MODULE *ABProtocol*

EXTENDS *Integers*, *Sequences*, *SequencesExercise*

CONSTANT $Data$ The set of all possible data objects.

VARIABLES $AVar$, The last $\langle value, bit \rangle$ pair A decided to send.
$BVar$, The last $\langle value, bit \rangle$ pair B received.
$AtoB$, Sequence of DATA messages in transit from sender to receiver.
$BtoA$ Sequence of ACK messages in transit from receiver to sender.

$$
\begin{aligned}
TypeOK \triangleq\ & \land AVar \in Data \times \{0, 1\} \\
& \land BVar \in Data \times \{0, 1\} \\
& \land AtoB \in Seq(Data \times \{0, 1\}) \\
& \land BtoA \in Seq(\{0, 1\})
\end{aligned}
$$

$vars \triangleq \langle AVar, BVar, AtoB, BtoA \rangle$ All variables.

$$
\begin{aligned}
Init \triangleq\ & \land AVar \in Data \times \{1\} \\
& \land BVar = AVar \\
& \land AtoB = \langle\rangle \\
& \land BtoA = \langle\rangle
\end{aligned}
$$

A sending a data message to B by putting the same message in the channel until an ACK is received.

$$
\begin{aligned}
ASnd \triangleq\ & \land AtoB' = Append(AtoB, AVar) \\
& \land \text{UNCHANGED } \langle AVar, BtoA, BVar \rangle
\end{aligned}
$$

B receiving a data message from A.

$$
\begin{aligned}
BRcv \triangleq\ & \land AtoB \neq \langle\rangle \quad \text{There is at least one message in the channel.} \\
& \land \text{IF } Head(AtoB)[2] \neq BVar[2] \\
& \quad \text{THEN } BVar' = Head(AtoB) \quad \text{Accept the message if } ACK \text{ bit is the alternate bit.} \\
& \quad \text{ELSE } BVar' = BVar \quad \text{Ignore the message and keep the same local state.} \\
& \land AtoB' = Tail(AtoB) \quad \text{Remove the received message from the channel.} \\
& \land \text{UNCHANGED } \langle AVar, BtoA \rangle
\end{aligned}
$$

B sending an ACK for the last data value received.

$$
\begin{aligned}
BSnd \triangleq\ & \land BtoA' = Append(BtoA, BVar[2]) \\
& \land \text{UNCHANGED } \langle AVar, BVar, AtoB \rangle
\end{aligned}
$$

A receiving an ACK from B.

$$
\begin{aligned}
ARcv \triangleq\ & \land BtoA \neq \langle\rangle \quad \text{There is at least one message in the channel.} \\
& \land \text{IF } Head(BtoA) = AVar[2] \quad \text{Check the } ACK \text{ bit.} \\
& \quad \text{THEN } \exists\, d \in Data : AVar' = \langle d, 1 - AVar[2] \rangle \quad \text{Alternate bit and send another message.} \\
& \quad \text{ELSE } AVar' = AVar \quad \text{Keep sending } AVar \text{ if } ACK \text{ bit doesn't match.} \\
& \land BtoA' = Tail(BtoA) \quad \text{Remove received message from the channel.} \\
& \land \text{UNCHANGED } \langle AtoB, BVar \rangle
\end{aligned}
$$

Lose a message in one of the channels.

$$
\begin{array}{l}
LoseMsg \triangleq \land \lor \text{ Lose a data message.} \\
\qquad\qquad\qquad\quad \land \exists\, i \in 1 \,..\, Len(AtoB) : AtoB' = Remove(i,\, AtoB) \\
\qquad\qquad\qquad\quad \land BtoA' = BtoA \\
\qquad\qquad\quad \lor \text{ Lose an } ACK \text{ message.} \\
\qquad\qquad\qquad\quad \land \exists\, i \in 1 \,..\, Len(BtoA) : BtoA' = Remove(i,\, BtoA) \\
\qquad\qquad\qquad\quad \land AtoB' = AtoB \\
\qquad\qquad \land \text{UNCHANGED } \langle AVar,\, BVar \rangle
\end{array}
$$

$$
\begin{array}{l}
Next \triangleq \lor ASnd \\
\qquad\quad \lor BRcv \\
\qquad\quad \lor BSnd \\
\qquad\quad \lor ARcv \\
\qquad\quad \lor CorruptMsg \\
\qquad\quad \lor LoseMsg
\end{array}
$$

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

$$ABS \triangleq \text{INSTANCE } ABSpec$$

THEOREM $Spec \Rightarrow ABS!Spec$

$$
\begin{array}{l}
FairSpec \triangleq Spec \land \mathrm{SF}_{vars}(ARcv) \quad \land \mathrm{SF}_{vars}(BRcv) \\
\qquad\qquad\qquad\quad \land \mathrm{WF}_{vars}(ASnd) \land \mathrm{WF}_{vars}(ASnd)
\end{array}
$$

THEOREM $FairSpec \Rightarrow ABS!FairSpec$

In the first eight lectures, you learned about writing the safety part of a TLA+ spec. Now you know how to specify liveness. You simply add weak and strong fairness conditions. Simple, yes. Easy, no. Liveness is inherently subtle. TLA+ is the simplest way I know to express it, and it's still hard. But don't worry if you have trouble with liveness. The safety part is by far the largest part and almost always the most important part of a spec. A major reason to add liveness is to catch errors in the safety part. If your fairness conditions don't imply the eventually or leads-to properties you expect to hold, it could be because the safety part doesn't allow behaviors that it should.

– *Leslie Lamport*

---

* Modification History
* Last modified Sun *Oct* 31 15:57:20 *CET* 2021 by *felipec*
* Created Sun *Oct* 31 08:13:45 *CET* 2021 by *felipec*