# B351 AI Project: Texas Hold ’em

**Steven Myers and Samuel Eleftheri**

*Indiana University, Bloomington, IN, USA*

May 5, 2017

**Our final project was to build a working Texas Hold’em Poker emulator, and an AI agent to play the game efficiently. Our approach was to accurately emulate poker as it is played professionally with realistic betting systems. We built an agent that can effectively determine the best move given its cards and river by evaluating the strength of its hand by comparing its relative strength to all other possibile hands. The agent is able to determine the probabilities of**

## 1 Introduction

Creating an agent for a game needs to be a well thought-out procedure. ”Agent-based AI is about producing autonomous characters that take in information from the game data, determine what actions to take based on the information, and carry out those actions.” (**Reference1**). This reference can make the process seem simple, but can be (and will be) incrediably complex. Simpler games will contain a initial state, a set of possible actions, the repercussions of the actions, and the goal state. Understanding this information is crucial to creating an intelligent agent for a game. We must be prepared for any unknown information if we are dealing with an unknown environment. After dealing with the information, we need to recognize what best action would reach the goal the quickest.

We can use chess as an example. In early development of intelligent agents, Chess was often used as a game to anaylze the performance of agents. Chess was a popular choice since the environment is known. Players cannot decieve each other since all the information is on the board. Since chess has a large amount of possible actions, early agents needed a effective search algorithm to find the best move. The agent must be able to simulate a numbers of possible moves ahead in order to react to the other player’s actions.

Implementing agents for games with unknown environments presents a new challenge. Predicting which action to perform with insufficient information requires a more observant agent.

| | |
|---|---|
| Pair | 3♥,$A$♦,3♠,4♣,$K$♥ |

## 2 Playing Texas Hold ’em as an AI Problem

Since Texas Hold’em has unknown information, it is important to develop a attentive agent that learns from previous rounds and a opponent’s actions. An agent has to consider the significance of its information and understand how strong of a impact it can make. Some of our information includes:

1. Known Information:
   - Player’s Cards
   - Current River Cards
   - Deck
   - Pot Size
   - Opponent’s Money
2. Unknown Information:
   - Opponent’s Cards
   - Future River Cards
   - Bluffing Probability

With the known information of our player’s cards and the current river cards, an agent can figure out how valuable our hand is. This is an essential procedure as it will greatly influence what action we execute. A higher value hand can result in the action of betting, when a lower value hand can result in folding if challenged by another character. Agents can then enhance the choice making algorithm by including unknown information. Agents can perform certain calculations to estimate

**Table 1:** *Example table*

| Name | | |
|---|---|---|
| First Card | Second Card | Probability |
| $A$♥ | $A$♣ | $P(A♣\|A♥)\,P(A♥)$ |

the chances that certain cards can be drawn, or the probability of opponents having a certain card.

Other players' actions will greatly influence an agent's decision algorithm. Having an agent that can understand an opponent's actions can be the difference between winning or losing money.

## 3 Code

The first step we took into coding our strategy was implementing a hand identification system. We began by creating a list of numbers which would represent the deck, a list of suits, and a dictionary of hand rankings to sort which hand is better. We then built a method that would take any amount of cards and return the best hand ranking. The method included a number of data structures that sorted and counted pairs.

## 4 Results

Discuss the result of your project – what went right and what went wrong. For the former, what would you do to make it even better? For the latter, what would you do differently?

The results of the project was not impressive, as much of the game's information was not used appropriately. The agent lacked the core ability to identify and apply both known and unknown data. It's intelligence for creating a probability for which action to performed was horridly implemented, due to my (Samuel Eleftheri) lack of preparations.

The things that went right was establishing an environment to implement our agent's performance. No problems were detected with recognizing hand ranking, perform the set of actions given to the players, and implementing multiple players into a game. The Player and Table classes recorded the cards and chips data efficiently.