MODULE *MinMax*1

This module and modules *MinMax*2 and *MinMax*2*H* are used as examples in Sections 1 and 2 of the paper "Auxiliary Variables in TLA+".

This module specifies a tiny system in which a user presents a server with a sequence of integer inputs, and the server responds to each input value i with with one of the following outputs: *Hi* if i is the largest number input so far, *Lo* if it's the smallest number input so far, *Both* if it's both, and *None* if it's neither.

The module is part of an example illustrating the use of a history variable. The example includes this module, module *MinMax*2, and module *MinMax*2*H* which adds a history variable to the specification of *MinMax*2 and shows that the resulting specification implements implements the specification of the current module under a suitable refinement mapping.

$\textsc{extends}\ \mathit{Integers}$

We define *setMax*(S) and *setMin*(S) to be largest and smallest value in a nonempty finite set S of integers.

$$\begin{aligned}
&\mathit{setMax}(S) \triangleq \textsc{choose}\ t \in S : \forall\, s \in S : t \geq s \\
&\mathit{setMin}(S) \triangleq \textsc{choose}\ t \in S : \forall\, s \in S : t \leq s
\end{aligned}$$

The possible values that can be returned by the system are declared to be constants, which we assume are not integers.

$$\begin{aligned}
&\textsc{constants}\ \mathit{Lo},\ \mathit{Hi},\ \mathit{Both},\ \mathit{None} \\
&\textsc{assume}\ \{\mathit{Lo},\ \mathit{Hi},\ \mathit{Both},\ \mathit{None}\} \cap \mathit{Int} = \{\}
\end{aligned}$$

The the value of the variable x is the value input by the user or the value output by the system, the variable *turn* indicating which. The variable y holds the set of all values input thus far. We consider x and *turn* to be externally visible and y to be internal.

$$\begin{aligned}
&\textsc{variables}\ x,\ \mathit{turn},\ y \\
&\mathit{vars} \triangleq \langle x,\ \mathit{turn},\ y\rangle
\end{aligned}$$

The initial predicate *Init*:

$$\begin{aligned}
\mathit{Init} \triangleq\ &\land x = \mathit{None} \\
&\land \mathit{turn} = \text{“input”} \\
&\land y = \{\}
\end{aligned}$$

The user's input action:

$$\begin{aligned}
\mathit{InputNum} \triangleq\ &\land \mathit{turn} = \text{“input”} \\
&\land \mathit{turn}' = \text{“output”} \\
&\land x' \in \mathit{Int} \\
&\land y' = y
\end{aligned}$$

The systems response action:

$$\begin{aligned}
\mathit{Respond} \triangleq\ &\land \mathit{turn} = \text{“output”} \\
&\land \mathit{turn}' = \text{“input”} \\
&\land y' = y \cup \{x\} \\
&\land x' = \textsc{if}\ x = \mathit{setMax}(y')\ \textsc{then}\ \textsc{if}\ x = \mathit{setMin}(y')\ \textsc{then}\ \mathit{Both}\ \textsc{else}\ \mathit{Hi} \\
&\qquad\qquad\qquad\qquad\qquad\ \ \textsc{else}\ \textsc{if}\ x = \mathit{setMin}(y')\ \textsc{then}\ \mathit{Lo}\ \ \textsc{else}\ \mathit{None}
\end{aligned}$$

The next-state action:

$$Next \triangleq InputNum \lor Respond$$

The specification, which is a safety property (it asserts no liveness condition).

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

Invariant to check (added by hengxin)

$$\begin{aligned} NoneCertificate \triangleq \Box[& \land x \in Int \\ & \land x < setMax(y \cup \{x\}) \\ & \land x > setMin(y \cup \{x\}) \\ & \equiv x' = None]_{vars} \end{aligned}$$

Below, we check that specification *Spec* implements specification *Spec* of module *MinMax*2 under a suitable refinement mapping. The following definitions of *Infinity* and *MinusInfinity* are copied from module *MinMax*2.

$$\begin{aligned} Infinity &\triangleq \text{CHOOSE } n : n \notin Int \\ MinusInfinity &\triangleq \text{CHOOSE } n : n \notin (Int \cup \{Infinity\}) \end{aligned}$$

$$\begin{aligned} M \triangleq \text{INSTANCE } & MinMax2 \\ \text{WITH } & min \leftarrow \text{IF } y = \{\} \text{ THEN } Infinity \text{ ELSE } setMin(y), \\ & max \leftarrow \text{IF } y = \{\} \text{ THEN } MinusInfinity \text{ ELSE } setMax(y) \end{aligned}$$

The following theorem asserts that *Spec* implements the specification Spec of module *MinMax*2 under the refinement mapping defined by the INSTANCE statement. The theorem can be checked with *TLC* using a model having *M* ! *Spec* as the temporal property to be checked.

$$\text{THEOREM } Spec \Rightarrow M!Spec$$

\* Modification History

\* Last modified *Tue Jun* 26 17:51:08 *CST* 2018 by *hengxin*

\* Last modified *Tue Jun* 26 17:29:03 *CST* 2018 by *hengxin*

\* Last modified *Fri Oct* 21 23:48:10 *PDT* 2016 by *lamport*

\* Created *Fri Aug* 26 14:28:26 *PDT* 2016 by *lamport*