MODULE *TowersOfHanoi*

****

TOWERS OF *HANOI* is a classical Puzzle Game. It consists of three Rods on Top of which Disks with various diameters can be stacked. In the beginning all disks are stacked with their order having decreasing diameter from bottom to top. The Puzzles idea is to move that stack, persisting the order to the far right rod.

Number of moves requied is $2\hat{}n - 1$, where n is the number of disks

Legal Moves :
– Move one Disk at a time
– ∀ Move : take upper disk from one stack, place it on top of another stack
- Disks can not be placed on top of a smaller disk

****

EXTENDS *Naturals*, *Sequences*, *TLC*

$FlattenSeq(seqs) \triangleq$
From TLA+ *CommunityModules SequencesExt*
IF $Len(seqs) = 0$ THEN $seqs$ ELSE
LET $flatten[i \in 1 \,..\, Len(seqs)] \triangleq$
IF $i = 1$ THEN $seqs[i]$ ELSE $flatten[i-1] \circ seqs[i]$
IN $flatten[Len(seqs)]$

CONSTANT *NumberOfDisks*
ASSUME $NumberOfDisks \in Nat$

$CorrectTower[disk \in 1 \,..\, NumberOfDisks] \triangleq disk$
$InitialPuzzle[tower \in 1 \,..\, 3] \triangleq$
IF $tower = 1$
THEN $CorrectTower$
ELSE $\langle\rangle$

VARIABLE *Towers*
$TowerDomain \triangleq$ DOMAIN *Towers*

$Init \triangleq Towers = InitialPuzzle$

$$TargetTowerIsEmpty(towerTo) \triangleq Len(towerTo) = 0$$

$$\begin{array}{l} DiskIsSmallerOrTowerIsEmpty(towerFrom,\ towerTo) \triangleq \\ \quad \text{LET } topElementOfOrigin \triangleq Head(towerFrom) \\ \qquad\quad topElementOfTarget \triangleq Head(towerTo) \\ \quad \text{IN IF } TargetTowerIsEmpty(towerTo) \\ \qquad\qquad \text{THEN TRUE} \\ \qquad\quad \text{ELSE } topElementOfTarget > topElementOfOrigin \end{array}$$

$$OriginTowerIsNotEmpty(towerFrom) \triangleq towerFrom \neq \langle\rangle$$

$$\begin{array}{l} CanMoveDisk(towerFrom,\ towerTo) \triangleq \\ \quad \land\ OriginTowerIsNotEmpty(towerFrom) \\ \quad \land\ DiskIsSmallerOrTowerIsEmpty(towerFrom,\ towerTo) \end{array}$$

$$\begin{array}{l} MoveDisk(from,\ to,\ towerFrom,\ towerTo) \triangleq \\ \quad \text{LET } fromWithoutTop \triangleq Tail(towerFrom) \\ \qquad\quad top \triangleq \langle Head(towerFrom)\rangle \\ \qquad\quad topWithTopOnTop \triangleq top \circ towerTo \\ \quad \text{IN } Towers' = [Towers \text{ EXCEPT} \\ \qquad\qquad\qquad ![from] = fromWithoutTop, \\ \qquad\qquad\qquad ![to] = topWithTopOnTop] \end{array}$$

$$\begin{array}{l} Next \triangleq \\ \quad \exists\, from,\ to \in TowerDomain : \\ \qquad \text{LET } towerFrom \triangleq Towers[from] \\ \qquad\quad towerTo \triangleq Towers[to] \\ \qquad \text{IN } \land\ CanMoveDisk(towerFrom,\ towerTo) \\ \qquad\quad \land\ MoveDisk(from,\ to,\ towerFrom,\ towerTo) \end{array}$$

$$\begin{array}{l} IsSorted(tower) \triangleq \\ \quad \forall\, i,\ j \in 1\,..\,Len(tower) : \\ \qquad i < j \Rightarrow tower[i] \leq tower[j] \end{array}$$

$$\begin{array}{l} OnlyContainsAllowedNumberOfDisks \triangleq \\ \quad Len(FlattenSeq(Towers)) = 5 \end{array}$$

$$\begin{array}{l} TypeOK \triangleq \\ \quad \land\ \forall\, tower \in TowerDomain : \\ \qquad\quad \text{LET } towerToCheck \triangleq Towers[tower] \\ \qquad\quad \text{IN } \land\ IsSorted(towerToCheck) \\ \qquad\qquad\quad \land\ Len(towerToCheck) \leq NumberOfDisks \\ \quad \land\ OnlyContainsAllowedNumberOfDisks \end{array}$$

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

THEOREM $Spec \Rightarrow \Box\, TypeOK$

$$InvariantOrElseFinished \triangleq$$
$$Towers[3] \neq CorrectTower$$

* Modification History
* Last modified *Wed Mar* 20 17:16:09 *CET* 2024 by *jeujeus*
* Created *Wed Mar* 20 13:27:31 *CET* 2024 by *jeujeus*