Modelling Go Fish using PLCC: Comparing Classic Heuristics and Deep Learning through a Bayesian Implementation

Artur Habuda s233190 | Frithjof Sletten s204477 | Mario Vicente s230243 | Stine Madsen s204425

December 5, 2024

1 Introduction

Propositional logic has long been a tool in many areas of knowledge; for theorem proving in mathematics, the analysis of arguments and their logical validity in philosophy, and representing and manipulating logical expressions, used in control theory or programming. Moreover, epistemic logic, which builds upon atomic propositional logic, and provides the tools to reason about knowledge, has further expanded these applications. These tools have been hypothesized to serve well the function of building blocks of artificially intelligent systems and their environments [1]. Games provide the perfect controlled playground to test this conclusion. Epistemic logic is especially relevant in games with imperfect information such as Go Fish which we will focus on in this paper. Go Fish is a multiplayer card game where players compete to collect sets of four cards with the same rank, known as "books". Like many popular games, Go Fish has several rule variations. We will consider two-player games with the following rules[2]:

1. Each player is initially dealt a hand of 7 cards.
2. A player can only ask their opponent for cards of a particular rank if they hold at least one card of that rank.
3. If the player guesses correctly, the opponent must give them all their cards of the requested rank, and the player gets another turn.
4. If the guess is incorrect, the opponent rejects the request by saying "Go Fish!"
5. In this case, the player "fishes" by drawing a card from the remaining deck, and it now becomes the opponent's turn to ask.
6. If a player runs out of cards during the game, they must draw a card from the deck.

After this introduction the rest of the paper follows our contributions:

- In section 2 we present the concepts of Epistemic logic, Dynamic Epistemic Logic and Probabilistic Logic of Communication and Change that are essential for understanding the project.
- In section 3, we use PLCC[3] to model a two-player game of Go Fish, which uses a Bayesian Kripke Model to capture the probability of each player's hand. We also define the Event Model, which updates the Bayesian model after each event. Finally, we provide an example demonstrating how these models are applied using a small deck of cards.
- In section 4 we present the alterations made to the model that make it possible to simulate the game. This section also introduces the Deep learning algorithm that we will compare our heuristics against.
- In section 5 we provide a brief description of all the heuristics implemented for this paper as well as an analysis of the outcomes of the benchmarking including all heuristics and the Neural Network.

Finally, we conclude and discuss future work in section 6.

2 Background

Epistemic Logic

Epistemic Logic is a branch of modal logic that formalizes reasoning about knowledge and belief in multi-agent systems. It introduces several key concepts such as:

- **A formalization of the idea of knowledge**: which is defined as the set of prepositions that hold in every possible world.
- **The K_a operator**: which is a modal operator that expresses the knowledge of a given agent "a".
- **The concept of Mutual Knowledge**: that includes in multi agent scenarios both the ideas of

Common and Distributed knowledge (the knowledge shared by all agents and their collective knowledge, respectively).

- **A set of axioms and properties**:
  - Truthfulness of Knowledge:
    $K_a\phi \implies \phi$
  - Introspection of knowledge:
    * Positive:
      $K_a\phi \implies K_aK_a\phi$
    * and negative:
      $\neg K_a\phi \implies K_a\neg K_a\phi$

Dynamic Epistemic Logic

Dynamic Epistemic Logic is an extension of Epistemic logic that models how knowledge and beliefs of agents change due to events, actions, or communications.

In the game of Go Fish for example there are two elements that affect the player´s knowledge:

- Asking for a rank allows the enemy player to know that you posses it but it also means that he will have to give any elements of that rank he was holding onto to you.
- Drawing a card also updates the belief of the agents although like in many card games, these beliefs are represented by a probability distribution over the cards.

Probabilistic Logic of Communication and Change

Probabilistic Logic of Communication and Change or PLCC is a framework that combines both DEPL (Dynamic Epistemic Probabilistic Logic) and LCC (Logic of Communication and Change). According to it´s creators, PLCC "captures in a unified framework subjective probability, arbitrary levels of mutual knowledge (including common knowledge) and a mechanism for multi-agent Bayesian updates that can model complex social epistemic scenarios."[3]. Meaning that it has the potential to model probabilistic uncertainty in multi agent environments even when working with epistemic logic including mutual knowledge, thus making it a really interesting framework for our study case.

PLCC uses several key constructs such as **Bayesian Kripke models**, **substitutions**, **event models**, and the **product update rule**.

- **Bayesian Kripke models** form the backbone of PLCC, representing agents' knowledge and beliefs via states, indistinguishability relations, and subjective probability distributions (Refer to Appendix B Definition 1).
- **Substitutions** are functions that update the valuation of atomic propositions to reflect factual changes in the environment (Refer to Appendix B Definition 2).
- **Event models**, define a set of possible events, their preconditions, and subjective occurrence probabilities, capturing the dynamics of communication or interaction in multi agent environments (Refer to Appendix B Definition 3).
- **The product update rule**, that integrates Bayesian conditioning with epistemic updates, constructing new models that account for changes in both beliefs and knowledge as a result of events (Refer to Appendix B Definition 4).

The semantics of PLCC formalize how formulas are evaluated in this setting, ensuring consistency and capturing the interplay between static knowledge and dynamic updates (Refer to Appendix B Definition 5).

3 Modelling

Bayesian Kripke Model

We start by defining the static Bayesian Kripke model per definition 1 used to represent each step of the game. First, we need a way of describing how cards are distributed among the players. Based on the game's rules, each player's primary focus is on the number of cards of each rank in the opponent's hand or the deck. Making the suit of each card irrelevant. As a result, each player's hand and the deck can be represented as a multiset. To encode this in our model, we introduce the atomic proposition Rr_i^k which denotes "player i has k cards of rank r". We will now impose constraints to ensure that each card originally in the deck is counted exactly once. To begin, we define the parameters of the game:

- The ranks in the deck $N = \{1, \ldots, n\}$.
- The multiplicity of each rank $M = \{0, \ldots, m\}$ where m denotes the size of each suit.
- The players $P = \{0, 1, 2\}$ where 0 represents the deck, treated as a player with no actions. We assume the players are truthful, rational and have perfect perception.
- The initial hand size h.

We also define the set of all possible multiplicities of a rank for the player i as $Rr_i = \{Rr_i^k \mid k \in M\}$.

First, we ensure that each player has exactly one multiplicity per rank.

$$c_1 = \bigwedge_{r \in N} \bigwedge_{i \in P} \left(\bigvee Rr_i \wedge \bigwedge_{\substack{Rr_i^k, Rr_i^l \in Rr_i \\ k \neq l}} \neg (Rr_i^k \wedge Rr_i^l) \right)$$

Secondly, the total multiplicity of each rank across all players' hands must match the suit size m.

$$c_2 = \bigwedge_{r \in N} \bigwedge_{\substack{(Rr_0^{k_0}, \ldots, Rr_p^{k_p}) \in \prod_{i \in P} Rr_i \\ m \neq \sum_{i \in P} k_i}} \neg \bigwedge_{i \in P} Rr_i^{k_i}$$

With these two constraints as common knowledge, a player can deduce the location of remaining cards based on their knowledge about the deck and their opponent's hand. We must also ensure that players cannot observe the multiplicity of the other players' ranks or the deck. This is captured by the indistinguishability relation, where for both players $j \in \{1, 2\}$, $\sim_j$ is the smallest equivalence relation on S such that:

$$Rr_i^k \sim_j Rr_i^l$$
$$\text{for every } i \in P, r \in N, k, l \in M, i \neq j$$

We can now define the initial game model $M_0 = (S, \sim, \mu, V)$. It consists of a single world, where all cards are in the deck, and both players know they are in this world:

$$S = \{w_0\}$$
$$\mu_i^{w_0}(w_0) = 1 \text{ for } i \in \{1, 2\}$$
$$V(p) \begin{cases} \{w_0\} & \text{if } p \in \{Rr_0^m \mid r \in N\} \\ \emptyset & \text{otherwise} \end{cases}$$

To update this model and allow players to take action, we need to define an event model that specifies how the existing model changes in response to each action.

Event Model

We divide the game's actions into four events:

$$E = \{\mathbf{ask}_{r,t,i,j}, \mathbf{give}_{r,x,y,i,j}, \mathbf{gofish}_{r,i,j}, \mathbf{draw}_{r,x',y',t',i,j}\}$$
$$\text{where } i, j \in \{1, 2\},\ i \neq j,$$
$$r \in N,\ x, y \in \{1, \ldots, m-1\},$$
$$x' \in \{0, \ldots, m-1\},\ y' \in \{1, \ldots, m\},$$
$$t \in \{1, \ldots, n\},\ t' \in \{1, \ldots, n \cdot m\}$$

- $\mathbf{ask}_{r,t,i,j}$ - player i, holding t cards of different ranks, asks player j if they have any cards of rank r.
- $\mathbf{give}_{r,x,y,i,j}$ - player i gives x cards of rank r to player j, who requested cards of rank r and already holds y of them.
- $\mathbf{gofish}_{r,i,j}$ - player i tells player j, who requested cards of rank r, to "Go Fish!". This means player i does not have the cards player j requested, and they must draw instead.
- $\mathbf{draw}_{r,x,y,t,i,j}$ - player i, holding x cards of rank r, was told to go fish by player j. Player i then draws a card of rank r from the remaining t cards, which contain y cards of rank r. This event is also used for dealing cards at the start of the game and when a player runs out of cards.

We can now define the non-trivial cases of the indistinguishability relation for both players $j \in \{1, 2\}$, $\sim_j$ which is the smallest equivalence relation on E where:

(1) $\forall i \neq j, r, r', x, x', y, y', t.$
$\mathbf{draw}_{r,x,y,t,i,j} \sim_j \mathbf{draw}_{r',x',y',t,i,j} \wedge$

(2) $\forall i \neq j, r, x, y, y'.t.$
$\mathbf{draw}_{r,x,y,t,i,j} \sim_i \mathbf{draw}_{r,x,y',t,i,j} \wedge$

(3) $\forall i \neq j, r, t, t'.$
$\mathbf{ask}_{r,t,i,j} \sim_j \mathbf{ask}_{r,t',i,j} \wedge$

(4) $\forall i \neq j, r, x, y, y'.$
$\mathbf{give}_{r,x,y,i,j} \sim_i \mathbf{give}_{r,x,y',i,j}$

(1) Expressing that player j can not tell what player i drew. And that when player i draws (2) they do not know how many cards of that rank there were in the deck. Furthermore when player i asks (3) player j, they cannot tell how many unique ranks player i has on their hand. Lastly, when player i gives (4) cards of rank r to player j, player i is unsure how many cards of this rank player j already had in their hand. However, it can still be inferred that player j had at least one card of this rank.

In addition to the previously introduced atomic proposition for representing cards, we will need three additional ones, to ensure the players adhere to the rules. $TURN_i$ indicates whether it is currently player i's turn, while $ASK_{r,i,j}$ denotes that player i has asked player j if they have any cards of rank r. Additionally, $GOFISH$ indicates that the player who has their turn must draw a card.

$$At = \{Rr_i^k \mid r \in N, i \in P, k \in M\} \cup \{TURN_i, ASK_{r,i,j}, GOFISH\}$$
$$\text{where } i, j \in \{1, 2\}\ \ i \neq j, r \in N$$

Now, we can define the main parts of the event model $A = (E, \sim, \text{PRE}, \text{pre}, \text{sub})$ per definition 3. We start by defining the preconditions of each event.

$$\text{PRE}(e) = \begin{cases} \{\phi^{ASK}_{r,t,i,j}\} & \text{if } e = \textbf{ask}_{r,t,i,j} \\ \{\phi^{GIVE}_{r,x,y,i,j}\} & \text{if } e = \textbf{give}_{r,x,y,i,j} \\ \{\phi^{GOFISH}_{r,i,j}\} & \text{if } e = \textbf{gofish}_{r,i,j} \\ \{\phi^{FISH}_{r,x,y,t,i,j}, \\ \quad \phi^{DEAL}_{r,x,y,t,i,j}, & \text{if } e = \textbf{draw}_{r,x,y,t,i,j} \\ \quad \phi^{EMPTY}_{r,x,y,t,i,j}\} \end{cases}$$

First, we introduce a formula that captures the recurring components of each precondition. It ensures that all events adhere to the card constraints and turn order, including the restriction that a player cannot request multiple ranks in a single turn.

$$\begin{aligned} \chi_{i,j,rs} = c_1 \wedge c_2 \wedge (TURN_i \wedge \neg TURN_j) \wedge \\ \bigwedge_{r \in rs} \neg ASK_{r,j,i} \wedge \\ \bigwedge_{r \in N \setminus rs} \neg(ASK_{r,i,j} \vee ASK_{r,j,i}) \end{aligned}$$

Players can ask for a rank at the start of their turn. In this case, they will ask for a random rank in their hand. For simplicity, we assume that players keep their books in their hands. Therefore, we also need to ensure that they do not request a card they already have four copies of.

$$\begin{aligned} \phi^{ASK}_{r,t,i,j} = \neg Rr^0_i \wedge \neg Rr^m_i \wedge \phi^{UNIQUE}_{i,t} \wedge \neg \phi^{DEAL} \\ \wedge \chi_{i,j,\emptyset} \wedge \neg GOFISH \end{aligned}$$

If the opponent has cards of the requested rank, they must give them to the player who asked. Otherwise, they will tell them to Go Fish.

$$\begin{aligned} \phi^{GIVE}_{r,x,y,i,j} &= ASK_{r,j,i} \wedge Rr^x_i \wedge Rr^y_j \wedge \chi_{j,i,\{r\}} \\ &\quad \wedge \neg GOFISH \\ \phi^{GOFISH}_{r,i,j} &= ASK_{r,j,i} \wedge Rr^0_i \wedge \chi_{j,i,\{r\}} \wedge \\ &\quad \neg GOFISH \end{aligned}$$

There are several scenarios in which a player must draw a card. The first is when the opponent says Go Fish. The second occurs at the beginning of the game when each player is dealt their hand in alternating order. The third happens when a player runs out of cards, and must draw a card from the pile at the start of their turn.

$$\begin{aligned} \phi^{DRAW}_{r,x,y,t,i,j} &= Rr^x_i \wedge Rr^y_0 \wedge \phi^{TOTAL}_t \wedge \chi_{i,j,\emptyset} \\ \phi^{FISH}_{r,x,y,t,i,j} &= \phi^{DRAW}_{r,x,y,t,i,j} \wedge GOFISH \\ \phi^{DEAL}_{r,x,y,t,i,j} &= \phi^{DRAW}_{r,x,y,t,i,j} \wedge \neg GOFISH \wedge \phi^{DEAL} \\ \phi^{EMPTY}_{r,x,y,t,i,j} &= \phi^{DRAW}_{r,x,y,t,i,j} \wedge \neg GOFISH \wedge \\ &\quad \neg \phi^{DEAL} \wedge \bigwedge_{r \in N} Rr^0_i \end{aligned}$$

In the preconditions, we have used the following definitions for simplification. ϕ^{DEAL} ensures that each player draws cards until they each have h cards at the start of the game. ϕ^{TOTAL}_t ensures there are exactly t cards remaining in the deck. $\phi^{UNIQUE}_{i,t}$ ensures player i has t unique ranks in their hand.

$$\phi^{DEAL} = \bigwedge_{\substack{(Rr^{k_0}_0, \ldots, Rr^{k_r}_0) \in \prod_{r \in N} Rr_0 \\ \sum_{r \in N} k_r \leq n \cdot m - 2h}} \neg \bigwedge_{r \in N} Rr^{k_r}_0$$

$$\phi^{TOTAL}_t = \bigwedge_{\substack{(Rr^{k_0}_0, \ldots, Rr^{k_r}_0) \in \prod_{r \in N} Rr_0 \\ t \neq \sum_{r \in N} k_r}} \neg \bigwedge_{r \in N} Rr^{k_r}_0$$

$$\phi^{UNIQUE}_{i,t} = \bigwedge_{\substack{(Rr^{k_0}_i, \ldots, Rr^{k_r}_i) \in \prod_{r \in N} Rr_i \\ t \neq \sum_{r \in N} 1 \\ 0 < k_r < m}} \neg \bigwedge_{r \in N} Rr^{k_r}_i$$

The latter two are useful for distributing subjective occurrence probabilities across each event, which we will now define with pre. When a player asks, they randomly select a rank from their hand. Since t represents the number of unique ranks in their hand, the probability of selecting a specific rank is $\frac{1}{t}$. Similarly, the probability of drawing a card of rank r depends on how many cards of that rank, y, remain in the deck and the total number of cards, t. Thus, the probability is $\frac{y}{t}$. Meanwhile, **give** and **gofish** must occur in a specific way in response to the question asked, as players are assumed to be truthful.

$$\text{pre}(e \mid \varphi) = \begin{cases} & \text{if } \exists r, t, i \neq j: \\ 1/t & \quad e = \textbf{ask}_{r,t,i,j} \ \& \\ & \quad \varphi = \phi^{ASK}_{r,t,i,j} \\ & \text{if } \exists r, x, y, i \neq j: \\ 1 & \quad e = \textbf{give}_{r,x,y,i,j} \ \& \\ & \quad \varphi = \phi^{GIVE}_{r,x,y,i,j} \\ & \text{if } \exists r, i \neq j: \\ 1 & \quad e = \textbf{gofish}_{r,i,j} \ \& \\ & \quad \varphi = \phi^{GOFISH}_{r,i,j} \\ & \text{if } \exists r, x, y, t, i \neq j: \\ & \quad e = \textbf{draw}_{r,x,y,t,i,j} \ \& \\ y/t & \quad \varphi \in \{\phi^{FISH}_{r,x,y,t,i,j}, \phi^{DEAL}_{r,x,y,t,i,j}, \\ & \qquad \phi^{EMPTY}_{r,x,y,t,i,j}\} \\ 0 & \text{otherwise} \end{cases}$$

We can now define sub, which describes how the truth assignment of each atomic proposition changes based on each event. For most properties, it acts as the identity function. The preconditions of each event lead to a similar outcome determined by the variables assigned to the event. As a result, we always assign an atomic proposition as true or false in all worlds. For example, in a $\mathbf{give}_{r,x,y,i,j}$ event, this ensures that when player i gives cards to player j, player i no longer holds any cards of that rank, and player j gained the same number of cards that player i lost.

$$\mathrm{sub}(e,p) = \begin{cases} \top & \text{if } e = \mathbf{ask}_{r,t,i,j} \ \&\ p = ASK_{r,i,j} \\ \bot & \text{if } e = \mathbf{give}_{r,x,y,i,j} \ \& \\ & \quad p \in \{Rr_i^x, Rr_j^y, ASK_{r,j,i}\} \\ \top & \text{if } e = \mathbf{give}_{r,x,y,i,j} \ \& \\ & \quad p \in \{Rr_i^0, Rr_j^{y+x}\} \\ \bot & \text{if } e = \mathbf{gofish}_{r,i,j} \ \& \\ & \quad p = ASK_{r,j,i} \\ \top & \text{if } e = \mathbf{gofish}_{r,i,j} \ \& \\ & \quad p = GOFISH \\ \bot & \text{if } e = \mathbf{draw}_{r,x,t,y,i,j} \ \& \\ & \quad p \in \{Rr_i^x, Rr_0^y, \\ & \qquad TURN_i, GOFISH\} \\ \top & \text{if } e = \mathbf{draw}_{r,x,t,y,i,j} \ \& \\ & \quad p \in \{Rr_i^{x+1}, Rr_0^{y-1}, \\ & \qquad TURN_j\} \\ p & \text{otherwise} \end{cases}$$

For simplicity, we chose to let players ask for a random rank in their hand. However, by adding the following, we could have narrowed down the player's choice by requiring them to select a rank they believe the other player is most likely to have[1].

$$\bigwedge_{q \in N \setminus \{r\}} P_i(\neg(Rr_j^0 \vee Rr_j^m)) \geq P_i(\neg(Rq_j^0 \vee Rq_j^m))$$

Initially, all cards are in the deck and it is player 1's turn as encoded in the following formula:

$$\chi = \chi_{1,2,\emptyset} \wedge \neg GOFISH \wedge \bigwedge_{r \in N} \bigwedge_{k \in M \setminus \{m\}} \neg Rr_0^k$$

These assumptions must all be common knowledge in the initial Bayesian model. As mentioned earlier, the draw event introduces significant uncertainty, as a player does not know which card their opponent drew from the pile. This raises the question: if we use a standard deck, with 13 ranks, 4 suits, and an initial hand size of 7, how many possible worlds would we have after the cards are drawn? This can be computed using the multiset coefficient combined with the inclusion-exclusion principle to account for the finite multiplicity of 4 for each rank. Fortunately, the multiplicities are evenly distributed, and we quickly encounter a case where the reserved elements exceed the cardinality of the hand. To account for the worst-case scenario, we remove the 7 cards in player 1's hand from different ranks before calculating the number of ways the second hand could be drawn. Finally, we avoid double-counting worlds where both players have the same hand.

$$\begin{aligned} \text{hand}_1 &= \left(\!\!\binom{13}{7}\!\!\right) - 13\left(\!\!\binom{13}{7-5}\!\!\right) \\ \text{hand}_2 &= \left(\!\!\binom{13}{7}\!\!\right) - 6\left(\!\!\binom{13}{7-5}\!\!\right) - 7\left(\!\!\binom{13}{7-4}\!\!\right) \\ |S| &= \text{hand}_1 \cdot \text{hand}_2 - \left\lfloor \frac{\text{hand}_1}{2} \right\rfloor = 2.295.733.083 \end{aligned}$$

This is a lot of possible worlds, and it could increase further depending on how the game plays out. Fortunately, the other events help narrow down this uncertainty, as seen in the following example.

Example

In the following example, we simplify our models by using $n = 3$ ranks, $m = 2$ suit size, and an initial hand size of $h = 2$. The first row of each world represents player 1's hand followed by the probability it assigns to that world. The same applies to the second row for player 2. For brevity, we focus on each player's hand and omit the remaining cards in the deck. We do not show each event that has occurred or the atomic propositions used to enforce the game rules. We also do not explicitly show reflexive or transitive arrows, though they are always assumed to be present. In each figure, the actual world is enclosed in a box. Initially, there are 21 possible worlds:

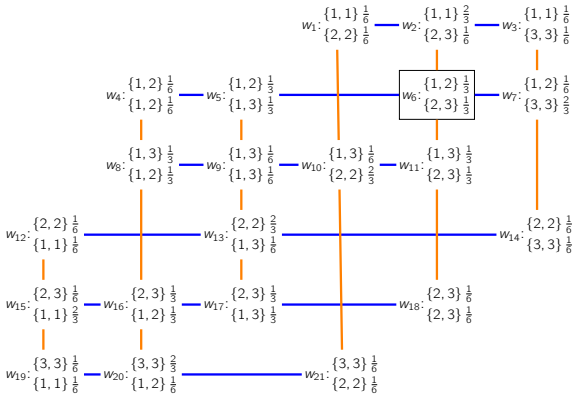

Figure 1: The world model after each player has drawn 2 cards.

[1] We would also have to record how many ranks share the same optimal probability to distribute the subjective occurrence probability properly.

Here, the Bayesian Kripke model captures an additional dimension in the decision-making, as players also take into account the likelihood of a particular hand being drawn from the deck. Notice that in each row, the probabilities assigned by player 1 to the different worlds sum to 1 since one of the worlds it cannot distinguish between must be the actual one. The same applies to each column for player 2.

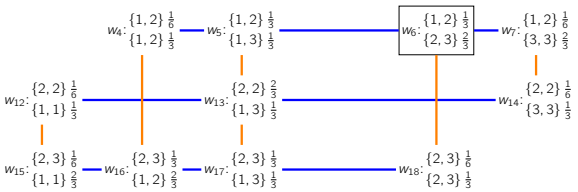

Figure 2: The world model after player 1 asks for cards of rank 2. Eliminating all worlds where player 1, did not have a card with rank 2. $ASK_{2,1,2}$ holds in all worlds.

When updating the model (see definition 4), we first eliminate the worlds that do not satisfy the precondition. In S' we have narrowed down the worlds which satisfy the precondition of the ask events.

$$S' = \{(w5, \mathbf{ask}_{2,2,1,2}), (w13, \mathbf{ask}_{2,1,1,2}), (w17, \mathbf{ask}_{2,2,1,2})\}$$

To show how probabilities are updated we will use w_{13} as an example. To make it clear what worlds we are writing about we will use them as arguments to the pre-function since they satisfy the corresponding precondition. Here we see w_{13} has a greater weight because the hand consists of two identical cards.

$$\mu_2(w_{13}, \mathbf{ask}_{2,1,1,2}) = \frac{\mu_2(w_{13}) \cdot \text{pre}_2(\mathbf{ask}_{2,1,1,2} \mid w_{13})}{\sum_{(s',e') \in S'} \mu_2(s') \cdot \text{pre}_2(e' \mid s')} = \frac{\frac{1}{6} \cdot 1}{\frac{1}{3} \cdot \frac{1}{2} + \frac{1}{6} \cdot 1 + \frac{1}{3} \cdot \frac{1}{2}} = \frac{1}{3}$$

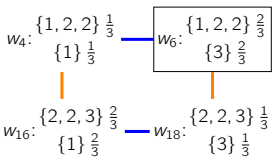

Figure 3: The world model after player 2 gives their cards of rank 2. Eliminating all worlds where player 2 did not have a card of rank 2 to give.

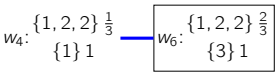

Figure 4: The world model after player 1 asks for cards of rank 1. Eliminating all worlds where player 1, did not have a card with rank 1. $ASK_{1,1,2}$ holds in all worlds

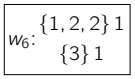

Figure 5: The world model after player 2 tells player 1 to go fish. Eliminating all worlds where player 2, has a card of rank 1. $GOFISH$ holds in all worlds

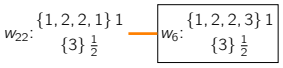

Figure 6: The world model after player 1 draws a card. Introducing a world to represent the possible cards player 1 could have drawn from the pile. This reintroduces uncertainty for the opponent

4 Implementation

Simulation of GoFish[2]

We previously calculated that there are 2.295.733.083 possible worlds after each player draws 7 cards. Therefore, if we want to run simulations we have to compromise otherwise the simulation will run out of memory. Our compromise is only representing one level of knowledge. This means we can not model statements such as player 1 knows player 2 knows that player 1 has 2 cards with rank 8 ($K_1K_2R8_1^2$). However, we are able to model statements such as player 1 knows player 2 has 2 cards with rank 8 $K_1R8_2^2$, and more importantly assign probabilities to worlds where that statement is true. This is critical because knowledge about the opponent having a certain amount of ranks is almost certainly lost, when the opponent draws a card, since it might introduce a possible world where they have 3 cards with rank 8 ($K_1(R8_2^2 \vee R8_2^3)$). When we only have one level of knowledge, the player's beliefs do not depend on each other, which means we can represent them independently of each other. This reduces the state space from $O(N \cdot M)$ to $O(N+M)$, where N and M are the combinations of cards each player believes their opponent might have.

However, the core part of the logic stays the same as we use Product Update (Definition 4) to calculate how states change probability every time a player learns something. We emulate that the players do not know which strategy their opponent uses, by assuming that they played a random legal move when calculating pre($\mathbf{ask}_{r,t,i,j} \mid s$), which is also the assumption we made when we modelled the program. For performance reasons, the simulation uses integers to model Rr_i^k and the other atomic propositions $TURN_i$, $GOFISH$, and $ASK_{r,i,j}$ are modelled implicitly through control flow.

[2] https://github.com/MarioRojoVicente/LogicalUncertaintyFinal

Deep Learning "playing" GoFish

With the goal of exploring the idea of an autonomous agent interacting in this environment a Q-Learning, value-based, model-free Reinforcement Learning(RL) algorithm, is implemented. This will try to learn a value, represented as a Q-value(Q(s, a)), which estimates the expected cumulative reward of taking an action *a* in a state *s*, and following the optimal policy thereafter. By only using the information of its own hand and the expected hand of the opponent, the deep learning agent is faced against other agents.

Winning as the only signal worth rewarding?

It has been hypothesized that setting a win(+1:win, -1:loose) as the only reward signal would give the highest degree of freedom for the model to learn heuristics of its own. The results showcase(Fig. 7) that the reward mechanism is too sparse. Given the size of possible state-action pairs that the agent can explore, it is infeasible to explore a representative enough sample of the state space to figure out optimal policies for every state. Thus, although well performing against "weak" heuristics such as "random" or "exploratory", the agent severely under-performs against sophisticated heuristics such as "Finish Fast" or "Hoarding".

Figure 7: Training outcomes of Q-learning based player against "FinishFast" heuristic.

A realistic avenue for improvement consists in introducing intermediate rewards, that would allow the agent to get feeback on its behaviour with higher frequency. Such devised reward system, consisting of (Win:+1.0, Loose:-1.0, Correct Prediction of opponent's card: +0.1, Incorrect Prediction of opponent's card:-0.05, Draw one card form the deck: -0.01, Complete a book: +0.5) yielded significantly better results(Fig.8, when trained against the same heuristic("FinishFast"):

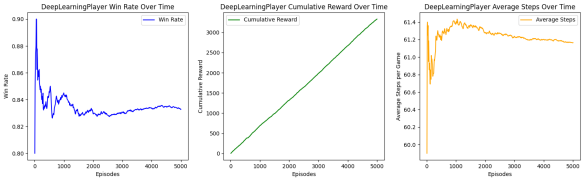

Figure 8: Training outcomes of Q-learning based player against "FinishFast" heuristic, with intermediate rewards.

5 Evaluation

To asses the significance of modeling epistemic knowledge in our particular scenario we implemented a set of heuristics. These can be subdivided into two groups, those limited to the use of propositional logic (GoForCloserPoint, referred to as GFCP in the following table) and those that include epistemic logic on their assessment of the best course of action. We also implemented a random choice heuristic that serves as a baseline for performance comparisons and included the deep learning model previously discussed.

These heuristics are all briefly explained:

- **Random**: asks for a random rank.
- **GoForCloserPoint**: asks for the rank the player is closer to completing.
- **Hoarding**: asks for the rank the enemy is more probable to have.
- **FinishFast**: asks for the rank of which is more probable the enemy player has all cards the player is missing.
- **Exploration**: asks for the rank we are less sure the enemy player has (reduces uncertainty).
- **ExplotationExploration**: mixes the policies of FinishFast and Exploration, but only exploits (acts as FinishFast) if the probability is above the established threshold.

Based on the outcomes of the benchmarking we can conclude that epistemic logic plays a significant role in strategies for card games such as Go Fish given the performance of the epistemic heuristics against GoForCloserPoint (the only heuristic together with Random that makes no use of epistemic logic). Another conclusion that can be drawn is that heuristics with a focus on exploitation prove better for the game of Go Fish given that our top performing heuristics (FinishFast and Hoarding) are purely exploitative.

Based on the Deep learning approach we can see that none of the classical heuristics are optimal, since the deep learning algorithm is able to learn an approach that beats each of them. Therefore if you can learn

the strategy of your opponent you can adapt and beat them as seen in table 1 in the Appendix.

6 Conclusion and Further Work

We have successfully implemented a complex logic scenario through the card game "GoFish". Although due to the infeasibility of representing second-order knowledge, first-order knowledge has been successfully handled by a set of agents which have interacted with the environment and each other, with different goals and strategies.

The performances of the agents, when paired against each other, reveal the strengths and weaknesses of their strategies. From the benchmarks we deduce that heuristics which make use of epistemic logic outperform those that only implement prepositional logic and that more aggressive heuristics perform generally better. Apart from that Hoarding seems to be the hardest strategy to adapt to, as expressed by it´s performance against Q-learning with a 25% win rate, probably because its strategy is the most complex, using probabilities to make informed decisions. Another interesting result is the difficulty(relatively to other heuristics) of deep learning to beat random. Perhaps this showcases, that it is more difficult to learn how to play against an agent that does not follow any pattern or one whose strategy is complex, and difficult to grasp

Nevertheless, the outcome of this work is considered successful. Through formalized propositional and epistemic logic, the game has been implemented; the agents developed and the interactions between them have been recorded. Therefore providing proof of the feasibility of this system to model complex environments.

On the other hand, limitations and avenues for improvement have been also uncovered:

- Even in a relatively simple and controlled scenario, such as this game, the order of magnitude of possible worlds that have to be considered constitutes a real challenge. On consumer hardware, a set of 5000 simulated games between two AI players, without considering higher-order knowledge than first, run for about 10 hours. This is a non-negligible aspect, especially when treating real-world complex scenarios.
- Deep learning approaches struggle with the extensive state space, and sparse reward mechanisms are unable to capture optimal strategies. A non-biased reward system for a reinforcement learning model such as the one implemented in this work, is yet to be devised.
- It would be interesting to extend our formalization and implementation to accommodate more than two players, as the game is often played with more players.

References

[1] Lasse Dissing Hansen and Thomas Bolander. Implementing theory of mind on a robot using dynamic epistemic logic. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 1615–1621. International Joint Conference on Artificial Intelligence Organization, 2020. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/224. URL https://ijcai20.org/. Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 ; Conference date: 07-01-2021 Through 15-01-2021.

[2] Bicycle Playing Cards. Go fish, 2024. URL https://bicyclecards.com/how-to-play/go-fish/. Accessed: 2024-12-03.

[3] Andreea Achimescu, Alexandru Baltag, and Joshua Sack. The probabilistic logic of communication and change. *Journal of Logic and Computation*, 29(7):1015–1040, January 2016. ISSN 0955-792X. doi: 10.1093/logcom/exv084. URL http://dx.doi.org/10.1093/logcom/exv084.

[4] Ronald Fagin and Joseph Y. Halpern. Reasoning about knowledge and probability. *Journal of the ACM*, 41(2):340–367, March 1994. ISSN 1557-735X. doi: 10.1145/174652.174658. URL http://dx.doi.org/10.1145/174652.174658.

[5] Jan van Eijck and François Schwarzentruber. Epistemic Probability Logic Simplified. In *Advances in Modal Logic*, Groningen, Netherlands, 2014. URL https://hal.science/hal-02534272.

[6] Johan van Benthem, Jan van Eijck, and Barteld Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006. ISSN 0890-5401. doi: https://doi.org/10.1016/j.ic.2006.04.006. URL https://www.sciencedirect.com/science/article/pii/S0890540106000812.

A Contributions Per Group Member

Mario Rojo Vicente: I was in charge of implementing the heuristics and the benchmarking code, as well as writing the Background and Evaluation sections of this paper.

Stine Lund Madsen: I primarily focused on modelling the game and discussing its base implementation. I was responsible for the Bayesian Kripke Model subsection and the calculation of possible worlds after cards had been drawn. Additionally, I contributed to the Event Model and Example subsections.

Frithjof Prochnow Sletten: I implemented the Python simulation, based on ideas found together with the group. I wrote the corresponding simulation section in the report. I also contributed to the Event Model and the Example subsections.

Artur Adam Habuda: Probability modeling, Deep Learning model design and implementation, introductory and closing statements.

B Full benchmarking results

	Random	GFGC	Hoarding	FinishFast	Exploration	ExtExr	Q-learning
Random	X	0.87	0.21	0.18	0.77	0.19	0.26
GFGC	0.13	X	0.07	0.04	0.22	0.05	0.10
Hoarding	0.79	0.93	X	0.44	0.96	0.52	0.25
FinishFast	0.82	0.96	0.56	X	0.97	0.61	0.16
Exploration	0.23	0.78	0.04	0.03	X	0.04	0.15
ExtExr	0.81	0.95	0.48	0.39	0.96	X	0.17
Q-learning	0.74	0.90	0.75	0.84	0.85	0.83	X

Table 1: This table captures the probability of an artificial intelligence based on the heuristic on every row beating one that follows the heuristics on every column, averaged over a 1000 games. Note that on the ExplotationExploration the threshold was hand-tuned to 0.2. Note that GFGC stands for GoForCloserPoint and ExtExr for ExplotationExploration. The results obtained by the deep learning player, are the outcome of a 500-round training process, against every opponent.

C The Probabilistic Logic of Communication and Change

The following definitions are all from the paper *The Probabilistic Logic of Communication and Change*[3].

Definition 1. (Bayesian Kripke models). Given sets Ag and At, a *Bayesian Kripke model* is a quadruple $M = (S, \sim, \mu, V)$ where:

- S is a non-empty set of states.
- $\sim$ is a family of equivalence relations $\sim_a$ on S, one for each agent $a \in Ag$.
- μ is a family of functions $\mu_a : S \to (S \to [0, 1])$, one for each agent $a \in Ag$, whose values are denoted by $\mu_a^s(s')$ and satisfy the conditions:
  - (SDP): if $s \sim_a t$ then $\mu_a^s(s') = \mu_a^t(s')$, for all $s' \in S$ (from [4]);
  - (CONS): $\mu_a^s(t) = 0$ if $s \nsim_a t$ (from [4]);
  - (CAUT): $s \nsim_a t$ if $\mu_a^s(t) = 0$ (from [5]);
  - (PROB): for every $s \in S$, $\sum_{t \in S} \mu_a^s(t) = 1$.
- $V : At \to \mathcal{P}(S)$ is a *valuation function*

Definition 2. (Substitutions [6]). A *substitution* is a function $\sigma : At \to \mathcal{L}_{\text{PLCC}}$ that maps all but a finite number of propositional atoms into themselves. Let $\text{dom}(\sigma) \stackrel{\text{def}}{=} \{p \in At \mid \sigma(p) \neq p\}$ be the the *domain* of σ. Let $sub_{\mathcal{L}_{\text{PLCC}}}$ denote the set of all such possible substitution functions and ϵ the identity substitution.

Definition 3. (Event Models). An *event model* over $\mathcal{L}_{\text{PLCC}}$ is the quintuple $A = (E, \sim, \text{PRE}, \text{pre}, \text{sub})$ where:

- E is a finite non-empty set of *events*.

- $\sim$ is a set of equivalence relations $\sim_a$ for each agent $a \in Ag$.
- PRE : $E \to \mathcal{P}(\mathcal{L}_{\text{PLCC}})$ is a map, such that $\Phi \stackrel{\text{def}}{=} \bigcup_{e \in E} \text{PRE}(e)$ is finite set of pairwise inconsistent formulas.
- pre is a family of functions $\text{pre}_a : \Phi \to (E \to [0,1])$ for each $a \in Ag$ assigning to each precondition $\phi \in \Phi$ a *subjective occurrence probability* distribution over E (i.e. $\sum_{e \in E} \text{pre}_a(\phi)(e) = 1$), such that $\text{pre}_a(\phi)(e) = 0$ iff $\phi \notin \text{PRE}(e)$.
- sub : $E \to sub_{\mathcal{L}_{\text{PLCC}}}$ assigns a substitution function to each event in E.

Definition 4. (Product Update). The update product of a static Bayesian Kripke model $M = (S, \sim, \mu, V)$ with an event model $A = (E, \sim, \text{PRE}, \text{pre}, \text{sub})$ is the weighted epistemic model $M \otimes A = (S \otimes E, \sim, \mu, V)$ where:

- $S \otimes E \stackrel{\text{def}}{=} \{(s,e) \mid s \in S, e \in E, (M,s) \vDash \bigvee \text{PRE}(e)\}$.
- $(s,e) \sim_a (s',e')$ iff $s \sim_a s'$ and $e \sim_a e'$.
- Ley $D \stackrel{\text{def}}{=} \sum_{(s',e') \sim_a (w,g)} (\mu_a^w(s') \cdot \text{pre}_a(e' \mid s'))$, and put:

$$\mu_a^{(w,g)}(s,e) \stackrel{\text{def}}{=} \begin{cases} \frac{\mu_a^w(s) \cdot \text{pre}_a(e|s)}{D} & \text{if } (s,e) \sim_a (w,g) \\ 0 & \text{otherwise} \end{cases}$$

  (Note that $D \neq 0$ for $(w,g) \in S \otimes E$)
- $V(p) = \{(s,e) \mid M, s \vDash \text{sub}(e)(p)\}$

Definition 5. (Semantics of PLCC). The semantics for $\mathcal{L}_{PLCC}$ is given by a relation $\vDash$ between pointed models (M,s), with M = $(S, \sim, \mu, V)$ and $s \in S$, and formulas ϕ, such that

$M, s \vDash true$	iff always
$M, s \vDash p$	iff always
$M, s \vDash \neg\phi$	iff $M, s \nvDash \phi$
$M, s \vDash \phi \wedge \psi$	iff $M, s \vDash \phi$ and $M, s \vDash \psi$
$M, s \vDash [a]\phi$	iff for all $t \in S$: if $s \sim_a, t$ then $M, t \vDash \phi$
$M, s \vDash [e]\phi$	iff $M, s \vDash \bigvee PRE(e)$ then $M \times A, (s,e) \vDash \phi$ where e is an event in action model A
$M, s \vDash [\pi]\phi$	iff for all $t \in S : if s R_\pi t$ then $M, t \vDash \phi$
$M, s \vDash \sum_{j=1}^{n} \alpha \cdot P_a(\Phi_j)$	iff $\sum_{j=1}^{n} \alpha \cdot \mu_a^s(\phi_j) \geq \beta$

where $\mu_a^s(\phi_j)$ is an abbreviation for $\sum_{s' \in S, s' \vDash \phi_j} \mu_a^s(s')$, and R_π is a binary relation given by

$sR_a t$	iff $s \sim_a t$
$sR_{\pi_1 \cup \pi 2} t$	iff $sR_{\pi_1} \cup R_{\pi_2} t$
$sR_{\pi_1;\pi_2} t$	iff $sR_{\pi_1}; sR_{\pi_2} t$ There is w, such that$sR_{\pi_1} w$ and $wR_{\pi_1} t$
$sR_{\pi_1 *} t$	iff $s(R_\pi)^* t$(where $(R_\pi)^*$ is the reflexive transitive closure of R_π)
$sR_{\pi ?} t$	iff $s = t$ and $s \vDash \phi$

Where $\vDash \varphi$ if $M, s \vDash \varphi$ for every pointed Bayesian Kripke model M, s.

D Probabilities Calculation: a detailed explanation

Belief Updates

*Intialization: each player starts believing that the oponent has 0 cards of each rank in their hands. **Probability of an Opponent Having a Rank**

- **Purpose:** Estimate the likelihood that the opponent has at least one card of a specific rank.
- **Computation:**

$$P(\text{opponent has rank } r) = \sum_{w \in W} p(w) \cdot \mathbb{1}(\text{rank } r \text{ count } > 0 \text{ in } w)$$

  where:

  - W: the set of all possible worlds consistent with the current beliefs.
  - $p(w)$: the probability of world w.
  - $\mathbb{1}$: an indicator function (1 if the opponent has the rank r, 0 otherwise).

Probability of an Opponent Having Exactly n Cards of a Rank

- **Purpose:** Determine the likelihood of the opponent holding exactly n cards of a specific rank.
- **Computation:**

$$P(\text{opponent has exactly } n \text{ cards of rank } r) = \sum_{w \in W} p(w) \cdot \mathbb{1}(\text{rank } r \text{ count } = n \text{ in } w)$$

Updating Probabilities After an Action

- When an action (e.g., asking for a card or drawing from the deck) is taken, some possible worlds become inconsistent with the new information.
- **Steps:**

  1. **Remove Inconsistent Worlds:** Worlds where the updated information (e.g., the opponent does not have rank r after being asked for it) does not match are discarded.
  2. **Re-normalize Probabilities:** Adjust the probabilities of the remaining worlds to ensure they sum to 1:

$$p'(w) = \frac{p(w)}{\sum_{w' \in W'} p(w')}$$

     where W' is the set of remaining worlds.

Heuristics Using Probabilities

HoardingHeuristic

- **Goal:** Ask for the rank that the opponent is most likely to have.
- **Computation:**
  - Compute $P(\text{opponent has rank } r)$ for each rank r.
  - Select the rank r with the highest probability.

ExplorationHeuristic

- **Goal:** Reduce uncertainty by targeting the rank with the most uncertainty (probability closest to 0.5).
- **Computation:**
  - Compute $P(\text{opponent has rank } r)$ for each rank r.

- Select the rank r where $P(\text{opponent has rank } r)$ is closest to 0.5:

$$r = \arg\min_r |P(\text{opponent has rank } r) - 0.5|$$

FinishFastHeuristic

- **Goal:** Complete books by asking for ranks where the opponent is likely to have all the remaining cards needed.
- **Computation:**
  - For each rank r, compute the probability of the opponent having exactly n cards of r, where n is the number needed to complete the book.
  - Select the rank r with the highest probability.

ExploitationExplorationHeuristic

- **Goal:** Combine exploitation (choose high-probability actions) and exploration (reduce uncertainty).
- **Computation:**
  1. Define a threshold T (e.g., 0.7).
  2. For each rank r:
     - If $P(\text{opponent has rank } r) > T$, choose the rank (exploitation).
     - Otherwise, use exploration logic to target ranks with the most uncertainty.

Deck Probabilities

- **When Drawing a Card from the Deck:** Compute the probability of drawing a card of rank r based on the remaining cards in the deck:

$$P(\text{draw rank } r) = \frac{\text{remaining cards of rank } r}{\text{total cards remaining in the deck}}$$