MODULE *example_2*

EXTENDS *Sequences*, *TLC*

--algorithm *telephone*

variables

to_send = ⟨1, 2, 3⟩, se envian los mensajes 1, 2, 3; en ese orden.
received = ⟨⟩, secuencia de mensajes recibidos.
in_transit = {}, buffer donde se guardan los mensajes.
can_send = TRUE ; define una bandera para asegurar que el mensaje fue recibido.

begin

while *Len*(*received*) ≠ 3 **do**
send y receiver: se realizan concurrentemente

send:
if *can_send* ∧ *to_send* ≠ ⟨⟩ **then**
in_transit := *in_transit* ∪ {*Head*(*to_send*)} ;
to_send := *Tail*(*to_send*) ;
can_send := FALSE ;
end if ;

receiver:
either
es no determinista: puede recibir el mensaje -
como no hacer nada, con la misma probabilidad.
with *msg* ∈ *in_transit* **do**
received := *Append*(*received*, *msg*) ;
in_transit := *in_transit* \ {*msg*} ;
either
can_send := TRUE ;
or
skip ;
end either ;

end with ;
or
skip ;
end either ;

end while ;

assert *received* = ⟨1, 2, 3⟩ ; queremos que los mensajes lleguen en el mismo orden.

end algorithm ;

si bien eliminamos el error de concurrencia al introducir la bandera, introducimos otro mas sutil: solamente se puede enviar si la otra persona lo ha recibido, pero que pasaria si el mensaje nunca llega?, esto es conocido como un error de liveness.

Otra forma seria introducir un "either" dentro del "with", para darle no determinismo a la confirmacion de llegada. Pero esto nos lleva a un estado de deadlock, es decir falla la confirmacion.

BEGIN TRANSLATION

VARIABLES to_send, $received$, $in\_transit$, can_send, pc

$vars \triangleq \langle to\_send, received, in\_transit, can\_send, pc \rangle$

$Init \triangleq$ Global variables
$\land\ to\_send = \langle 1, 2, 3 \rangle$
$\land\ received = \langle\rangle$
$\land\ in\_transit = \{\}$
$\land\ can\_send = \text{TRUE}$
$\land\ pc =$ "Lbl_1"

$Lbl\_1 \triangleq$ $\land\ pc =$ "Lbl_1"
$\land$ IF $Len(received) \neq 3$
THEN $\land$ IF $can\_send \land to\_send \neq \langle\rangle$
THEN $\land\ in\_transit' = (in\_transit \cup \{Head(to\_send)\})$
$\land\ to\_send' = Tail(to\_send)$
$\land\ can\_send' = \text{FALSE}$
ELSE $\land$ TRUE
$\land$ UNCHANGED $\langle to\_send, in\_transit, can\_send \rangle$
$\land\ \lor\ \land\ pc' =$ "Lbl_2"
$\lor\ \land$ TRUE
$\land\ pc' =$ "Lbl_1"
ELSE $\land\ Assert(received = \langle 1, 2, 3 \rangle,$
"Failure of assertion at line 44, column 1.")
$\land\ pc' =$ "Done"
$\land$ UNCHANGED $\langle to\_send, in\_transit, can\_send \rangle$
$\land$ UNCHANGED $received$

$Lbl\_2 \triangleq$ $\land\ pc =$ "Lbl_2"
$\land\ \exists\, msg \in in\_transit :$
$\land\ received' = Append(received, msg)$
$\land\ in\_transit' = in\_transit \setminus \{msg\}$
$\land\ \lor\ \land\ can\_send' = \text{TRUE}$
$\lor\ \land$ TRUE
$\land$ UNCHANGED can_send
$\land\ pc' =$ "Lbl_1"
$\land$ UNCHANGED to_send

$Next \triangleq\ Lbl\_1 \lor Lbl\_2$
$\lor$ Disjunct to prevent deadlock on termination

$$(pc = \text{“Done”} \land \text{UNCHANGED } vars)$$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Termination \triangleq \Diamond(pc = \text{“Done”})$

END TRANSLATION

* Modification History
* Last modified Sat May 18 08:36:26 ART 2019 by *danilo*
* Created *Thu* May 16 10:44:51 ART 2019 by *danilo*