MODULE *ABSpec*

EXTENDS *Integers*

CONSTANT *Data* The set of all possible data values.

VARIABLES *AVar*, The last $\langle value,\ bit \rangle$ pair A decided to send.
BVar The last $\langle value,\ bit \rangle$ pair B received.

Type correctness means that *AVar* and *BVar* are tuples $\langle d,\ i \rangle$ where $d \in Data$ and $i \in \{0, 1\}$.

$$\begin{aligned} TypeOK \triangleq\ & \land AVar \in Data \times \{0, 1\} \\ & \land BVar \in Data \times \{0, 1\} \end{aligned}$$

It's useful to define *vars* to be the tuple of all variables, for example so we can write $[Next]\_vars$ instead of $[Next]\_\langle \ldots \rangle$

$$vars \triangleq \langle AVar,\ BVar \rangle$$

Initially *AVar* can equal $\langle d,\ 1 \rangle$ for any *Data* value d, and *BVar* equals *AVar*.

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

When $AVar = BVar$, the sender can "send" an arbitrary data d item by setting $AVar[1]$ to d and complementing $AVar[2]$. It then waits until the receiver "receives" the message by setting *BVar* to *AVar* before it can send its next message. Sending is described by action A and receiving by action B.

$$\begin{aligned} A \triangleq\ & \land AVar = BVar \\ & \land \exists\, d \in Data : AVar' = \langle d,\ 1 - AVar[2] \rangle \\ & \land BVar' = BVar \end{aligned}$$

$$\begin{aligned} B \triangleq\ & \land AVar \neq BVar \\ & \land BVar' = AVar \\ & \land AVar' = AVar \end{aligned}$$

$$Next \triangleq A \lor B$$

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

For understanding the spec, it's useful to define formulas that should be invariants and check that they are invariant. The following invariant *Inv* asserts that, if *AVar* and *BVar* have equal second components, then they are equal (which by the invariance of *TypeOK* implies that they have equal first components).

$$Inv \triangleq (AVar[2] = BVar[2]) \Rightarrow (AVar = BVar)$$

$$DeliveryLiveness \triangleq \forall\, v \in Data \times \{0, 1\} : (AVar = v) \leadsto (BVar = v)$$

FairSpec is *Spec* with the addition requirement that it keeps taking steps.

$$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$$

FairABSpec is *Spec* with the additional requirement that both A and B keep taking steps.

$FairABSpec \triangleq Spec \land \mathrm{WF}_{vars}(A) \land \mathrm{WF}_{vars}(B)$

FairBSpec is *Spec* with the additional requirement that every sent value to be received, but allows the sender to stop sending.

$FairBSpec \triangleq Spec \land \mathrm{WF}_{vars}(B)$

\* Modification History
\* Last modified Sat *Aug* 11 22:07:13 *CST* 2018 by *hengxin*
\* Last modified Sat May 12 20:51:26 *CST* 2018 by *hengxin*
\* Last modified *Wed Oct* 18 04:07:37 *PDT* 2017 by *lamport*
\* Created *Fri Sep* 04 07:08:22 *PDT* 2015 by *lamport*