An Empirical Approach to Mastermind

0705387

Abstract

This project aims to build an Empirical model of the game of Mastermind and to visualise the set of inferences that can be made from the results of the player's guesses. In doing this, we can explore all possible inferences that can be made about the state of the game at every move. From the perspective of the end user, this helps to improve their playing strategy through experimentation and experience. The prominent use of dependencies within the model highlights the power behind Empirical Modelling. If the user of the model adopts the role of a modeller, observables can be changed at will thereby providing an insight into how the model works, which helps the modeller gain a better understand the dynamics of the game of Mastermind in addition to the logical thought process that is required in order to break the code.

1 Introduction

Empirical modelling is an area of study concerned with addressing the issues of building an understanding of a problem. In classical computer science, there are many examples of the scientific application of problem solving using mathematic theory and computing technology to find a set of results. While classical computer science does well to enhance this side of computing, computers are also often used to aid in the understanding of complex physical and theoretical systems through modelling. In order to increase a user's understanding of a problem it is necessary to build models of the environment in which paths through the complex system can be explored by either the computer or the user.

The art of building a realistic model of a complex system is of critical importance to the validity of the results obtained through the solution of the model. Using a poor model to find an optimal solution to a complex problem rarely results in the same solution proving to be optimal in the real world – this is often due to the fact that the model is an over simplification of the complex system. The problem worsens when the system is so complex that behaviours within the system are not always defined. This means that the modeller cannot possibly define a realistic model through which to draw understanding without undergoing some kind of iterative and dynamic building process.

Empirical modelling is useful because it helps the modeller build a greater understanding of how the model responds to given stimuli through experimentation and exploring observables with the model. This submission is concerned with building an intuitive model of the classic code-breaking game, Mastermind.

2 Mastermind

Originally popularised in the 1970s by Mordecai Meirowitz, Mastermind is a two-player board game comprising of a board of 9-10 guess slots where each guess slot has space for 4 code orbs and 4 score pegs. There are an ample supply of code orbs (separated into 6 different colours; red, yellow, green, blue, orange and purple) and score pegs (separated into 2 different colours; red and white).
One player is assigned as the code-maker, while the other takes the role of the code-breaker. Initially the code-maker chooses a code by secretly selecting 4 code orbs of any colour they like and place them in any order they like into a code-bay that's hidden from the code-breaker. This code remains unchanged for the rest of the game and serves as the 'secret code' that the code-breaker has to attempt to match (or 'break') by using the remaining code orbs to submit guesses that are then scored (using score pegs) by the code-maker.

Scores are awarded in the following manner; a red score peg is awarded for every code orb whose position and colour match the secret code, and a white score peg is awarded for each code orb guess whose colour is contained within the secret code, but is positioned incorrectly in the guess (note: a code orb in the correct colour and position is only given a red score peg, not a white one as well). The score pegs are placed beside the guess by the code-maker in any order they wish so that it is impossible to infer anything about which score peg relates to which guess orb directly from its position.
Using previous guesses and their resulting scores, code-breakers have to derive the secret code before they run out of available guesses.

3 The Model

This project aims to model a fully functional playing environment for the game of Mastermind. The user is able to play as the code-breaker against a randomised computer opponent and submits guesses using the %mastermind syntax within the tkeden interpreter. The display updates with each guess submission, automatically scores the guess against the secret code and calculates the remaining possible guesses through a process called inference generation.

Due to the dynamic and exploratory nature of Empirical Modelling, and by extension tkeden, a modeller or user is perfectly able to interact with the model through observing the model's variables and changing them as they see fit. This opens up the possibility of cheating since the user could observe or alter the secret code at any given moment. While for the average user of the model this behaviour may not be very entertaining, it certainly can demonstrate the power behind tkeden's dependency system. Despite this, the majority of this model's variables are entirely dependent on the user submitting a guess – upon which there is a large series of functions that are executed in a linear fashion (as can be observed from the source code). From a programmer's perspective, this model serves as a bridge between the world of classical computing and Empirical Modelling due in part to this significant linear element, but also owing to the fact that all display elements are automatically updated (in a procedural manner) upon the change of 'logical' observables.

In addition, the game has the capacity to be readily extended upon simple modification of 3 related observables within the source code that allows a user to increase or decrease the difficulty of the game. This is achieved by redefining the colours available for the code orbs. Initially the model is set up to use just 6 colours however this can be increased to a maximum of 12 unique colours by means of adding extra colours to the "colours" observable, adding relevant single character colour labels to the "shortColours" observable, and finally by adjusting the number of colours in play ("numOfColours").

This extendible element, when combined with its interrelated dependencies, allows the model to grow (or shrink) correctly beyond the normal realms of the game of Mastermind to provide variations. This therefore affords validation that the Empirical Modelling approach to building models has desirable behaviour in terms of a model's future scalability since dependencies are maintained regardless of changes to the model's observables.

4 Inference Generation

One of the primary motivations behind modelling the game of Mastermind stems from the fact that Mastermind's search space, while is well defined and known, is sufficiently large enough to render it impractical for a human player to manually search the space for a solution in the same way that a computer might. As such, a human player exhibits very different behaviour to that of a computer when attempting to break a secret code. In particular, unlike a computer, a human will sometimes (all be it logically) use up guesses to confirm a hypothesis by submitting a guess (which perhaps only changed from the previous guess by a single orb) that lies clearly outside the remaining combination space through considering previous guesses and respective scores. Overall, this specific phenomenon turns out to be not only an expensive move (when you consider the fact that there are only nine guesses available, and at least two of those guesses – the current and previous guesses – have been used to establish only one of the orbs in the secret code), but also a move that seems altogether illogical (since by only swapping a single orb between the current guess and the previous, the user is only able to, at best, gain information about that single orb in the secret code, whereas a more aggressive strategy dictates that multiple orbs be switched actually increases the probability of information gain).

Therefore, through the deliberation of computer generated inferences, a player can interact with the Mastermind model to see if they can improve their playing style.

When the code comprises of 4 orbs of any 6 colours, the search space of the game of Mastermind contains 1296 different possible combinations/secret codes.

Depending on the results obtained for each guess the code-breaker submits, a computer can eliminate possibilities from the search space by comparing each remaining possibility in the search space against the submitted code and its resulting score. Any combination whose score, when compared with the latest submitted code, is identical to the latest score received is intuitively still a remaining possibility, as discovered by (Rao, 1982). Conversely, if the scores don't match, the computer eliminates the combination and advances to the next. Eventually, and after multiple guesses from the code-breaker, there will only be a single remaining possibility – this will be the real secret code.

To accomplish this behaviour in tkeden the computer has to store the entire search space (at least initially) for combination checking. Due to memory space limitations within tkeden it was necessary to

utilise two separate 'working' files with which to compute the remaining combinations. When the game loads, the first working file is filled with all 1296 possible code combinations. When a user submits a guess, the computer iterates through all combinations computing their score against the submitted guess. If the result matches the result the code-breaker received, that combination is written to the second working file.

Once all iterations have occurred, the second working file will contain all possible combinations remaining. At this point, a Boolean switch is flipped to indicate that the second file now contains the remaining search space. Upon submission of the next guess, tkeden uses the switch to determine which file the search space is in, and begins checking through the possible combinations in the same way as before, except this time writing out remaining combinations to the other file. Using these stored search spaces, tkeden can iterate through the relevant file and display the remaining possible combinations to the user in the form of an inference port. For each of the 4 orbs within the code, a string (containing colour code characters) is shown that represents what remaining colours that orb could be.

Using these inferences, the player can then try to confirm their validity to themselves by considering the past set of guesses and scores (which helps to identify new ways of playing), and provides hints on the next guess that could be made. Indeed, the author benefitted from this over several games.

5 Comparison to Other Studies

A study into the use of Empirical Modelling for the construction of the game of "Hunt the Wumpus" – a turn based single-player game invented by (Yob, 1976) in which the player uses information regarding his senses to discover the location of a monster, the Wumpus, to which he is tasked with shooting – by (Cole, 2005) highlighted how logical inferences generated through a player's exploration of "Wumpus World" can be used to develop their knowledge of the game's state, and help them improve their game. On the same basis, this study explores these ideas with respect to the game of Mastermind.

One major difference between the models built was that, in the "Hunt the Wumpus" model, a player was able to hide the game map and play solely using the inferences generated. Making this addition to the Mastermind model, however, seems counter-productive since the player relies on the visual to see the results and guesses he has previously made, which he requires to draw conclusions from his past moves.

Modelling for the purpose of developing an expandable testing and creating platform for the "Duko" set of games, including "Sudoku", researched by (King, 2006) successfully demonstrated the benefits of utilising Empirical Modelling technology in the development and playing of software games through the addition of expandable elements. Similarly, it is this behaviour employed in the Mastermind model that enables users to experiment with the game in whatever way they see fit, which may otherwise not have been possible using different software development tools and methodologies.

6 Conclusion

In conclusion, this study has further shown in support of (Cole, 2005) that, through the use of inferences that one can generate based on past events within a game, a player can learn and improve their playing through exploratory actions within the game world.

Furthermore, this study helps to promote the use of Empirical Modelling technologies in building both the user's and the modeller's construal of the game of Mastermind through experimentation with its observables, thereby supporting the work of (Pope & Beynon, 2010).

Finally, we provide evidence that Empirical Modelling tools have a benefit over other software development methods in creating models that are fully expandable due to the inclusion of dependencies that are upheld with respect to varying observables.

References

Cole, G. (2005). *Hunt the Wumpus: An Empirical Approach.* University of Warwick Empirical Modeling Library.

King, K. (2006). *Is Empirical Modelling Puzzling?* University of Warwick Empirical Modeling Library.

Nelson, T. (1999). *Investigations into the MasterMind Board Game*. Retrieved December 15, 2010, from http://www.tnelson.demon.co.uk/mastermind/

Pope, N., & Beynon, M. (2010). *Empirical Modelling as an unconventional approach to software development.* ent. Proc. SPLASH 2010 Workshop on Flexible Modeling Tools, Reno/Tahoe Nevada, USA.

Rao, T. M. (1982). *An Algorithm to Play the Game of Mastermind.* SUNY College at Brockport, NY: Dept. of Math/Computer Science.

Yob, G. (1976). Hunt the Wumpus: The Genesis of Wumpus. *The Best of Creative Computing*, 1:247–250.