MODULE $AB2H$

This is spec $AB2$ with history variables $AtoB$ and $BtoA$ added so the spec implements spec $AB$ under the identity refinement mapping.

```
EXTENDS Integers, Sequences

CONSTANT Data, Bad
ASSUME Bad ∉ (Data × {0, 1}) ∪ {0, 1}
    \* We need to asssume that Bad is different from any of the legal
    \* messsages.

VARIABLES AVar, BVar,    \* The same as in module ABSpec
          AtoB2,  \* The sequence of data messages in transit from sender to receiver
          BtoA2   \* The sequence of ack messages in transit from receiver to sender
                  \* Messages are sent by appending them to the end of the sequence
                  \* and received by removing them from the head of the sequence.

AB2 ≜ INSTANCE AB2

\* We define RemoveBad so that RemoveBad(seq) removes from the sequence seq all elements that
\* equal Bad.
RECURSIVE RemoveBad(_)
  RemoveBad(seq) ≜
  IF seq = ⟨⟩ THEN ⟨⟩
              ELSE (IF Head(seq) = Bad THEN ⟨⟩ ELSE ⟨Head(seq)⟩)
                    ∘ RemoveBad(Tail(seq))

RECURSIVE RemoveBad2(_)
RemoveBad2(seq) ≜
  IF seq = ⟨⟩ THEN ⟨⟩
              ELSE IF Head(seq) = Bad
                     THEN RemoveBad(Tail(seq))
                     ELSE ⟨Head(seq)⟩ ∘ RemoveBad(Tail(seq))

VARIABLES AtoB, BtoA   \* Note that TLA+ allows multiple VARIABLE statements.

SpecH ≜ ∧ AB2!Spec
        ∧ □ ∧ AtoB = RemoveBad(AtoB2)
            ∧ BtoA = RemoveBad(BtoA2)

AB ≜ INSTANCE AB

\* The following theorem asserts that SpecH implements/refines the AB protocol. However, it can't
\* be checked by TLC because it doesn't have the form TLC requires of a specification.
THEOREM SpecH ⇒ AB!Spec
```

We now define *SpecHH* to be a specification that is equivalent to *SpecH* and that *TLC* can check. We write the definition of *SpecHH* in a way that should makes it clear that *SpecHH* is equivalent to *SpecH*.

$$TypeOKH \triangleq \land AB2!TypeOK \\ \land AtoB \in Seq(Data \times \{0, 1\}) \\ \land BtoA \in Seq(\{0, 1\})$$

$$InitH \triangleq \land AB2!Init \\ \land AtoB = RemoveBad(AtoB2) \\ \land BtoA = RemoveBad(BtoA2)$$

$$NextH \triangleq \land AB2!Next \\ \land AtoB' = RemoveBad(AtoB2') \\ \land BtoA' = RemoveBad(BtoA2')$$

We would normally define *varsH* to be the tuple of all the variables of the current module. However, we can use the following shorter definition instead because

UNCHANGED $\langle\langle AVar, \ldots, BtoA2\rangle, AtoB, BtoA\rangle$

equals

UNCHANGED $\langle AVar, \ldots, BtoA2, AtoB, BtoA\rangle$

$$varsH \triangleq \langle AB2!vars, AtoB, BtoA\rangle$$

$$SpecHH \triangleq InitH \land \Box[NextH]_{varsH}$$

The following theorem asserts that *SpecHH* and *SpecH* are equivalent specifications. It is equivalent to

$\land SpecHH \Rightarrow SpecH$
$\land SpecH \Rightarrow SpecHH$

*TLC* can check the first of these implications by showing that *SpecH* is a property satisfied by the specification *SpecHH*, but not the second.

THEOREM $SpecHH \equiv SpecH$

We can deduce that *SpecH* implies $AB!Spec$ from $SpecHH \equiv SpecH$ and the following theorem, which *TLC* can check.

THEOREM $SpecHH \Rightarrow AB!Spec$

* Modification History
* Last modified Sat *Jun* 11 21:51:02 *CST* 2022 by *wengjialin*
* Last modified Sun May 13 23:53:20 *PDT* 2018 by *lamport*
* Created *Wed Mar* 25 11:53:40 *PDT* 2015 by *lamport*