MODULE *ABJonRoSpec*

EXTENDS *Integers*, *Sequences*

CONSTANT *Data*

$Remove(i,\ seq) \triangleq [j \in 1\ ..\ (Len(seq) - 1) \mapsto \text{IF } j < i \text{ THEN } seq[j] \text{ ELSE } seq[j+1]]$

VARIABLES *AVar*, Current state $\langle data,\ sequence\ bit \rangle$ of A
BVar, Current sequence bit of *B*
AtoB, Lossy *FIFO* channel from A to *B*
BtoA Lossy *FIFO* channel from *B* to A

The set of all sequence bits is 0 or 1
$SequenceBits \triangleq \{0, 1\}$

The set of all messages is everything from *Data* as the value, and everything in *SequenceBits* as the *seqBit*
$Messages \triangleq [value : Data,\ seqBit : SequenceBits]$

$$
\begin{aligned}
TypeOK \triangleq\ & \land (AVar \in Messages) && \text{AVar must be a message}\\
& \land (BVar \in SequenceBits) && \text{BVar is a sequence bit}\\
& \land (\forall\, i \in 1\ ..\ Len(AtoB) : AtoB[i] \in Messages) && \text{AtoB is a sequence of messages}\\
& \land (\forall\, i \in 1\ ..\ Len(BtoA) : BtoA[i] \in SequenceBits) && \text{BtoA is a sequence of sequence bits}
\end{aligned}
$$

The variables in the system
$vars \triangleq \langle AVar,\ BVar,\ AtoB,\ BtoA \rangle$

$$
\begin{aligned}
Init \triangleq\ & \land \exists\, d \in Data : AVar = [value \mapsto d,\ seqBit \mapsto 1] && \text{A starts with random data and a } seqBit \text{ of 1}\\
& \land BVar = AVar.seqBit && \text{B has already received message with sequence bit 1 from A}\\
& \land AtoB = \langle\rangle && \text{Nothing in the channel}\\
& \land BtoA = \langle\rangle && \text{Nothing in the channel}
\end{aligned}
$$

A to *B* A can read/write *AVar*, consume the *BtoA* channel, and produce the *AtoB* channel

If there are messages from *B* and the first message is an acknowledgement
of the last message A sent, then we're ready to advance to the next message
to send.

$$
\begin{aligned}
ASendNextMessage \triangleq\ & \land Len(BtoA) > 0\\
& \land AVar.seqBit = Head(BtoA)\\
& \land \exists\, d \in Data : AVar' = [value \mapsto d,\ seqBit \mapsto 1 - AVar.seqBit]\\
& \land AtoB' = AtoB\\
& \land BtoA' = Tail(BtoA) && \text{Consume the ack from } B\\
& \land \text{UNCHANGED } \langle BVar \rangle && \text{Don't touch } BVar
\end{aligned}
$$

If there are messages from *B* and the first message is not an acknowledgement
of the last message A sent, then ignore it.

$$
\begin{aligned}
AIgnoreBadAck \triangleq\ & \land Len(BtoA) > 0\\
& \land AVar.seqBit \neq Head(BtoA)\\
& \land AVar' = AVar\\
& \land AtoB' = AtoB
\end{aligned}
$$

$\land\ BtoA' = Tail(BtoA)$ Consume the bad message from B
$\land$ UNCHANGED $\langle BVar \rangle$ Don't touch $BVar$

At any time, A can retransmit what it previously sent

$ARetransmit \triangleq$ $\land\ AVar' = AVar$
$\land\ AtoB' = Append(AtoB,\ AVar')$ Produce the new message to the channel to B
$\land\ BtoA' = BtoA$
$\land$ UNCHANGED $\langle BVar \rangle$ Don't touch $BVar$

B to A B can read/write $BVar$, consume the $AtoB$ channel, and produce the $BtoA$ channel

If there are messages from A and the first message has a different sequence number than our current message, it's new! Consume and acknowledge it.

$BAcknowledge \triangleq$ $\land\ Len(AtoB) > 0$
$\land\ BVar \neq Head(AtoB).seqBit$
$\land\ BVar' = Head(AtoB).seqBit$ Remember the new sequence number
$\land\ AtoB' = Tail(AtoB)$ Consume the message from A
$\land\ BtoA' = BtoA$ Don't need to send a message to A to receive new data from A
$\land$ UNCHANGED $\langle AVar \rangle$ Don't touch

If there are messages from A and the first message has the same sequence number as our current message, ignore it.

$BIgnoreBadMsg \triangleq$ $\land\ Len(AtoB) > 0$
$\land\ BVar = Head(AtoB).seqBit$
$\land\ BVar' = BVar$
$\land\ AtoB' = Tail(AtoB)$ Consume the $AtoB$ channel
$\land\ BtoA' = BtoA$
$\land$ UNCHANGED $\langle AVar \rangle$

At any time, B can retransmit what it previously sent

$BRetransmit \triangleq$ $\land\ BVar' = BVar$
$\land\ AtoB' = AtoB$
$\land\ BtoA' = Append(BtoA,\ BVar')$ Produce the $BtoA$ channel
$\land$ UNCHANGED $\langle AVar \rangle$

Simulate loss by removing messages from $AtoB$ while maintaining $FIFO$

$DropAtoBMessage \triangleq$ $\land\ Len(AtoB) > 0$
$\land\ \exists\, i \in 1\,..\,Len(AtoB) : AtoB' = Remove(i,\ AtoB)$
$\land$ UNCHANGED $\langle AVar,\ BVar,\ BtoA \rangle$

Simulate loss by removing messages from $BtoA$ while maintaining $FIFO$

$DropBtoAMessage \triangleq$ $\land\ Len(BtoA) > 0$
$\land\ \exists\, i \in 1\,..\,Len(BtoA) : BtoA' = Remove(i,\ BtoA)$
$\land$ UNCHANGED $\langle AVar,\ BVar,\ AtoB \rangle$

$Next \triangleq$ $\lor\ ASendNextMessage$
$\lor\ AIgnoreBadAck$

$\lor\ ARetransmit$
$\lor\ BAcknowledge$
$\lor\ BIgnoreBadMsg$
$\lor\ BRetransmit$
$\lor\ DropAtoBMessage$
$\lor\ DropBtoAMessage$

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

THEOREM $Spec \Rightarrow \Box TypeOK$

$AB \triangleq$ INSTANCE $ABSpec$

THEOREM $Spec \Rightarrow AB!Spec$

* Modification History
* Last modified *Wed Apr* 06 12:59:57 *PDT* 2016 by *jonro*
* Created *Wed Apr* 06 10:20:29 *PDT* 2016 by *jonro*