

EXTENDS *Integers*

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

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

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

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

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

$$\textit{vars} \triangleq \langle \textit{AVar}, \textit{BVar} \rangle$$

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

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

When  $\textit{AVar} = \textit{BVar}$ , the sender can “send” an arbitrary data *d* item by setting  $\textit{AVar}[1]$  to *d* and complementing  $\textit{AVar}[2]$ . 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.

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

$$\begin{aligned} B &\triangleq \wedge \textit{AVar} \neq \textit{BVar} \\ &\quad \wedge \textit{BVar}' = \textit{AVar} \\ &\quad \wedge \textit{AVar}' = \textit{AVar} \end{aligned}$$

$$\textit{Next} \triangleq A \vee B$$

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

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).

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

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

$$\textit{FairSpec} \triangleq \textit{Spec} \wedge \text{WF}_{\textit{vars}}(\textit{Next})$$

\ \* Modification History  
 \ \* Last modified Wed May 16 09:44:01 CST 2018 by tangruize  
 \ \* Last modified Wed Oct 18 04:07:37 PDT 2017 by lamport

\\* Created *Fri Sep 04 07:08:22 PDT 2015* by *lampo*