MODULE *ElevatorControl*

1. Loop through elevators (currently only one) a. Is elevator available
   i. Yes 1. Is a request made?
      a. Yes i. Make elevator unavailable. ii. Move elevator to service request
      b. No, do nothing
   ii. No, do nothing for that elevator because it is servicing a request

This currently does not work, but looks like it should. It gives the following output when running the model checker:

> Attempted to check equality of integer 1 with non-integer: "floor" The error occurred when *TLC* was evaluating the nested expressions at the following positions: The error call stack is empty.

I have tried several troubleshooting paths with no progress from this point.

EXTENDS *Naturals*, *TLC*
CONSTANT *TotalElevators*, *TotalFloors*
VARIABLE *elevator*, *request*

$$
\begin{array}{l}
\mathit{TypeInvariant} \triangleq \\
\quad \land \mathit{elevator} \in [(1 \,..\, \mathit{TotalElevators}) \to [\\
\qquad\qquad \mathit{floor} \quad\quad\quad\quad : (1 \,..\, \mathit{TotalFloors}), \\
\qquad\qquad \mathit{available} \quad\quad\quad : \{\text{TRUE, FALSE}\}, \\
\qquad\qquad \mathit{requestedFloor} \;\; : (0 \,..\, \mathit{TotalFloors})]] \\
\quad \land \mathit{request} \in [(1 \,..\, \mathit{TotalFloors}) \to \{\text{TRUE, FALSE}\}]
\end{array}
$$

$$
\begin{array}{l}
\mathit{Init} \triangleq \land \mathit{TypeInvariant} \\
\qquad\quad \land \mathit{elevator} = [\mathit{elev} \in (1 \,..\, \mathit{TotalElevators}) \mapsto \\
\qquad\qquad\qquad\quad [\mathit{elevator} \text{ EXCEPT } !.\mathit{floor} = 1,\; !.\mathit{available} = \text{TRUE},\; !.\mathit{RequestedFloor} = 0]] \\
\qquad\quad \land \mathit{request} = [\mathit{req} \in (1 \,..\, \mathit{TotalFloors}) \mapsto \text{FALSE}]
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextElevator}(\mathit{elev},\, \mathit{req}) \triangleq \\
\quad \land (\mathit{elevator}[\mathit{elev}]).\mathit{available} = \text{TRUE} \\
\quad \land \mathit{elevator}' = \text{IF } \mathit{request}[\mathit{req}] = \text{TRUE} \\
\qquad\qquad\qquad\quad \text{THEN } [\mathit{elevator} \text{ EXCEPT } ![\mathit{elev}].\mathit{available} = \text{FALSE},\; ![\mathit{elev}].\mathit{requestedFloor} = \mathit{req}] \\
\qquad\qquad\quad \text{ELSE } \mathit{elevator} \\
\quad \land \mathit{request}' \;\; = \text{IF } \mathit{request}[\mathit{req}] = \text{TRUE} \\
\qquad\qquad\qquad\quad \text{THEN } [\mathit{request} \text{ EXCEPT } ![\mathit{req}] = \text{FALSE}] \\
\qquad\qquad\quad \text{ELSE } \mathit{request}
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextFloor}(\mathit{elev}) \triangleq \\
\quad \land (\mathit{elevator}[\mathit{elev}]).\mathit{available} = \text{FALSE} \\
\quad \land \mathit{elevator}' = \text{IF } (\mathit{elevator}[\mathit{elev}]).\mathit{floor} = (\mathit{elevator}[\mathit{elev}]).\mathit{requestedFloor} \\
\qquad\qquad\qquad\quad \text{THEN } [\mathit{elevator} \text{ EXCEPT } ![\mathit{elev}].\mathit{available} = \text{TRUE},\; ![\mathit{elev}].\mathit{requestedFloor} = 0] \\
\qquad\qquad\quad \text{ELSE IF } (\mathit{elevator}[\mathit{elev}]).\mathit{requestedFloor} = 0 \\
\qquad\qquad\qquad\quad \text{THEN } \mathit{elevator} \\
\qquad\qquad\quad \text{ELSE IF } (\mathit{elevator}[\mathit{elev}]).\mathit{floor} > (\mathit{elevator}[\mathit{elev}]).\mathit{requestedFloor}
\end{array}
$$

```
                    THEN [elevator EXCEPT ![elev].floor = (elevator[elev].floor % TotalFloors) - 1]
                ELSE IF (elevator[elev]).floor < (elevator[elev]).requestedFloor
                    THEN [elevator EXCEPT ![elev].floor = (elevator[elev].floor % TotalFloors) + 1]
                ELSE elevator
  ∧ UNCHANGED request

RequestMade ≜
  ∀ req ∈ (1 .. TotalFloors) : request[req] = TRUE

NextRequest(req) ≜
  ∧ elevator.available = TRUE
  ∧ request' = IF RequestMade = FALSE
                  THEN [request EXCEPT ![req] = TRUE]
               ELSE request
  ∧ UNCHANGED elevator

NextOperation(elev, req) ≜
  ∨ NextElevator(elev, req)
  ∨ NextFloor(elev)
  ∨ NextRequest(req)

Next ≜
  ∧ ∃ elev ∈ (1 .. TotalElevators) : (∃ req ∈ (1 .. TotalFloors) : NextOperation(elev, req))
  ∧ PrintT(elevator)
  ∧ PrintT(request)


Spec ≜ Init ∧ □[Next]⟨elevator, request⟩
```

THEOREM $Spec \Rightarrow TypeInvariant$

```
\* Modification History
\* Last modified Fri May 29 23:29:16 PDT 2015 by Me
\* Created Tue May 26 16:13:01 PDT 2015 by Me
```