Nathan Korzekwa
Dr. Goel
CS4635
Project 2

What is the *Meaning* of MEANING?

Introduction:

After spending much time restricting myself and trying to come up with a formal predicate-based language for domain knowledge (much in the style of GDL), I looked back on my last project and realized that much of the game-domain knowledge was already factored out, in a sense. As a consequence, I re-evaluated how I could do the factoring, and I found it easiest to express the knowledge in a class-inspired structure, like frames. Not only does this express the knowledge easily, it also translates into code very cleanly.

How to Run:
Simply go to the source directory and execute "python engine.py."

Architecture:

The architecture of this project is split into three parts: *Game Domain Knowledge, Game Analysis Tools,* and *Agents.* The Game Domain Knowledge is what we were asked to "factor out" in this project, and, as such, it represents all knowledge about the game itself. The Game Analysis Tools are specific tools that measure desirability or other metrics of a certain game state, and have methods tailored to specific Agent types. Lastly, the Agents themselves are pretty self explanatory; they behave purely to maximize their returns relative to some metric. In this particular setup, the Thoughtful Agent simulates a full game-tree of Tic-Tac-Toe, and the Naive Agent simulates a game tree of depth one, using a heuristic function to determine its move. The details of this particular heuristic function are unchanged from last project and will be discussed later.

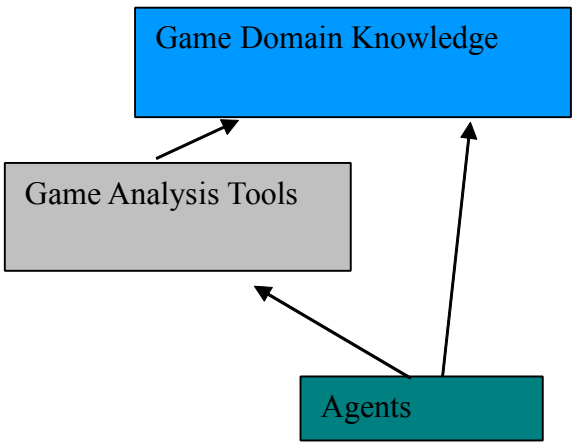

Figure 1: Architecture Diagram

Game-Level Ontology:

As mentioned earlier, I have created an ontological representation of the game state that is heavily inspired by frames and classes. This ontology is realized in the *Game Domain Knowledge* section of the program's architecture, and it can be broken down into three main parts:

1. The Context frame
2. The Role frame
3. The 'move' tuple

The Context Frame is the driving force behind the entire system, as it contains the vast majority of the information that agents and evaluation functions care about.

Name	Context
Ground	3x3 Matrix
Play	Constructor
Winner	String
Legal Moves	Set

As seen above, the Context Frame is an instance Frame that has four principal slots. The first, 'Ground' is simply a matrix which defines the state of the board itself. This is in turn used in the Mid-Level Ontology (described in the next section) for move and state evaluation. The 'Play' slot is a constructor that creates a new frame from the current frame, using a prototype inheritance scheme, based on a move that a player makes. The 'Winner' slot provides the winner of the game through a when-requested daemon that analyzes the game, and finally, the 'Legal Moves' slot returns a list of *moves* also through a *when-requested* daemon.

The other main parts are pretty simple and self-explanatory: the Role frame describes a player, and determines if it is that player's turn; the Evaluation function determines the desirability for a particular Context; and the move tuple is simply a coordinate pair (row, column).

The Agents:

The Agents themselves -- in this case Thoughtful and Naive -- are pure implementations of game logic and reasoning, however in different implementations they may depend on game-specific heuristic functions. Thoughtful is very thorough, and it simulates a game tree to the true terminal states not relying on heuristics, in this implementation. Naive, is not thorough at all and it only simulates one half a ply before making it's decision. In this

case, the heuristic function rates moves that complete long chains of moves higher than ones that don't complete chains or complete shorter ones.

Domain-Knowledge Example:
Examine the Frame below:

Name	Context
Ground	$\begin{bmatrix} X & X & \_ \\ \_ & O & \_ \\ O & \_ & \_ \end{bmatrix}$
Play	Constructor
Winner	'None'
Legal Moves	{(1,0), (2,1), (0,2), (1, 2), (2, 2)}

In this state, the role 'X' is active, and as such, the agent assigned to that role moves next.

If the Agent is Naive, then if would run the appropriate heuristic from the Game Analysis Tools on each of the legal moves in the context. In this case, the move with the highest heuristic score is (0, 2) with a score of 2. Following that is (1, 0) with a score of 1, and then the rest with a score of zero. Since Naive is a max player, it selects (0, 2) from the moves and constructs a new Context frame with this information, producing:

Name	Context
Ground	$\begin{bmatrix} X & X & X \\ \_ & O & \_ \\ O & \_ & \_ \end{bmatrix}$
Play	Constructor
Winner	'X'
Legal Moves	{}

If the Agent is Thoughtful, it will make the same decision, but without relying on a heuristic function, and all of the moves *except* (0, 2) will resolve to a -1 score, since all of the other moves result in losses.

Analysis of Reasoning Traces:

The reasoning traces are quite straightforward. They simply output what move the agent took at each turn, and why they took that move. For example, examine the traces for two Naive Agents pitted against each other:

```
[X]: In turn 0, I played at (1, 2) because it had a heuristic
score of 0, although it was randomly chosen from among ties.
[X]: In turn 2, I played at (0, 0) because it had a heuristic
score of 1, although it was randomly chosen from among ties.
[X]: In turn 4, I played at (0, 2) because it had a heuristic
score of 1, although it was randomly chosen from among ties.
[X]: In turn 6, I played at (1, 1) because it had a heuristic
score of 2, although it was randomly chosen from among ties.
[X]: In turn 8, I played at (2, 1) because it had a heuristic
score of 2, the absolute best of all the legal moves that turn.

[O]: In turn 1, I played at (2, 2) because it had a heuristic
score of 0, although it was randomly chosen from among ties.
[O]: In turn 3, I played at (0, 1) because it had a heuristic
score of 1, although it was randomly chosen from among ties.
[O]: In turn 5, I played at (1, 0) because it had a heuristic
score of 1, although it was randomly chosen from among ties.
[O]: In turn 7, I played at (2, 0) because it had a heuristic
score of 2, although it was randomly chosen from among ties.
```

Due to the way I modded the board in finding the longest chains, there is a small bug/feature in which there are a lot of tie scores generated. This is a non-issue for the most part, but it does produce more monotonous output in the reasoning trace. Naive Agent's reasoning is basically a depth-one instance of Minimax, with a crude heuristic function to mock as terminal states, and as such, the reasoning traces are simple.

The reasoning traces for the Thoughtful Agent, a full-depth instance of Minimax, however, are a bit more complicated.

Examine:

```
[O]: In turn 1, I chose to play at (1, 1) because a smart opponent
would have played at (0, 1) with a score of 0, in response to
my ideal play at (0, 2). I moved there because a smart opponent
would have played at (2, 0) with a score of 0, in response to
my ideal play at (1, 0). I moved there because a smart opponent
would have played at (1, 2) with a score of 0, in response to
my ideal play at (2, 1). I moved there because a smart opponent
would have played at (2, 2) with a score of 0
[O]: In turn 3, I chose to play at (0, 2) because a smart opponent
would have played at (2, 0) with a score of 0, in response to
```

```
my ideal play at (1, 0). I moved there because a smart opponent
would have played at (1, 2) with a score of 0, in response to
my ideal play at (2, 1). I moved there because a smart opponent
would have played at (2, 2) with a score of 0
[O]: In turn 5, I chose to play at (1, 0) because a smart opponent
would have played at (1, 2) with a score of 0, in response to
my ideal play at (2, 1). I moved there because a smart opponent
would have played at (2, 2) with a score of 0
[O]: In turn 7, I chose to play at (2, 1) because a smart opponent
would have played at (2, 2) with a score of 0
```

This output illustrates Minimax's reasoning all the way down to the bottom move. You'll notice that all of the 'scores' at any given time are zeros. This is because the output came from a Thoughtful vs Thoughtful game where both players were optimal and assumed, correctly, that their opponents were optimal. It shows the back-and-forth action of maximizing its own choice in relation to a minimum choice that it's opponent would make, in natural language.

Differences from Previous Project:

To be completely honest, I didn't do a very thorough job in my analysis last time. In the last analysis, however, my primary focus was to evaluate the performance of my agents from an outside perspective, without regard to form or elegance. This time, however, *how* knowledge is represented and *how* an agent came to its moves is much more important.

This time, instead of merely listing some statistics, we look deeper into what makes a good system, and we examine why the agents behave the way we do -- all in a complete ontology defined earlier.

Also, as a note, I fixed the problems with the last assignment -- the Thoughtful v. Thoughtful setup always ties now, showing that they are, at last, optimal.