases} \text{Si } N > 0 \rightarrow S = 0 \wedge P = 0 \\ \text{Si } S > 0 \rightarrow N = 0 \wedge P = 0 \\ \text{Si } P > 0 \rightarrow N = 0 \wedge S = 0. \end{cases}$

Estas 9 condiciones forman el invariante.

③ El puente es seguro:

Para comprobar que el puente es seguro, tenemos que comprobar que se cumplen las normas en todo momento. Es decir, se cumple el invariante durante la ejecución.

\rightarrow Inicialmente, el puente está vacío $\Rightarrow N = S = P = 0$

\rightarrow Inicialmente se cumple que $\text{Esp}N \geq 0$, $\text{Esp}S \geq 0$ y $\text{Esp}P \geq 0$. Y puesto que no hay coches ni peatones en el puente, ninguna de estos 3 grupos comparte puente con otro grupo. \square

\rightarrow Tenemos que ver ahora si, suponiendo que se cumple el invariante ante de cada operación, al terminar se sigue cumpliendo.

En el caso de los coches, saldrá hora en el caso en el que los coches vienen del Norte, pues si vienen del Sur, solo hay que cambiar la variable N por S , (esta N por $Coches$)

• Mots. entre - cas (dn):

cuando un coche del norte llega al puente, se tiene a esperar hasta que se cumpla la condición de que $S = 0 \wedge P = 0$. (después de comprobar si pueden pasar, que se anotaría en el apartado de ejecución).

luego se cumple el enunciante. Como inicialmente también se cumple, tenemos demostrada la seguridad del puente.

④ Absencia de deadlocks

Un deadlock hace que el programa se quede bloqueado y no pueda avanzar de ningún modo. Vamos a ver que esto no sucede en el puente. En el apartado ③ hemos probado que o bien el puente es clásico inicialmente, o bien contiene Coches del Norte, o bien del sur, o bien peatones. Para garantizar que no hay deadlocks, tenemos que las variables Condición `hayDentroN`, `hayDentroS` y `hayDentroP`, siempre van a haber al menos 1 que es cierta para cada situación del puente.

- a) Inicialmente, $N=S=P=0$, luego las 3 condiciones se cumplen y darán para que alguno de los 3 grupos se meta al puente.
- b) Si hay Coches del Norte en el puente ($N > 0$), el enunciante nos dice que $S = P = 0$, luego `hayDentroN` sera verdadera. Cuando el último coche del Norte haya salido, tenemos que $N = 0$, y al notificar a `hayDentroS` y `hayDentroP`, se revuelven ambas verdades permitiendo pasar a uno de los dos grupos restantes.
- c) El caso en el que hay Coches del Sur en el puente ($S > 0$) es análogo a que haya Coches del Norte ($N > 0$).
- d) Si hay peatones en el puente ($P > 0$), el enunciante implica que no hay coches dentro del puente ($N = S = 0$), por lo que `hayDentroP` es cierta. Cuando salga el último peatón ($P = 0$), se notificará a `hayDentroN` y `hayDentroS` de que ya no hay peatones en el puente, haciendo las ciertas y podrán pasar o bien los coches del Norte o bien los del Sur.

\Rightarrow Como alguna condición siempre es cierta, no hay deadlocks.

Justo antes se ha comprobado que el número de coches y de peatones esperando es menor que la cota ($\text{EspS} \leq \text{CotaS} \wedge \text{EspP} \leq \text{CotaP}$). Luego, como sabemos que, después del cruce, $S=P=0 \Rightarrow$ No hay coches del sur ni peatones en el puente, luego es seguro.

- **Llaves - Con (dn):**

Cuando el coche del norte quiere salir del puente ($N > 0$), N se baja en 1 unidad, luego tenemos que, actualizando N , $N \geq 0$. Si $N > 0$, no se hace nada pues todavía hay coches del norte en el puente, pero si $N = 0$, se notifica a los coches del sur y los peatones para que uno de los dos entre al puente, pero no ambos. Luego es seguro.

- **Wants - ante pedestrán ():**

Cuando un peatón llega al puente, se pone a esperar y se comprueba que $\text{EspN} \leq \text{CotaN} \wedge \text{EspS} \leq \text{CotaS}$. Cuando se cumpla, se le da espera hasta que no haya coches en el puente ($S=0 \wedge N=0$). Luego, como después de esperar sabemos que $S=0 \wedge N=0$, pueden entrar los peatones, haciendo $P > 0$. Luego se cumple el invarianto y el puente es seguro.

- **Llaves - pedestrán ():**

Cuando un peatón quiere salir del puente ($P > 0$), P se baja en 1 unidad y, actualizando la clonable, $P \geq 0$ cuando sale 1 peatón. Si $P > 0$, se hace nada, pues todavía hay peatones en el puente. Si $P = 0$, se notifica a ambos grupos de coches para que uno de los dos entre al puente, pero no ambos. Luego el puente es seguro.

Después de ejecutar cada operación llegamos a que el puente es seguro.

⑤ Ausencia de finalización

La finalización ocurre cuando en proceso nunca llegue a hacer nada (no pueda ejecutarse) porque el programa o el resto de procesos hagan que no se pueda ejecutar nunca. Es decir, dicho proceso espera infinitamente para poder ejecutarse.

Para saber que no hay finalización, tenemos que ver que en cualquier momento, los 3 grupos que tiene nuestro problema da a pasar por el puente como mínimo 1 vez.

Para garantizarlo, hacemos uso de las clavables CotaV, CotaS y CotaP, y veremos que no hay finalización para ningún grupo.

a) Coches Norte: Cuando un coche del norte llega al puente, se pone a esperar y se aumenta EspN en 1 unidad. Este proceso continua y siempre que tengamos que $\text{EspN} \leq \text{CotaN}$, esto implica que tanto los coches del sur como los peatones podrán pasar (se cumplen a ciertas las clavables $\text{puedanPasN} \wedge \text{puedanPasP}$) al puente. Sin embargo, cuando $\text{EspN} > \text{CotaN}$, estas condiciones no se cumplen a ciertas y hace que los coches del norte tengan preferencia de paso, haciendo que pasen al puente \Rightarrow No hay finalización.

b) Coches Sur: Análogo a Coches del Norte \Rightarrow no hay finalización.

c) Peatones: Cuando un peatón llega al puente, se pone a esperar y se aumenta el valor de la variable EspP en 1 unidad. Este proceso continua mientras sigan llegando peatones y, mientras tengamos $\text{EspP} \leq \text{CotaP}$, se cumplen a ciertas las condiciones $\text{puedanPasN} \wedge \text{puedanPasS}$. Cuando $\text{EspP} > \text{CotaP}$, se les da prioridad a los peatones para que pasen y acaban pasando \Rightarrow No hay finalización