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.

EXTENDS *Integers*

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

$$setMax(S) \triangleq \text{CHOOSE } t \in S : \forall\, s \in S : t \geq s$$
$$setMin(S) \triangleq \text{CHOOSE } t \in S : \forall\, s \in S : t \leq s$$

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

CONSTANTS *Lo*, *Hi*, *Both*, *None*
ASSUME $\{Lo, Hi, Both, None\} \cap Int = \{\}$

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.

VARIABLES x, *turn*, y
$vars \triangleq \langle x, turn, y\rangle$

The initial predicate *Init*:

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

The user's input action:

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

The systems response action:

$$\begin{aligned} Respond \triangleq\ & \land turn = \text{"output"} \\ & \land turn' = \text{"input"} \\ & \land y' = y \cup \{x\} \\ & \land x' = \text{IF } x = setMax(y') \text{ THEN IF } x = setMin(y') \text{ THEN } Both \text{ ELSE } Hi \\ & \qquad\qquad \text{ELSE IF } x = setMin(y') \text{ THEN } Lo \text{ ELSE } None \end{aligned}$$

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)

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

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.

$Infinity \triangleq \text{CHOOSE } n : n \notin Int$

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

$$M \triangleq \text{INSTANCE } MinMax2 \\ \text{WITH } min \leftarrow \text{IF } y = \{\} \text{ THEN } Infinity \text{ ELSE } setMin(y) + 1, \\ max \leftarrow \text{IF } y = \{\} \text{ THEN } MinusInfinity \text{ ELSE } setMax(y)$$

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.

THEOREM $Spec \Rightarrow M\,!\,Spec$

---

* Modification History
* *Last* modified *Wed Oct* 31 19:44:20 *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*