MODULE *DieHard*

In the movie Die Hard 3, the heros must obtain exactly 4 gallons of water using a 5 gallon jug, a 3 gallon jug, and a water faucet. Our goal: to get *TLC* to solve the problem for us.

First, we write a spec that describes all allowable behaviors of our heros.

EXTENDS *Naturals*

This statement imports the definitions of the ordinary operators on natural numbers, such as +.

We next declare the specifications varibles.

VARIABLES *big*, — The number of gallons of water in the 5 gallon jug.
small — The number of gallons of water in the 3 gallon jug.

We now define *TypeOK* to be the type invariant, asserting that the value of each variable is an element of the appropriate set. A type invariant like this is not part of the specification, but it's generally a good idea to include it because it helps the reader understand the spec. Moreover, having *TLC* check that it is an invariant of the spec catches errors that, in a typed language, are caught by type checking.

Note: TLA+ uses the convention that a list of formulas bulleted by $\land$ or $\lor$ denotes the conjunction or disjunction of those formulas. Indentation of subitems is significant, allowing one to eliminate lots of parentheses. This makes a large formula much easier to read. However, it does mean that you have to be careful with your

$$\mathit{TypeOK} \triangleq \begin{array}{l} \land\ \mathit{small} \in 0 \,..\, 3 \\ \land\ \mathit{big} \in 0 \,..\, 5 \end{array}$$

Now we define of the initial predicate, that specifies the initial values of the variables. I like to name this predicate *Init*, but the name doesn't matter.

$$\mathit{Init} \triangleq \begin{array}{l} \land\ \mathit{big} = 0 \\ \land\ \mathit{small} = 0 \end{array}$$

Now we define the actions that our hero can perform. There are three things they can do:

- Pour water from the faucet into a jug.
- Pour water from a jug onto the ground.
- Pour water from one jug into another

We now consider the first two. Since the jugs are not calibrated, partially filling or partially emptying a jug accomplishes nothing. So, the first two possibilities yield the following four possible actions.

$$\mathit{FillSmallJug} \triangleq \begin{array}{l} \land\ \mathit{small}' = 3 \\ \land\ \mathit{big}' = \mathit{big} \end{array}$$

$$\mathit{FillBigJug} \triangleq \begin{array}{l} \land\ \mathit{big}' = 5 \\ \land\ \mathit{small}' = \mathit{small} \end{array}$$

$$\mathit{EmptySmallJug} \triangleq \begin{array}{l} \land\ \mathit{small}' = 0 \\ \land\ \mathit{big}' = \mathit{big} \end{array}$$

$$EmptyBigJug \triangleq \land big' = 0$$
$$\land small' = small$$

We now consider pouring water from one jug into another. Again, since the jugs are not callibrated, when pouring from jug A to jug B, it makes sense only to either fill A or empty B. And there's no point in emptying B if this will cause A to overflow, since that could be accomplished by the two actions of first filling A and then emptying B. So, pouring water from A to B leaves A with the lesser of (i) the water contained in both jugs and (ii) the volume of A. To express this mathematically, we first define $Min(m, n)$ to equal the minimum of the numbers m and n.

$$Min(m, n) \triangleq \text{IF } m < n \text{ THEN } m \text{ ELSE } n$$

Now we define the last two pouring actions. From the observation above, these definitions should be clear.

$$SmallToBig \triangleq \land big' = Min(big + small, 5)$$
$$\land small' = small - (big' - big)$$

$$BigToSmall \triangleq \land small' = Min(big + small, 3)$$
$$\land big' = big - (small' - small)$$

We define the next-state relation, which I like to call *Next*. A *Next* step is a step of one of the six actions defined above. Hence, *Next* is the disjunction of those actions.

$$Next \triangleq \lor FillSmallJug$$
$$\lor FillBigJug$$
$$\lor EmptySmallJug$$
$$\lor EmptyBigJug$$
$$\lor SmallToBig$$
$$\lor BigToSmall$$

We define the formula *Spec* to be the complete specification, asserting of a behavior that it begins in a state satisfying *Init*, and that every step either satisfies *Next* or else leaves the pair $\langle big, small \rangle$ unchanged.

$$Spec \triangleq Init \land \Box[Next]_{\langle big, small \rangle}$$

Remember that our heros must measure out 4 gallons of water. Obviously, those 4 gallons must be in the 5 gallon jug. So, they have solved their problem when they reach a state with $big = 4$. So, we define *NotSolved* to be the predicate asserting that $big \neq 4$.

$$NotSolved \triangleq big \neq 4$$

We find a solution by having *TLC* check if *NotSolved* is an invariant, which will cause it to print out an "error trace" consisting of a behavior ending in a states where *NotSolved* is false. Such a behavior is the desired solution. (Because *TLC* uses a breadth-first search, it will find the shortest solution.)