Empirical Modelling: River Crossing Problem

Adam Riley
Deparment of Computer Science
University of Warwick

CONTENTS

1 INTRODUCTION

This report will detail the Empyrical Modelling aspects of the River Crossing model. This model has been developed to demonstrate the links between discrete logical problems and the use of graph theory as a solution to such problems. By using the EM environment it was possible to merge these two ideas into a single, graphical format, such that users may more fully comprehend their relationships.

2 MODEL OVERVIEW

2.1 Problem

As describes on the left hand side of the model, this problem involves the transportation of three inter-related objects, in this case a wolf, a sheep and some valuable seeds, from one side of a river to the other. The user, taking on the role of the owner of these items, is tasked with safely moving each objects from the left bank of te river, over to the right bank.

2.2 Constraints

It is common in logic puzzles such as this to include a set of constraints, often similar to real-world outcomes in such situations, in order to increase the problems complexity.

In puzzles such as the 'Jugs' problem, also modelled in the EM environment, the constraints are implied, for example, the user cannot over-fill a jug. The size of the provided jugs acts as the constraints for this exercise. In the River Crossing problem the constraints are set out thus:

1) Whilst on the same bank, the user can control the behaviour of each of the items

of cargo.

2) Cargo can only be transported when the raft and the cargo are on the same bank.
3) If left alone on the same bank, the wolf will eat the sheep.
4) If left alone on the same bank, the sheep will eat the seeds.
5) The raft used to cross the river can hold only the user and a single piece of cargo.
6) The user may choose to take no cargo across the river on any given trip.
7) The problem is solved once all 3 items of cargo reach the far bank.

These contraints form the basis of this problems complexity and, given the discrete number of steps, aloow for the user to compelte the scenario in a finite number of steps. This property of the model makes it suitable for a solution using graph theory. Given the low number of variables in the game, the graph space is small enough to be interpreted easily during gameplay to understand the current state of the environment and all available moves.

2.3 Visual Cues

The visual aspect of this model has been designed with simplicity in mind. The background environment, representing the river and two banks, is modelled with 3 rectangles. The central rectangle is blue, to signify the water, with two green rectangles on either side to represent the banks.

The 3 pieces of cargo, modelled in this game as a wolf, sheep and seeds, were designed to be recognisable to the user without the need for labels. To this end, their shape and colours have been chosen with maximum recognisability in mind.

The raft, used to transport the cargo from one side ot the river to the other, is modelled using a common wodden log structure. This is acheived with a single brown rectangle segmented by black lines at uniform intervals.

All interaction by the player is through the use of the five buttons at the top of the screen. Each of these buttons will remain active whilst the player remains in an allowed state. If the use makes an erroneous move, causing one of the pieces of cargo to be destroyed, all buttons except the 'Reset' button will be rendered inactive. This is to guide the playe through the scenario and ensure that the constraints of the game are met and that a solution is always available. When inactive, these buttons become 'greyed out' and are visually and physically unavailable to the player.

3 Observables

As it was built in the js-eden environment, all of the observables are defined in the js-e language. This allws for the entire model to be defined in a consistent format, without the use of the %scout notation.

3.1 Environment

The environment, as described above, is modelled using coloured rectangle, to help the user to distinguish between the each of the banks and the river.

The environment is also sized appropriately for most modern computers to be able to view. By sizing the game to be approximately 1024 pixels across, there are a very high percentage of monitors able to view the entire screen, and play in their native resolution, allowing for maximum accessibility.

3.2 Animals

Each animal is a compound model, represented by a compination of shapes in varying sizes and orientations. Using the wolf as an example, this animal is comprised of ten observables. The combination of these variables is designed to allow the user to associate the final observable with its intended representation, in this case, of a wolf.

This approach has been taken with each of the items of cargo such that the user will be able to distinguish one from another. As the items of crago are unique and finite, it is more important for the user to be able to differentiate between each of the items than for them to be accurate representations. For this reason, the animal observables were designed to be a compromise between recognisable and unique.

Each animal is also linked to the shared x position vlaues 'animalRight' and 'animalLeft' through a position variable. These represent the positions of the animals when on each of the banks. Again, as stated previously, the distinction between the two available states of the animals ('on left bank' or 'on right bank') is more important than their relative positioning, therefore it was decided that they should share these variables in order to be consistent.

The placement of these items of cargo is also important. As the constraints of this model state that the items of cargo may be a danger to each other, this has been signified on the game with adjacency. Leaving two items of adjacent cargo on a bank unattended will end the game. This allows for a level of abstraction. Even if the player were unable to distinguish between the items of cargo, they may infer the constraints of the game by observing the positions of the cargo during their failure states.

These positions also correspond to the positions of the buttons used for interaction. The animals are positioned Wolf, Sheep Seeds, running from top to bottom, whilst the buttons are positioned Take Wolf, Take Sheep, Take Seeds, from left to right. This again, allows the player to infer the use of the buttons simply from their relative positions.

3.3 Raft

Similar to the animal representations, the raft is a compond object comprised of a rectangle and multiple lines. The x position of each of these shapes is also controlled by a shared variable named 'raftPos'. This variable, alternated whenever a successful transfer of cargo (or an empty journey) is completed, moves the raft from a position on the left bank to a position on the right bank and vice versa.

3.4 Graph

This area, located at the bottom of the screen, represents the current state of the game, and all legal state transitions such that the goal remains acheivable. If the user makes a transition not signified on the graph as a transitions from the green node (curent state) to an adjacent red node (legal states), the game will terminate with failure.

The state transition diagram is represented with nodes, depicted as circles, and transitions, depicted as lines. Each line represents a relationship between states which equates to a single button press by the user. The transitions are bi-directional and, as such, are not labelled with arrows.

The green node allows the user to identify their current state in the state transition diagram and each adjacent node to the green node, signifies each possible move for them. The player is free to make transitions from left to right and right to left, but the goal state is located on the far right of the state transitions diagram, labelled 'R:R:R:R'. This label, as descirbed in the models legend, signifies that each of the items of cargo and the raft in on the right hand bank, therefore, the game has reached the goal state.

4 Procedures

Due to its simplicity, this game requires procedures only when a button press is made by the player. Each of these procedures is written to ensure that none of the game constraits can be violated and, if that is the case, to either disallow the action, or signify the end of the run and require a reset.

This ensures that the player can enter some of the incorrect states available to the game, but connot violate the fundamental rules of the problem, such as using the raft to transport cargo when they do not share a bank.

5 Graph Theory

Is is possible, for all finite state games, to construct a styate transition diagram which details all available 'moves' that a player can make, and which lead to failure states. This allows players to identify optimum solutions for games and if possible, avoid failure entirely.

In small games, such as the one presented here, and similar games, with small nukmbers of possible states, graph theory can rove to be an invaluable approach. This model aims to introduce this relationship to the player by not only providing a partial state transition diagram for the player to view whilst finding a solution, but also identifying the players current state on the diagram as a green node.

By presenting the player with the solution in this way, they may more fully understand the inter-relationships of logic problems such as this one, and their corresponding state transition diagrams.

6 Conclusion

This model was developed with learning in mind. By visually representing the graph theory which can be used to form a solution to logic problems, users may be able to further understand their use and formulate a similar solution to other problems.

The visual cues in the game are designed to be easily distinguished by the players, holding relative positioning and colour as a high priority, such that the player may more asily understand the model and focus on the relationship between the problem and the graph.