Evil Games

Robert Chuchro

Problem Statement

In this project, we introduced a template for modeling extremely hard instances of games, also known as "Evil" games. We demo our solution by using logic programming to build a Worksheet for Evil Mastermind.

Games that can be Evil

We say that a game can be modeled as an evil game if it has the following properties:

1. Each instance of a game is a randomly selected answer from a finite set of possible solutions
2. On each turn, the player guesses a solution to the game
3. The Game Engine/Manager provides feedback on correctness of proposed solution
4. Game continues to loop (2) and (3) until the player has correctly guessed solution

Some examples of well known games that fit this criteria are Hangman, Wheel of Fortune, Safe Cracking Puzzles, and Mastermind.

How to Make a Game Evil

In order to play an evil version of a game, the game is modified as follows:

1. The game never generates a random solution at the beginning of the game.
2. Instead, maintain a set of valid candidate solutions.
3. After each guess by the player, give feedback that removes the fewest number of candidates from the set of viable solutions.
4. This guess and feedback loop continues until the player is able to reduce the set of candidates to just 1 viable solution. When the player makes this guess, they have finally won.

The motivation for playing evil versions of games goes beyond the thrill of playing a more challenging game. Removing the inherent randomness involved in each instance of a game makes each game more fair. In the original versions of these games, the performance of any particular instance often comes down to luck when there is no clear best guess on a given turn. In addition, the game now becomes completely deterministic as there is no need to randomly generate a solution to start an instance of a game. This allows evil games to be modeled using Game Description Language (GDL) for General Game Playing.

Mastermind

In this project, we focused on building a demo for Mastermind using logic programming. In Mastermind, there are K slots and C colored balls. It is a single player game where the goals of the game is to find the correct permutation of placing at most 1 color of each ball into the K slots.

At each turn, the player guesses a permutation and the game engine returns two pieces of information:

- The number of balls which are the correct color
- The number of balls which are the correct color and in the correct slot

The goal of the player is to guess the correct permutation in the fewest amount of attempts.

Solution

To play the evil version of Mastermind, the game engine generates all possible permutations as candidates at the start of the game.

When the player makes a guess, the game engine considers for each possible hint, how many valid candidate solutions remain that are consistent with the hint. It then chooses to give the hint that maximizes the number of remaining candidate solutions. After choosing a hint, the engine always maintains a set of candidate solutions that is consistent with the hint that was given.

Logic Programming

A major reason logic programming was used to build Evil Mastermind is because of the aggregate operator *setofall*. Using this operator we are able to find the set of all candidate solutions that are consistent with a given hint, which is the main computational challenge of this problem. Each hint can be viewed as a constraint which is modeled intuitively using logic programming.

In addition, we added features to the user experience such as active feedback, indicating how many valid solutions remained after a given guess. Using Worksheets, all the information was readily available as each candidate is simply stored in the lambda and we use another aggregate operator *countofall* to get the number of remaining candidate solutions.

Another reason we used Worksheets is to be able to easily develop an interactive front end user interface as part of our solution. With any other programming language, tying a backend program with a GUI is not a simple task, while Worksheets is a natural extension of Epilog which allows the developer to build a stylesheet which has immediate access to information from the underlying logic program.

Future Work

As mentioned earlier, evil games are completely deterministic which allows it to be modeled as a game for General Game Playing. A future project can be to design a two player game, where one player plays a single player Evil game, while the other player acts as the adversary giving hints. The reward would be a function of how quickly the player is able to force the adversary to admit a solution is found.

In addition, it would be interesting if the game engine was also able to compute what the best guess for a player could have been in a given state – a guess that results in the fewest candidate solutions remaining. The major challenge here would be finding an efficient way to enumerate all possible guesses.