MODULE *Nim*

EXTENDS *Naturals*, *Sequences*

This is a model for the game *Nim*. A description can be found here: https://*plus.maths.org*/content/play-win-nim

VARIABLES *playerTurn*, *heaps*, *gameOver*, *winner*
$vars \triangleq \langle playerTurn, heaps, gameOver, winner \rangle$

from: https://*learntla.com*/core/advanced-*operators.html*

RECURSIVE $SumSeq(\_)$
$SumSeq(s) \triangleq$ IF $s = \langle\rangle$ THEN 0 ELSE
  $Head(s) + SumSeq(Tail(s))$

Due to state explosion, it's a good idea to keep the heap relatively small.

The *playerTurn* variable assumes the value "START" only in the first state and thenceforth can have the states "P1" and "P2"

The *gameOver* variable becomes true when there is only one item total left in the heaps.

$$
\begin{aligned}
Init \triangleq\ & \land heaps = \langle 1, 1, 1 \rangle \\
& \land playerTurn = \text{"START"} \\
& \land gameOver = \text{FALSE} \\
& \land winner = \text{"NONE"}
\end{aligned}
$$

$$
\begin{aligned}
TypeOK \triangleq\ & \land playerTurn \in \{\text{"P1"}, \text{"P2"}, \text{"START"}\} \\
& \land winner \in \{\text{"P1"}, \text{"P2"}, \text{"NONE"}\} \\
& \land Len(heaps) = 3 \\
& \land gameOver \in \{\text{TRUE}, \text{FALSE}\}
\end{aligned}
$$

We can take from a heap when it is not empty. At least one must be taken, up to the full amount of the heap such that the total sizes of the heaps is greater than zero. For example, given a heap $\langle 0, 0, 5 \rangle$ a *playerTurn* can from 1 to 4 from it, since that *playerTurn* wants to leave one item for the other *playerTurn* to take (*i.e.* a *playerTurn* won't lose on purpose).

$$
\begin{aligned}
TakeFromHeap(heapNum) \triangleq\ & \land heaps[heapNum] \neq 0 \\
& \land \exists\, amount \in 1 \,..\, heaps[heapNum] : \\
& \quad \land amount < SumSeq(heaps) \\
& \quad \land heaps' = [\\
& \qquad heaps \text{ EXCEPT} \\
& \qquad\quad ![heapNum] = heaps[heapNum] - amount \\
& \quad]
\end{aligned}
$$

The *playerTurn* whose turn it is can attempt to take from one of the three heaps.

$$
\begin{aligned}
Move \triangleq\ & \lor\ TakeFromHeap(1) \\
& \lor\ TakeFromHeap(2) \\
& \lor\ TakeFromHeap(3)
\end{aligned}
$$

$$
\begin{aligned}
Next \triangleq\ & \land\ gameOver = \text{FALSE} \\
& \land\ Move \\
& \land\ playerTurn' = \text{IF } playerTurn = \text{“P2”} \lor playerTurn = \text{“START”} \\
& \qquad\qquad \text{THEN “P1”} \\
& \qquad\qquad \text{ELSE “P2”} \\
& \land\ gameOver' = \text{IF } SumSeq(heaps') = 1 \\
& \qquad\qquad\qquad \text{THEN TRUE} \\
& \qquad\qquad\qquad \text{ELSE FALSE} \\
& \land\ winner' = \text{IF } gameOver' = \text{TRUE} \\
& \qquad\qquad \text{THEN } playerTurn' \\
& \qquad\qquad \text{ELSE “NONE”}
\end{aligned}
$$

These are my attempts at writing temporal formulas to specify that both *playerTurns* can win. They don't work because they say either that eventually $playerTurn =$ “P1” AND $playerTurn =$ “P2” must be true simultaneously or that either $playerTurn =$ “P1” OR $playerTurn =$ “P2” must be true.

But what *Iwant* to specify is that there exist at least one state where $playerTurn =$ “P1” and at least one state where $playerTurn =$ “P2”

$$
\begin{aligned}
&P1EventuallyLoses \triangleq \Diamond(gameOver \land playerTurn = \text{“P1”}) \\
&P2EventuallyLoses \triangleq \Diamond(gameOver \land playerTurn = \text{“P2”}) \\
&BothplayerTurnsCanWin \triangleq \lor \Diamond(gameOver \land playerTurn = \text{“P1”}) \\
&\qquad\qquad\qquad \lor \Diamond(gameOver \land playerTurn = \text{“P2”})
\end{aligned}
$$

A sanity check invariant, it should be trivially true based on the *Next* predicate.

$$
\begin{aligned}
GameOverAtOneLeft \triangleq\ & \text{IF } SumSeq(heaps) = 1 \\
& \qquad \text{THEN } gameOver = \text{TRUE} \\
& \qquad \text{ELSE } gameOver = \text{FALSE}
\end{aligned}
$$

$$Spec \triangleq Init \land \Box[Next]_{vars} \land \mathrm{WF}_{vars}(Next)$$