(*

A farmer is on one shore of a river and has with him a fox, a chicken, and a sack of grain. He has a boat that fits one object besides himself.

In the presence of the farmer nothing gets eaten, but if left without the farmer, the fox will eat the chicken, and the chicken will eat the grain. How can the farmer get all three possessions across the river safely?

*)

MODULE *boat*

EXTENDS *Integers*, *FiniteSets*

CONSTANTS *Farmer*, *Fox*, *Chicken*, *Grain*

$CREATURES \triangleq \{Farmer, Fox, Chicken, Grain\}$

$alone(animals, side) \triangleq (animals \in \text{SUBSET } side) \land \neg Farmer \in side$

$$somebodyGetsEaten(l, r) \triangleq \begin{array}{l} \lor\ alone(\{Fox, Chicken\}, l) \\ \lor\ alone(\{Fox, Chicken\}, r) \\ \lor\ alone(\{Chicken, Grain\}, l) \\ \lor\ alone(\{Chicken, Grain\}, r) \end{array}$$

$safe(l, r) \triangleq \neg somebodyGetsEaten(l, r)$

$$safeBoats(from, to) \triangleq \{boat \in \text{SUBSET } from : \begin{array}{l} \land\ Farmer \in boat \\ \land\ Cardinality(boat) \leq 2 \\ \land\ safe(from \setminus boat, to \cup boat)\} \end{array}$$

```
**************************************************************************
--algorithm RiverCrossing{

variables left = CREATURES ; right = {} ;

process ( LeftToRight = 0 )
    { l: while ( left ≠ {} )
         { await (Farmer ∈ left) ;
           with ( boat ∈ safeBoats(left, right) )
             {
               left := left \ boat ;
               right := right ∪ boat
             }
         }
    }

process ( RightToLeft = 1 )
    { r: while ( left ≠ {} )
         { await (Farmer ∈ right) ;
```

```
            with ( boat ∈ safeBoats(right, left) )
              {
                left := left ∪ boat ;
                right := right \ boat
              }
          }
      }
}
```

****

BEGIN TRANSLATION

VARIABLES $left$, $right$, pc

$vars \triangleq \langle left, right, pc \rangle$

$ProcSet \triangleq \{0\} \cup \{1\}$

$Init \triangleq$ Global variables

$\land\ left = CREATURES$

$\land\ right = \{\}$

$\land\ pc = [self \in ProcSet \mapsto \text{CASE } self = 0 \to \text{“l”}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square \quad self = 1 \to \text{“r”}]$

$l \triangleq \land\ pc[0] = \text{“l”}$

$\qquad \land\ \text{IF } left \neq \{\}$

$\qquad\qquad \text{THEN } \land\ (Farmer \in left)$

$\qquad\qquad\qquad\qquad \land\ \exists\, boat \in safeBoats(left, right) :$

$\qquad\qquad\qquad\qquad\qquad\qquad \land\ left' = left \setminus boat$

$\qquad\qquad\qquad\qquad\qquad\qquad \land\ right' = (right \cup boat)$

$\qquad\qquad\qquad\qquad \land\ pc' = [pc \text{ EXCEPT } ![0] = \text{“l”}]$

$\qquad\qquad \text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![0] = \text{“Done”}]$

$\qquad\qquad\qquad\qquad \land\ \text{UNCHANGED } \langle left, right \rangle$

$LeftToRight \triangleq l$

$r \triangleq \land\ pc[1] = \text{“r”}$

$\qquad \land\ \text{IF } left \neq \{\}$

$\qquad\qquad \text{THEN } \land\ (Farmer \in right)$

$\qquad\qquad\qquad\qquad \land\ \exists\, boat \in safeBoats(right, left) :$

$\qquad\qquad\qquad\qquad\qquad\qquad \land\ left' = (left \cup boat)$

$\qquad\qquad\qquad\qquad\qquad\qquad \land\ right' = right \setminus boat$

$\qquad\qquad\qquad\qquad \land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“r”}]$

$\qquad\qquad \text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Done”}]$

$\qquad\qquad\qquad\qquad \land\ \text{UNCHANGED } \langle left, right \rangle$

$RightToLeft \triangleq r$

$Next \triangleq LeftToRight \lor RightToLeft$

$\lor$ Disjunct to prevent deadlock on termination
$((\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land \text{UNCHANGED } vars)$

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

$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})$

END TRANSLATION

* Modification History
* Last modified *Wed Jun* 04 21:54:45 *EDT* 2014 by *lorinhochstein*
* Created *Mon Jun* 02 20:41:25 *EDT* 2014 by *lorinhochstein*