Modelling and Analysis of a Cyber-Physical System
Cyber-Physical Programming — Practical Assignment 1

Melânia Pereira Paulo R. Pereira
{pg47520, pg47554}@alunos.uminho.pt

May 26, 2022

The list of adventurers

$$\textbf{data } \mathit{Adventurer} = P_1 \mid P_2 \mid P_5 \mid P_{10} \textbf{ deriving } (\mathit{Show}, \mathit{Eq})$$

Adventurers + the lantern

$$\textbf{type } \mathit{Objects} = \mathit{Adventurer} + ()$$

The time that each adventurer needs to cross the bridge

$$\begin{array}{l} \mathit{getTimeAdv} :: \mathit{Adventurer} \to \mathit{Int} \\ \mathit{getTimeAdv} = \bot \end{array}$$

The state of the game, i.e. the current position of each adventurer + the lantern. The function (const False) represents the initial state of the game, with all adventurers and the lantern on the left side of the bridge. Similarly, the function (const True) represents the end state of the game, with all adventurers and the lantern on the right side of the bridge.

$$\begin{array}{l} \textbf{type } \mathit{State} = \mathit{Objects} \to \mathit{Bool} \\ \textbf{instance } \mathit{Show\ State} \textbf{ where} \\ \quad \mathit{show}\ s = (\mathit{show} \cdot (\mathsf{fmap}\ \mathit{show}))\ [s\ (i_1\ P_1), \\ \quad\quad s\ (i_1\ P_2), \\ \quad\quad s\ (i_1\ P_5), \\ \quad\quad s\ (i_1\ P_{10}), \\ \quad\quad s\ (i_2\ ())] \\ \textbf{instance } \mathit{Eq\ State} \textbf{ where} \\ \quad (\equiv)\ \mathit{s1}\ \mathit{s2} = \mathit{and}\ [\mathit{s1}\ (i_1\ P_1) \equiv \mathit{s2}\ (i_1\ P_1), \\ \quad\quad \mathit{s1}\ (i_1\ P_2) \equiv \mathit{s2}\ (i_1\ P_2), \\ \quad\quad \mathit{s1}\ (i_1\ P_5) \equiv \mathit{s2}\ (i_1\ P_5), \\ \quad\quad \mathit{s1}\ (i_1\ P_{10}) \equiv \mathit{s2}\ (i_1\ P_{10}), \\ \quad\quad \mathit{s1}\ (i_2\ ()) \equiv \mathit{s2}\ (i_2\ ())] \end{array}$$

The initial state of the game

$$\begin{array}{l} \mathit{gInit} :: \mathit{State} \\ \mathit{gInit} = \underline{\mathit{False}} \end{array}$$

Changes the state of the game for a given object

$$\begin{array}{l} \mathit{changeState} :: \mathit{Objects} \to \mathit{State} \to \mathit{State} \\ \mathit{changeState}\ a\ s = \textbf{let } v = s\ a \textbf{ in } (\lambda x \to \textbf{if } x \equiv a \textbf{ then } \neg\ v \textbf{ else } s\ x) \end{array}$$

Changes the state of the game of a list of objects

$$\begin{array}{l} \mathit{mChangeState} :: [\mathit{Objects}] \to \mathit{State} \to \mathit{State} \\ \mathit{mChangeState}\ \mathit{os}\ s = \mathit{foldr}\ \mathit{changeState}\ s\ \mathit{os} \end{array}$$

For a given state of the game, the function presents all the possible moves that the adventurers can make.

$$allValidPlays :: State \rightarrow ListDur\ State$$
$$allValidPlays = \bot$$

For a given number n and initial state, the function calculates all possible n-sequences of moves that the adventures can make

$$exec :: Int \rightarrow State \rightarrow ListDur\ State$$
$$exec = \bot$$

Is it possible for all adventurers to be on the other side in $\leqslant 17$ min and not exceeding 5 moves ?

$$leq17 :: Bool$$
$$leq17 = \bot$$

Is it possible for all adventurers to be on the other side in < 17 min ?

$$l17 :: Bool$$
$$l17 = \bot$$

Implementation of the monad used for the problem of the adventurers. Recall the Knight's quest.

$$\mathbf{data}\ ListDur\ a = LD\ [Duration\ a]\ \mathbf{deriving}\ Show$$
$$remLD :: ListDur\ a \rightarrow [Duration\ a]$$
$$remLD\ (LD\ x) = x$$
$$\mathbf{instance}\ Functor\ ListDur\ \mathbf{where}$$
$$\quad \mathsf{fmap}\ f = \bot$$
$$\mathbf{instance}\ Applicative\ ListDur\ \mathbf{where}$$
$$\quad pure\ x = \bot$$
$$\quad l_1 < * > l_2 = \bot$$
$$\mathbf{instance}\ Monad\ ListDur\ \mathbf{where}$$
$$\quad return = \bot$$
$$\quad l \ggg k = \bot$$
$$manyChoice :: [ListDur\ a] \rightarrow ListDur\ a$$
$$manyChoice = LD \cdot concat \cdot (\mathsf{map}\ \ remLD)$$