MODULE *CarTalkPuzzle*

Car Talk is a *U.S.* radio program about car repair. Each show includes a puzzle, which is often a little mathematical problem. Usually, those problems are easy for a mathematically sophisticated listener to solve. However, I was not able immediately to see how to solve the puzzle from the 22 *October* 2011 program. I decided to specify the problem in TLA+ and let *TLC* (the TLA+ model checker) compute the solution. This is the specification I wrote. (I have tried to explain in comments all TLA+ notation that is not standard mathematical notation.) Once *TLC* had found the solution, it was not hard to understand why it worked.

Here is the problem. A farmer has a 40 pound stone and a balance scale. How can he break the stone into 4 pieces so that, using those pieces and the balance scale, he can weigh out any integral number of pounds of corn from 1 pound through 40 pounds.

The following statement imports the standard operators of arithmetic such as $+$ and $\leq$ (less than or equals). It also defines the operator $..$ so that $i \,..\, j$ is the set of all integers k with $i \leq k \leq j$.

EXTENDS *Integers*

For generality, I solve the problem of breaking an N pound stone into P pieces. The following statement declares N and P to be unspecified constant values.

CONSTANTS N, P

I define the operator *Sum* so that if f is any integer-valued function, and S any finite subset of its domain, then $Sum(f, S)$ is the sum of $f[x]$ for all x in S. (In TLA+, function application is indicated by square brackets instead of parentheses, as it is in ordinary math.)

A RECURSIVE declaration must precede a recursively defined operator. The operator CHOOSE is known to logicians as *Hilbert*'s Epsilon. It is defined so that CHOOSE $x \in S : P(x)$ equals some unspecified value v such that $P(v)$ is true, if such a value exists.

RECURSIVE $Sum(\_, \_)$

$$Sum(f, S) \triangleq \text{IF } S = \{\} \text{ THEN } 0 \text{ ELSE LET } x \triangleq \text{CHOOSE } x \in S : \text{TRUE IN } f[x] + Sum(f, S \setminus \{x\})$$

I now define the set *Break* of all "breaks", where a break represents a method of breaking the stone into P (integer-weight) pieces. The obvious definition of a break would be a set of weights. However, that doesn't work because it doesn't handle the situation in which two of pieces have the same weight. Instead, I define a break of the N pound stone into P pieces to be a function B from $1 \,..\, P$ (the integers from 1 through P) into $1 \,..\, N$ such that $B[i]$ is the weight of piece number i. To avoid solutions that differ only by how the pieces are numbered, I consider only breaks in which the pieces are numbered in non-decreasing order of their weight. This leads to the following definition of the set *Break* of all breaks.

In TLA+, $[S \to T]$ is the set of all functions with domain S and range a subset of T. $\forall$ and $\exists$ are the universal and existential quantifiers.

$$Break \triangleq \{B \in [1 \,..\, P \to 1 \,..\, N] : \; Sum(B, 1 \,..\, P) = N \land \forall\, i \in 1 \,..\, P : \forall\, j \in (i+1) \,..\, P : B[i] \leq B[j]\}$$

To weigh a quantity of corn, we can put some of the weights on the same side of the balance scale as the corn and other weights on the other side of the balance. The following operator is true for a weight w, a break B, and sets S and T of pieces if w plus the weight of the pieces in S equals the weight of the pieces in T. The elements of S and T are piece numbers (numbers in $1 \,..\, P$), so $Sum(B, S)$ is the weight of the pieces in S.

$$IsRepresentation(w,\ B,\ S,\ T) \triangleq \quad S \cap T = \{\} \\ \land\ w + Sum(B,\ S) = Sum(B,\ T)$$

I now define $IsSolution(B)$ to be true iff break B solves the problem, meaning that it can be used to balance any weight in $1\ ..\ N$.

SUBSET S is the set of all subsets of S (the power set of S).

$$IsSolution(B) \triangleq \quad \forall\, w \in 1\ ..\ N : \\ \exists\, S,\ T \in \text{SUBSET}\ (1\ ..\ P) : IsRepresentation(w,\ B,\ S,\ T)$$

I define *AllSolutions* to be the set of all breaks B that solve the problem.

$$AllSolutions \triangleq \{B \in Break : IsSolution(B)\}$$

We can now have *TLC* compute the solution to the problem as follows. We open this module in the TLA+ *Toolbox* (an Integrated Development Environment for the TLA+ tools). We then create a new *TLC* model in which we assign the values 40 to N and 4 to P. We specify in the model that *TLC* should compute the value of *AllSolutions* and run *TLC* on the model. (We do this by entering *AllSolutions* in the Evaluate Constant Expression section of the model's Model Checking Results page.) After running for 22 seconds on my 3 year old 2.5 *GHz* laptop, it prints the result

$\{\langle 1,\ 3,\ 9,\ 27\rangle\}$

In TLA+, a k-tuple is represented as a function f with domain $1\ ..\ k$. Therefore, *TLC* prints a break B, which with $P = 4$ is a function with domain $1\ ..\ 4$, as the tuple $\langle B[1],\ B[2],\ B[3],\ B[4]\rangle$. Its output therefore indicates that there is a single break that solves the Car Talk puzzle, and it breaks the stone into pieces of weights 1, 3, 9, and 27 pounds.

You have undoubtedly observed that the weights of the four pieces are 3^0, 3^1, 3^2, and 3^3. You may also have observed that 40 equals 1111 base 3. These facts should give you enough of a hint to be able to answer this:

For what values of N and P is the problem solvable, and what is a solution for those values?

It's a good idea to check that the definition of *AllSolutions* really generates solutions. The following operator defines *ExpandSolutions* to be a set of sequences, one for each solution in *AllSolutions*. Each of those sequences is of length N, where the element i shows how to weigh i pounds of corn. For example, for the single solution with $N = 40$ and $P = 4$, element 7 of the sequence is

$\langle 7,\ \{3\},\ \{1,\ 9\}\rangle$

indicating that to weight 7 pounds of corn, we can put the 3 pound weight on the same side of the balance as the corn and the 1 and 9 pound weights on the other side. For simplicity, I have made the definition work only only when the solution breaks the stone into pieces with unequal weights. As an exercise, modify the definition so it prints the elements using sequences instead of sets, as in

$\langle 7,\ \langle 3\rangle,\ \langle 1,\ 9\rangle\rangle$

so it works if the weights of the pieces are not all distinct.

The definition below uses the following notation:

$\times$ is the *Cartesian* product of sets.

$[w \in 1\ ..\ N \mapsto F(w)]$ is the N tuple with $F(i)$ as element i.

$$ExpandSolutions \triangleq$$

$$
\begin{array}{l}
\textsc{let}\ \mathit{PiecesFor}(w,\, B) \triangleq \textsc{choose}\ ST \in (\textsc{subset}\ (1\,..\,P)) \times (\textsc{subset}\ (1\,..\,P)) : \\
\qquad\qquad\qquad\qquad \mathit{IsRepresentation}(w,\, B,\, ST[1],\, ST[2]) \\
\quad \mathit{Image}(S,\, B) \triangleq \{B[x] : x \in S\} \\
\quad \mathit{SolutionFor}(w,\, B) \triangleq \langle w, \\
\qquad\qquad\qquad\qquad \mathit{Image}(\mathit{PiecesFor}(w,\, B)[1],\, B), \\
\qquad\qquad\qquad\qquad \mathit{Image}(\mathit{PiecesFor}(w,\, B)[2],\, B)\rangle \\
\textsc{in}\quad \{[w \in 1\,..\,N \mapsto \mathit{SolutionFor}(w,\, B)] : B \in \mathit{AllSolutions}\}
\end{array}
$$

Created by *Leslie Lamport* on 26 *October* 2011