Cooperation and Competition: Modeling Intention and Behavior in Dual-Agent Interactions

by

Lily Zhang

S.B., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.
Cooperation and Competition: Modeling Intention and Behavior in Dual-Agent Interactions

by

Lily Zhang

Submitted to the Department of Electrical Engineering and Computer Science on May 25, 2018, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

A major goal of artificial intelligence research today is to build something that can cooperate with humans in an intelligent manner. In order to do so, we first must understand the mental mechanisms human use when solving problems of cooperation in dual-agent interactions, or between two people. We used reinforcement learning and Bayesian modeling to create a mathematical representation of this mental model. Our model is comprised of a high-level planner that understands abstract social intentions, and it employs two low-level planners that perform cooperative and competitive planning. To validate the model, we ran two experiments via Amazon Mechanical Turk to capture how humans attribute other players’ behaviors and how they themselves behave in problems of cooperation such as the prisoner’s dilemma. We compared our model against lesioned models and found that our model, which used both cooperative and competitive planning strategies, was the most representative of the data collected from both experiments.

Thesis Supervisor: Joshua B. Tenenbaum
Title: Professor

Thesis Supervisor: Max Kleiman-Weiner
Title: Ph.D.
Acknowledgments

Thank you to Dr. Max Kleiman-Weiner and Professor Josh Tenenbaum for your supervision and support throughout this project. I have learned so much over the past year, and I’ve had a great time working on this project.

Thank you to my mom, dad, and brother for always being there for me to lean on not just this year, but throughout my time at MIT. I couldn’t have done this without you guys!

Thank you to all my friends who have kept me going, especially during these last few weeks of thesis-writing. Every single hug, snack, and word of encouragement has meant the world to me.
Contents

1 Background ................................................. 17
   1.1 Introduction .............................................. 17
   1.2 Related Work ............................................. 18
      1.2.1 Social Dilemma ...................................... 18
      1.2.2 Iterated Prisoner’s Dilemma ......................... 19
      1.2.3 Stochastic Grid Games ......................... 21
      1.2.4 Social Goal Inference and Attribution .............. 21

2 Stochastic Grid Games .................................... 23
   2.1 Game Rules .............................................. 23
   2.2 Types of Games ......................................... 24
      2.2.1 Prisoner’s Dilemma Game ......................... 24
      2.2.2 Four-Way Game .................................... 25
      2.2.3 Double Fence Game ................................. 28
      2.2.4 Prisoner’s Dilemma with Fences ................... 28
   2.3 Computational Framework ............................ 29
      2.3.1 Low-level models ................................. 29
      2.3.2 High-level models .............................. 31
      2.3.3 Software Architecture .......................... 35

3 Attribution Experiment .................................. 37
   3.1 Experimental Design ................................. 37
   3.2 Software Architecture .............................. 39
List of Figures

1-1 A prisoner’s dilemma payoff matrix. The Nash equilibrium strategy is to betray because the expected value of staying silent is $0.5(-1) + 0.5(-3) = -2$, whereas the expected value of betraying is $0.5(0) + 0.5(-2) = -1$.

2-1 A classic prisoner’s dilemma game. The payoff matrix based on the game rules in Section 2.1 shows the expected reward of each player depending on if the players are cooperative or competitive. If both players were to go for the outer goals, they would each lose 3 points (1 point per turn) and earn 10 points, receiving $10 - 3 = 7$ points overall. If they were to both go for their closer goal, they would collide in the center square, and only one player would make it to their goal. Because each player has an equally likely chance of reaching their goal first, the expected reward is $0.5(10) + 0.5(0) = 5$, giving a net $5 - 3 = 2$ points (3 points were lost total, 1 for each of 2 turns and and 1 for the collision). Finally, if one player went for the inner goal while the other went for the outer goal, the competing player would receive $10 - 2 = 8$ points, while the cooperating player would receive $0 - 2 = -2$ points. In prisoner’s dilemma fashion, the expected value of competing ($0.5(2) + 0.5(-2) = 5$) is higher than that of cooperating ($0.5(7) + 0.5(-2) = 2.5$).
2-2 The four way game, which is another social dilemma game. If both players cooperate, they each take 5 turns and both receive the 10 point reward, ending with net 10 - 5 = 5 points each. If one player competes while the other cooperates, it takes the competing player only 4 turns to reach the 10 points reward, ending with net 10 - 4 = 6 points. However, the cooperating player does not reach their goal, so they end with 0 - 4 = -4 points. In the case that both players compete, each player has a 50% chance of reaching their goal in 4 turns (which incurs 5 points because of the extra point lost from colliding in the center square), ending with an expected point value of 0.5(10) + 0.5(0) - 5 = 0. Again, the expected value of competing (0.5(6) + 0.5(0) = 3) is higher than that of cooperating (0.5(5) + 0.5(-4) = 0.5).

2-3 A game using fences (denoted by the red dotted lines), which a player has a 75% chance of crossing successfully. Because players may not be able to make the moves that they intended to, the game may result in a different outcome from what the players were originally intending. The added probability complicates the expected payoff, so the calculation was omitted.

2-4 The PDF (prisoner’s dilemma with fences) game. It is the original prisoner’s dilemma game from Figure 2-1 but with fences added in front of the goals, combining social dilemma with testing for intention versus outcome. The added probability complicates the expected payoff, so the calculation was omitted.
The HMM graphical model representing the CCAgent’s action observation and belief update in dual-agent interactions. The top nodes represent player $p_1$, and the bottom nodes represent player $p_2$. On each step $i$ from 0 to $n$, each player $p$ has a belief $P(C_{p\_other,i})$ of how cooperative the other player is. Using this belief, they generate a policy $\pi_p$ from which they select their next action $a_{p,i}$. Then, each player observes the other player’s $a_i$ and updates their own belief state $C_{p\_other,i+1}$. The shaded nodes represent the observations and the dotted lines represent the belief updates.

Plots of each game sequence showing the human (solid) and model (dotted) values (using a single $\beta$ and $d$) per judgment point per player. (The yellow player is displayed in orange for better visibility.) Games 0-4 are prisoner’s dilemma games, games 5-9 are four-way games, games 10-14 are double fence games, and games 15-28 are PDF games.

A comparison between the optimized single decay and optimized asymmetric decays. The dotted line shows the transition from $C_0$ to $C_1$ using a single decay $d = 0.159$, and the solid line shows the transition using the asymmetric decays $d_a = 0.0867$ and $d_b = 0.307$. The $C_1$ for asymmetric decays is higher, showing that the judgments end up being more cooperative-leaning.

Plots of each game sequence showing the human (solid) and model (dotted) values (using a single $\beta$ and asymmetric $d$) per judgment point per player. The fit compared to Figure 3-1 is improved, indicating that these parameters are reflective of the human mental model.

A comparison of the human versus model values per judgment point using a single decay $d$ (left) and asymmetric $ds$ (right). The correlation using a single $d$ was $R = 0.899$, $R^2 = 0.809$, $RMSE = 0.107$. The correlation using asymmetric $ds$ was $R = 0.938$, $R^2 = 0.879$, $RMSE = 0.0748$. 
3-5 The fit between human and model values per judgment point for the lesioned models (Cooperate-Random on the left, and Random-Compete on the right). The values are quite scattered, with no strong positive correlation.

4-1 The relationship between the server and its clients, designed such that minimal information needs to be passed back and forth. When a player makes a move on step $i$, the client will send its current state and the selected action to the server. Once the server receives the message from both players, it will calculate the next joint-state and send it back to each client.

4-2 A plot of each model’s model likelihood as a stacked bar for each participant, sorted by how representative CCAgent is of that participant. This provides an overview of how the different models captured the human mental model at a granular, per player level. CCAgent is the dominant model, occupying the most area in this graph.

4-3 Plots of the log data likelihood $\ln(P(D|M))$ (left) and the model likelihood $P(M|D)$ (right) per model. It is evident that the CCAgent is the best fit to the experimental data, especially when comparing the model likelihoods.

4-4 The average likelihood of selecting the participant’s next action for each model, a metric of “correctness.” CCAgent is a hair higher than CompeteAgent, RandomCCAgent, and OutcomeAgent, but that can be attributed to the fact that at certain states in the game, there is only one possible, rational next movement, and that is captured by all four of these models.
4-5 A closer look at specific model comparisons. The top row shows the log data likelihoods for each model (the colors correspond to Figures 4-2 and 4-4), and the bottom row shows the model likelihoods. From left to right, the comparisons show CCAgent vs the low-level models, CCAgent vs RandomCCAgent, and CCAgent vs OutcomeAgent.

It is evident that CCAgent is the best fit to the experimental data, especially when comparing the model likelihoods.

5-1 A histogram showing the RMSE between each participant’s judgments and the sample mean judgments in the attribution experiment, or their "accuracy of judgment." The distribution is right-skewed, with the highest density between 0.2 and 0.3, showing that there exists a range of accuracy. This could potentially be used to determine if there’s a correlation between how accurately people make judgments and how they play the grid games.

A-1 This flowchart shows all 5 game sequences for the prisoner’s dilemma game. The sequences from top to bottom correspond to games 0, 3, 1, 4, and 2 from the experiment.

A-2 This flowchart shows all 5 game sequences for the four-way game. The sequences from top to bottom correspond to games 5, 7, 6, 8, and 9 from the experiment. Game 9 is notable because there is a high degree of miscoordination between the two players before they eventually reach the cooperative outcome.

A-3 This flowchart shows all 5 game sequences for the double fence game. They correspond to games 10, 11, 12, 13, and 14 from the experiment.
This flowchart shows the first 8 game sequences for the prisoner’s dilemma with fences game, corresponding to games 15-22 from the experiment. Each game sequence from A-1 now has 3 possible outcomes: both players reach the goals, only blue reaches the goal, and only yellow reaches the goal. This resulted in 14 total game sequences (one was removed due to symmetry).

This flowchart shows the last 6 game sequences for the prisoner’s dilemma with fences game, corresponding to games 23-28 from the experiment.
List of Tables

3.1 A comparison of the optimized values from fitting the CCAgent model with different set of free parameters. Using asymmetric decays had a marked improvement in the fit, but using asymmetric $\beta$s did not. . . 46

3.2 A comparison of the optimized values using the CCAgent, Cooperate-Random Agent, and Random-Compete Agent and how well each model fit to the experimental data. Clearly, the CCAgent outperformed both lesioned models. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

4.1 A table showing which free parameters each model depended on. . . 56

4.2 A comparison between using different sets of free parameters. As in Section 3.3.1, a single $\beta$ and asymmetric decays was found to improve the fit and also run within a reasonable amount of time. . . . . . . 56

4.3 The optimized parameters for each model. Both CoopAgent and Com- peteAgent have very low $\beta$s, which mean they are quite noisy models. In comparison, the high-level agents have higher $\beta$s, indicating they are able to fit to the two player data without needing to introduce much noise. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
Chapter 1

Background

1.1 Introduction

Cooperation is a tricky and ambiguous problem that humans face every single day, from determining if others are being cooperative, to deciding whether or not to behave cooperatively. Say you’re taking a class where only the top 10% of students get A’s. Do you cooperate with your classmates and study together in an effort to all get A’s? Do you instead compete and study alone to increase your own chances of getting the A? Your classmate asks you to take a break with them - is this a friendly gesture to help you destress? Or are they trying to get you to study less and fail? Sometimes people intend to cooperate, but due to some miscoordination, the outcome ends up going the other way. Let’s say your friend lent you their flashcards, but it ended up being the wrong material, causing you to do poorly in the class. Did your friend make a well-intentioned mistake? Or did they have some ulterior motive?

As you can see, cooperation can be hard to determine, especially when the data is sparse and noisy. However, humans as young as pre-verbal infants are able to do this seemingly common sense attribution on a daily basis [7]. If we want to build machines that can work with humans, we must first understand how cooperative decision making processes in the mind work, and how we as humans attribute and respond to cooperative and competitive behavior in dual-agent interactions between two people. By building a mathematical model of this type of human cooperation, we
can gain insight on the mechanisms at play when humans try to cooperate with each other, a vital facet of life encompassing everything from sharing on the playground to international diplomacy. From a computer science standpoint, we can eventually use these ideas to create artificial intelligence that can interact and cooperate with humans in a human-like and intelligent manner. This is especially important in this day and age given the rapid ubiquity of "smart" technology everywhere, from our phones, to our cars, to our homes.

This thesis consists of five total chapters. Chapter one provides an introduction to and related work about the problem at hand. Chapter two discusses the stochastic grid games and computational framework used to model this problem. Chapter three describes the first of two experiments, the attribution experiment. In this experiment, we explored how humans attribute cooperative and competitive behavior to players playing stochastic grid games. Chapter four discusses the second experiment, the two player experiment where human participants played the stochastic grid games against each other. This allowed us to evaluate how humans actually played these games and how to best model the way they played the game. Finally, chapter five provides a summary of the experiments and findings and also outlines future work to be done in this area.

1.2 Related Work

There has been much research surrounding evolutionary and game theoretic studies on cooperation and how humans behave in these situations.

1.2.1 Social Dilemma

A heavily studied cooperation problem is social dilemmas, or situations with a conflict between personal and collective interest. Maximizing one’s own personal interest is advantageous if everyone else continues to maximize the collective interest. However, if everyone chooses to maximize their own personal interest, everyone would be worse off than if everyone chose to maximize the collective interest. The dilemma presents
Figure 1-1: A prisoner’s dilemma payoff matrix. The Nash equilibrium strategy is to betray because the expected value of staying silent is $0.5(-1) + 0.5(-3) = -2$, whereas the expected value of betraying is $0.5(0) + 0.5(-2) = -1$.

itself when, in a single instance, maximizing one’s own personal interest is actually the dominant strategy. This is because the expected value from being selfish is higher than the expected value from maximizing the collective interest, despite the fact that total collective interest in the population is better than total personal interest.

Figure 1-1 shows an example of a prisoner’s dilemma problem [13], one of the most widely studied social dilemmas. In the prisoner’s dilemma, two prisoners’ decisions on how to act determine the lengths of their prison sentences, where "stays silent" is the collective interest and "betrays" is the personal interest. Assuming that the other person is equally likely to stay silent or betray, the expected value from betraying is higher (a lower prison sentence) than from staying silent. Thus, betrayal is the dominant strategy. So how is it that cooperation is able to manifest in a selfish population?

1.2.2 Iterated Prisoner’s Dilemma

An interesting case arises with iterated prisoner’s dilemma, which was studied extensively by Robert Axelrod in his book The Evolution of Cooperation [1]. Iterated prisoner’s dilemma is when prisoner’s dilemma is conducted for multiple rounds with memory of the previous rounds. Previously, in a single round, there was no penalty for mutual defection. However, with repeated rounds, the players now have an ongoing relationship that requires more strategic reasoning, and defection is not necessarily the dominant strategy anymore.

Axelrod hosted a tournament to study the best-performing strategy when playing iterated prisoner’s dilemma. There were four main distinguishing properties of

<table>
<thead>
<tr>
<th>A, B</th>
<th>B stays silent</th>
<th>B betrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>A stays silent</td>
<td>-1, -1</td>
<td>-3, 0</td>
</tr>
<tr>
<td>A betrays</td>
<td>0, -3</td>
<td>-2, -2</td>
</tr>
</tbody>
</table>


strategies that performed well. The most important property was being "nice," which meant not being the first to defect. The top 8 entries were all "nice" strategies, and the rest were all not "nice." However, it was also important not to be overly nice or overly mean - when the other player defects, it's important to retaliate, but when the other player returned to cooperation, it's also important to forgive. The final property was to not be envious, which meant not trying to score more points than the other player.

A strategy called tit for tat (TFT), submitted by Anatol Rapoport [14], won the tournament despite being the simplest algorithm. TFT started in the cooperative mode, and on subsequent rounds it switched to whichever mode the other player used on the previous round. TFT was also able to perform well with itself and generally avoid being exploited, indicating that this type of reciprocity is an important social norm.

Nowalk and Sigmund [12] introduced a new strategy called Pavlov, or "win-stay, lose-shift." If a player won a round, they would stay with the same strategy, and if they lost a round, they would shift to the opposite strategy. "Win-stay lose-shift" echoes a Pavlovian effect of responding positively to a reward and negatively otherwise, which is behavior found in many animals [5].

Pavlov is superior to TFT in its ability to account for noise and correct for mistakes. When two TFT players play against each other, if one makes a mistake and accidentally defects, the two will be stuck in a cycle of switching between defection and cooperation indefinitely. However, two Pavlov players will mutually defect for only one round before switching back to mutual cooperation. In a study of mutual-selection dynamics with error, Pavlov was selected as the dominant model if the benefit of cooperation exceeded a certain threshold, but TFT was never dominant [9]. An iteration of TFT called "generous tit for tat" can account for mistakes by employing a fixed probability of switching to the cooperative strategy, but it is still not as efficient as Pavlov [12]. Being able to account for noise is important as cooperation in the real world is quite susceptible to errors, and a high likelihood of miscoordination could greatly disrupt equilibrium strategies [2].
1.2.3 Stochastic Grid Games

De Cote and Littman studied problems of cooperation and dominant strategies by using stochastic two-player grid games. These games, which our research builds upon, are defined by \( \langle S, s_0, A_1, A_2, T, U_1, U_2 \rangle \). \( S \) is the set of all possible states, with a starting state of \( s_0 \) where \( s_0 \in S \). Each agent has a set of actions, \( A_1 \) and \( A_2 \) respectively, forming the joint action space \( A_1 \times A_2 \). For each joint state and joint action pair \( (s, a_1, a_2) \), the one step state-transition function is defined as \( T(s, a_1, a_2) = P(s'|s, a_1, a_2) \), which gives the probability of choosing a new state \( s' \in S \) given the current joint state and joint action. The transition functions of each joint state and joint action pair all together make up a policy \( \pi(a|s) \). Each agent tries to maximize its own utility \( U \), which is a function of \( s', s, a_1, a_2 \), and the policy that encodes the maximum expected utility is the optimal policy. Using these games, de Cote and Littman were able to build an algorithm that could find fair Nash equilibrium solutions in repeated rounds [3].

Expanding on de Cote and Littman’s work with stochastic grid games, Max Kleiman-Weiner et al. developed a hierarchical model of social planning to model how humans would behave in repeated rounds of different two-player games. The full model that coordinated low-level action plans and high-level strategic goals was the best at capturing the rates of cooperation from the behavioral experiment, indicating that it mostly closely reflected the hierarchical model of social planning that humans use to cooperate and compete [10].

These social dilemma games and strategies mirror real-life paradigms when it comes to cooperation and competition, and they provide a basis for how our model can infer these different behaviors and respond appropriately.

1.2.4 Social Goal Inference and Attribution

Though it’s important to study how people behave in problems of cooperation, it is equally important to understand is how people infer and attribute cooperative behavior, especially under suboptimal conditions. Remarkably enough, humans are
able to create rich inferences from stimuli as simple as shapes moving in a 2D plane [8].

One type of social goal attribution was formalized by Tomer Ullman et al. by developing a model that could infer if one agent was helping or hindering the other, with no knowledge of either agent’s goal. The model was also able to perform judgments from relatively few observations. The model was able to capture some of the complexity of the human mental model, providing a far better fit than a simple perceptual cue-based model [16].

People can also attribute levels of intelligence based off of viewing a series of actions. In a study by Marta Kryven et al., people attributed intelligence to one of two main types of planners: those who used some sort of a strategy, and those who reached a positive outcome [11]. This illuminated the idea that there can be multiple ways to attribute intelligence, whether that was based on if the agent behaved in a human-like way or was able to reach the goal in fewer steps.

This introduced another interesting dichotomy of being clever versus being lucky. Gerstenberg et al. explored how people assigned credit when something unexpected happened, whether it was due to that person’s skill and intellect or if it was a fluke [6]. He found that both the expectations and the context played a major role in how people placed responsibility upon others.

Though our study will focus specifically on attributing and inferring cooperative and competitive behavior, these different branches of attribution are important to keep in mind for exploration and possible future directions our model could expand.
Chapter 2

Stochastic Grid Games

To study this problem of modeling cooperative and competitive intention and behavior, we used two-player stochastic grid games as our environment based off of the same formalization as de Cote and Littman’s work [3] referenced in Section 1.2.3. We assumed that each player operated under the principle of rationality [4], which states that the agents will take the most efficient actions to reach their respective goals. These games include and build upon ones originally used in Kleiman-Weiner’s study on repeated stochastic games [10].

2.1 Game Rules

Each player is represented by either a blue circle or a yellow circle, and they have a goal of their own color worth 10 points that they are trying to reach. Each grid is configured such that there can be multiple ways for each player to reach their goal, and players can also have more than one goal of their own color. Players are also able to enter each other’s goals to no effect. An example of a game can be seen in Figure 2-1.

Players are allowed to move one square up, down, left, right, or stay in place. Players lose 1 point per turn they take, even if they just wait in place. The game updates simultaneously, which means that both players must choose their next move before the moves are made and the game state is updated.
The game ends when at least one player reaches their goal. For both players to receive the 10 point reward, they must enter their respective goals on the same turn, which is the cooperative outcome. Otherwise, only one player will receive the 10 point reward, which is the competitive outcome in favor of whoever reached the goal first. The game will always end in either a cooperative or competitive outcome.

There are also illegal movements. Because the two players have no way of communicating beyond what movements they are making, collisions are possible. All collisions cost 1 extra point for the player(s) that initiated the collision. There are three types of collisions. Only one player can occupy a square at a time. When two players try to move into the same square, a coin flip will determine which player is successful (which means both players have an equal chance of "winning" and successfully making it into the target square), and both players lose an extra point. If players try to swap spots, they will be bounced back to their previous square, and both players will lose an extra point. Finally, a player cannot enter a square that another player occupies. If a player tries to move into the square of a stationary player, that player will lose an extra point.

Another feature of the games is fences, which are denoted by red dotted lines, as seen in Figure 2-3. Fences are barriers in between spaces, and if a player tries to move across a fence, they only have a 75% chance of successfully doing so.

2.2 Types of Games

The following describes the types of games that were used in the experiments to test for cooperative and competitive intention and behavior.

2.2.1 Prisoner’s Dilemma Game

We can structure the grid game to be like a prisoner’s dilemma [3 [10, as seen in Figure 2-1. In this game, each player has two goals. Reaching the outer goals takes 3 steps, whereas reaching the inner goals takes 2 steps. If a player were playing alone, it makes sense to take the shortest path in order to maximize their score. However,
the introduction of another player makes things more complicated, which is reflected in the payoff matrix shown in Figure 2-1. Each player could behave cooperatively and try to reach the goals at the same time, or they could behave competitively and try to reach their own goal first. As seen, the expected value of competing and trying to maximize one’s own score is higher than that of cooperating.

It may seem like the only cooperative route is for both players to go for the outer goals, and the only competitive route is for both to go for the inner goals. However, there are more possibilities - for example, one player could first choose to go for the inner goal and then wait for the other player, which would also lead to a cooperative outcome despite one player ending in the inner goal. It is also possible to make mistakes on this grid. For example, two players intending to cooperate could both choose to move into the center square on their first turn and collide on accident. The player who moved into the intersection successfully could wait a turn after the intersection for the other player to move closer to their goal, ultimately resulting in a cooperative outcome. This type of interaction demonstrates that miscoordination can be an attribute of cooperation and does not necessarily detract from it.

2.2.2 Four-Way Game

Another example of a social dilemma is the four-way game, shown in Figure 2-2. In this game, each player’s goal is located opposite a four-way intersection. Because only one player can occupy a square at a time, the cooperative outcome is only possible if a player waits for the other player to catch up before proceeding to their own goal. Otherwise, whichever player occupies the intersection square first will reach the goal first, resulting in the competitive outcome. Because there is an equal chance of each player ending up in the intersection square first, there competitive outcome is equally likely to be in the favor of either player.

As seen in the matrix in Figure 2-2, the payoff structure for the four-way game follows that of a social dilemma as well (assuming no unintended collisions). As with the prisoner’s dilemma game, this game also allows for mistakes of coordination when deciding which player should move into the intersection square first.
Figure 2-1: A classic prisoner’s dilemma game. The payoff matrix based on the game rules in Section 2.1 shows the expected reward of each player depending on if the players are cooperative or competitive. If both players were to go for the outer goals, they would each lose 3 points (1 point per turn) and earn 10 points, receiving $10 - 3 = 7$ points overall. If they were to both go for their closer goal, they would collide in the center square, and only one player would make it to their goal. Because each player has an equally likely chance of reaching their goal first, the expected reward is $0.5(10) + 0.5(0) = 5$, giving a net $5 - 3 = 2$ points (3 points were lost total, 1 for each of 2 turns and and 1 for the collision). Finally, if one player went for the inner goal while the other went for the outer goal, the competing player would receive $10 - 2 = 8$ points, while the cooperating player would receive $0 - 2 = -2$ points. In prisoner’s dilemma fashion, the expected value of competing $(0.5(2) + 0.5(8) = 5)$ is higher than that of cooperating $(0.5(7) + 0.5(-2) = 2.5)$. 

<table>
<thead>
<tr>
<th></th>
<th>Yellow cooperates</th>
<th>Yellow competes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue cooperates</td>
<td>7* , 7*</td>
<td>−2, 8</td>
</tr>
<tr>
<td>Blue competes</td>
<td>8, −2</td>
<td>2, 2</td>
</tr>
</tbody>
</table>

*assuming no miscoordination
Figure 2-2: The four way game, which is another social dilemma game. If both players cooperate, they each take 5 turns and both receive the 10 point reward, ending with net 10 - 5 = 5 points each. If one player competes while the other cooperates, it takes the competing player only 4 turns to reach the 10 points reward, ending with net 10 - 4 = 6 points. However, the cooperating player does not reach their goal, so they end with 0 - 4 = -4 points. In the case that both players compete, each player has a 50% chance of reaching their goal in 4 turns (which incurs 5 points because of the extra point lost from colliding in the center square), ending with an expected point value of 0.5(10) + 0.5(0) - 5 = 0. Again, the expected value of competing (0.5(6) + 0.5(0) = 3) is higher than that of cooperating (0.5(5) + 0.5(-4) = 0.5).
Figure 2-3: A game using fences (denoted by the red dotted lines), which a player has a 75% chance of crossing successfully. Because players may not be able to make the moves that they intended to, the game may result in a different outcome from what the players were originally intending. The added probability complicates the expected payoff, so the calculation was omitted.

### 2.2.3 Double Fence Game

In the game shown in Figure 2-3, there are two sets of fences. The players start the same distance away from the first fence and both try to cross their respective fence. If only one player gets over the first set of fences, they could demonstrate cooperative intent by waiting for the other player to cross successfully before continuing towards their goal. In addition, there are a second set of fences in front of the goals. The two players could try to move into their own goals at the same time, but one player could cross the fence successfully while the other does not, resulting in what looks like a competitive outcome despite the intended cooperation. The fences could also hold up a competitive player, causing them to ultimately reach their goal at the same time as the other player, a cooperative outcome contrary to the competitive intent. By using these fences, we can demonstrate that understanding cooperative intent requires more than just examining the final outcome of the players - it’s necessary to infer the entire sequence of movements made by both players.

### 2.2.4 Prisoner’s Dilemma with Fences

To capture both concepts of social dilemma and intention versus outcome, we created the prisoner’s dilemma with fences game (PDF) shown in Figure 2-4. PDF is the
original prisoner’s dilemma game shown in Figure 2-1 but with a fence added in front of each goal. The original structure of the grid provides the social dilemma aspect, and because the fences can block movements, the final outcome may not reflect the original intention.

Figure 2-4: The PDF (prisoner’s dilemma with fences) game. It is the original prisoner’s dilemma game from Figure 2-1 but with fences added in front of the goals, combining social dilemma with testing for intention versus outcome. The added probability complicates the expected payoff, so the calculation was omitted.

2.3 Computational Framework

In our model, an extension of Kleiman-Weiner’s hierarchical planner [10], each player is represented by a high-level planning agent that can infer abstract social goals and employs two separate low-level types of policy generation, a cooperative planner and a competitive planner. We believe that this high-level planner composed of both cooperative and competitive elements is more representative of the mental model humans use in these problems of cooperation than a planner that uses a single policy.

2.3.1 Low-level models

The low level models generate policies $\pi(a|s)$ that define the likelihood of taking actions $a$ given state $s$. The policies differ based on the overall abstract goal - either cooperative or competitive.
2.3.1.1 CoopAgent

The cooperative planner, or CoopAgent, uses value iteration \[15\] over the joint action space (defined in Equations 2.1 and 2.2) to compute an optimal MDP policy and its value for the joint agent based on all the possible joint states and actions in a given game. The joint agent can be thought of as a puppetmaster that controls both players in the game, which means that the players will always behave cooperatively. The joint policy is then marginalized over each individual player in order to get each player’s individual cooperative policy. The parameter $\beta$ is used to compute the softmax that determines how noisy the transition probabilities are. In the extreme cases, if $\beta = \infty$, the likelihood $P(a_1, a_2|s)$ of the joint action $(a_1, a_2)$ with the highest likelihood will equal 1 and the other joint actions will have likelihoods of 0. If $\beta = 0$, all joint actions will have uniform likelihoods.

\[
P(a_1, a_2|s) = \pi^G(s, a_1, a_2) \propto e^{(\beta Q^G(s, a_1, a_2))} \quad (2.1)
\]

\[
Q^G(s, a_1, a_2) = \Sigma_{s'} P(s'|s, a_1, a_2)[U^G(s', s, a_1, a_2) + \max_{(a_1', a_2')} Q^G(s', a_1', a_2')] \quad (2.2)
\]

2.3.1.2 CompeteAgent

The competitive planner, or CompeteAgent, calculates its policy recursively for $k$ levels. It starts with one of five predefined level-0 models of the other player: 1) static, where the other player doesn’t move, 2) solo, where the other player is playing as if it’s alone; 3) friend, where the other player moves out of the way; 4) cooperative, where the other player behaves cooperatively, and 5) random, where the other player moves randomly. In our model, we use the level-0 solo model with $k = 1$. The $\beta$ parameter used in cooperative policy generation is also used here. The level-0 model
The CompeteAgent then calculates its policy by best-responding to the level-0 model and maximizing its own reward, and it can do so recursively for $k$ levels using the following equations:

$$P(a_i|s, k = 0) = \pi_0^k(s) \propto e^{(\beta Q_0^k(s, a_i)}$$  \hspace{1cm} (2.5)

$$Q_0^k(s, a_i) = \sum_{s'} P(s'|s, a_i)\left[U_i(s, s, a_i, s') + \max_{(a'_i)} Q_0^k(s', a'_i)\right]$$  \hspace{1cm} (2.6)

2.3.1.3 RandomAgent

To provide a point of comparison for the Coop- and CompeteAgents, we created a lesioned model, the RandomAgent which does no planning at all. There is a uniform distribution over all valid actions, which means $P(a = a_i|s) = \frac{1}{N}$ for all valid actions $a_i$, where $N$ is the total number of next actions.

2.3.2 High-level models

When playing the game, the high-level agent observes the other player’s action and infers abstract social goals of cooperation and competition. Based on its inference, it then uses the low-level planners to generate a policy to select the next action. This Hidden Markov Model (HMM) can be seen in Figure 2-5. We have three high-level models, the CCAgent, the RandomCCAgent, and the OutcomeAgent (the latter two are lesioned models used for comparison).

2.3.2.1 CCAgent

Our primary high-level agent is the Cooperate-Compete Agent, or CCAgent for short. We believe that the CCAgent is the most representative of the mental model humans
use when playing these grid games and making inferences about cooperation and competition.

The CCAgent starts with a prior of 0.5, which means that initially it has no knowledge of how cooperative or competitive the other player is, and both policies (cooperative and competitive) are weighted equally. On each step \( i \), the agent observes the other player’s action \( a_2 \). Then using the policies for both the cooperative and competitive planners, it determines the relative likelihood of the player being in that state given if it’s cooperative \( C \) or competitive \( C' \), denoted by \( P(s, a_1, a_2|C_i) \) and \( P(s, a_1, a_2|C'_i) \), respectively. It then calculates a new posterior probability \( P(C_{i+1}|s, a_1, a_2) \) and \( P(C'_{i+1}|s, a_1, a_2) \) of the player. The calculation is also parameterized by decay \( d \). The decay parameter "decays" the posterior probability to a less extreme value, pulling down higher values and raising up lower values. As decay approaches 0 from the left, the agent is more likely to remain in the same mode as its prior. As decay approaches 1 from the right, the agent is more likely to switch between low-level modes. This is formalized in the following equations:

\[
T = \begin{bmatrix}
1 - d & d \\
d & 1 - d
\end{bmatrix}
\begin{bmatrix}
P(C_i) \\
P(C'_i)
\end{bmatrix}
\]

\[
P(C_{i+1}|s, a_1, a_2) \propto P(s, a_1, a_2|C_i) \times T_1
\]

\[
P(C'_{i+1}|s, a_1, a_2) \propto P(s, a_1, a_2|C'_i) \times T_2
\]

The action observation and belief update can be represented by a Hidden Markov Model, as shown in Figure 2-5. Because the low-level planners provide a layer of abstraction, the high-level planner can use them without considering the details.

The posterior probability \( P(C_{i+1}) = P(C_{i+1}|s, a_1, a_2) \) is the model’s beliefs of the cooperativeness of the other player after observation and what the agent uses to select its next policy. To create a generous model that always has some probability of switching, we bound \( P(C_{i+1}) \) by \([b, 1 - b]\) where \( 0 \leq b \leq 0.5 \). It then uses \( P(C_{i+1}), b \), and the low-level policies \( \pi_{\text{cooperate}} \) and \( \pi_{\text{compete}} \) to generate a weighted policy \( \pi_{\text{weighted}} \).
Figure 2-5: The HMM graphical model representing the CCAgent’s action observation and belief update in dual-agent interactions. The top nodes represent player $p_1$, and the bottom nodes represent player $p_2$. On each step $i$ from 0 to $n$, each player $p$ has a belief $P(C_{p\_other,i})$ of how cooperative the other player is. Using this belief, they generate a policy $\pi_p$ from which they select their next action $a_{p,i}$. Then, each player observes the other player’s $a_i$ and updates their own belief state $C_{p\_other,i+1}$. The shaded nodes represent the observations and the dotted lines represent the belief updates.
where:

$$\pi_{\text{weighted}} = (P(C_i+1)((1 - 2b) + b)\pi_{\text{cooperate}} + (P(C'_i+1)((1 - 2b) + b)\pi_{\text{compete}} \quad (2.10)$$

This weighing of policies means that the CCAgent follows a generous tit for tat mentality [12] where it will more likely respond with whatever it observes the other agent did on the previous step, with some probability of switching to the opposite. Essentially, if the other agent is observed to be more cooperative, the agent will weigh the cooperative policy more heavily. Inversely, if the other agent is observed to be more competitive, the agent will weight the competitive policy more heavily, acting reciprocally.

2.3.2.2 RandomCCAgent

We built the RandomCCAgent in order to have a benchmark to demonstrate that observing and updating the belief state about the other agent is necessary. This agent never updates its belief about the other agent’s level of cooperation, which means that the cooperative and competitive policies are always equally weighted. In other words, for each step, the RandomCCAgent is equally likely to use the cooperative and competitive policy to generate its next action regardless of what the other agent is doing.

2.3.2.3 OutcomeAgent

Another important aspect of the CCAgent is that it makes observations after each action and is constantly updating its belief state of the other agent’s level of cooperation. To validate this aspect of the CCAgent, we built the OutcomeAgent, which only observes the final state of the previous round and then updates its strategy based on the win-stay lose-shift learning strategy described in Section 1.2.2. The OutcomeAgent starts believing the other agent is cooperative, with $P(C_0) = 1$. After each round, if an agent makes it into its own goal, it sees that it has "won" and continues using its current strategy. Otherwise, the agent sees that it has "lost" and
shifts to the opposite strategy. Despite being a lesioned model, we do expect the OutcomeAgent to capture some portion of the population as it employs a learning strategy that is found in nature [5].

2.3.3 Software Architecture

There were three main classes that were used to build out the computational framework, and everything was coded in Python. There was a Game object that encoded the properties and rules of the game, as described in Section 2.1. Each new game configuration from Section 2.2 was a subclass of the Game class. There was also an Agent object that represented each of the different low-level and high-level models in Sections 2.3.1 and 2.3.2, and these agents would be the "players" and "observers" of the game. These agents were built using the planning and policy generation code original written by Kleiman-Weiner [10]. Finally, there was a World class that encoded the interactions between the game and the agents.
Chapter 3

Attribution Experiment

The attribution experiment tested how humans assign cooperative and competitive intent to players playing the grid games, and it served to validate how accurately our model’s observed beliefs reflected those of humans. Two rounds of the attribution experiment were run on Amazon Mechanical Turk (MTurk) with different sets of games. There were 52 and 51 participants, respectively.

3.1 Experimental Design

After consenting to the terms and conditions of our experiment, the participant was shown a series of tutorial slides that instructed the user of the game rules described in Section 2.1. Each slide had a short animation demonstrating an aspect of the game, and participants were able to play the animations as many times as they wanted. See https://youtu.be/_1tS16GUXsY for a video of the attribution experiment tutorial.

After completing the tutorial, the participant began the experiment. In the experiment, the participant watched full game sequences between people playing different grid games (a game sequence here refers to an entire game from the starting state to the final state of one or both players reaching the goal(s)). At various points throughout each game sequence, the game paused, and the participant was asked to evaluate how cooperative each player’s behavior seemed on a sliding scale from "Not Cooperative" to "Cooperative." Participants were not able to evaluate the players
until the current step of the sequence finished playing. Participants were also able to
replay the entire game sequence up to the paused point if they needed to.

The first round of the experiment consisted of 15 total different game sequences
across 3 games, the prisoner’s dilemma game, the four-way game, and the double fence
game (shown in Figures 2-1, 2-2, and 2-3). For each game, there were 5 predefined
series of movements per game, which can be seen in Figures A-1, A-2, and A-3. Each
game sequence displayed one (or more) of cooperative intent, competitive intent, both
cooperative and competitive intent, cooperative but miscoordinated intent, and intent
that differed from the final outcome.

The second round of the experiment consisted of 14 total different game sequences,
all of which were the PDF game seen in Figure 2-4. Because there was a 25% chance
of a fence blocking a player’s movement, each prisoner’s dilemma game sequence from
the first round of the experiment now had 3 possible outcomes - the two players jointly
reaching the goals, only blue reaching the goal, or only yellow reaching the goal. This
resulted in 14 total game sequences for the PDF game (in the case where both players
move to the outward goals, blue reaching the goal first and yellow reaching the goal
first are symmetric, so only one was used), which can be seen in Figures A-4 and A-5.

Participants had to evaluate all the game sequences in the set, and the sequences
were shown in a random order so as to remove any possible bias or correlation between
separate game sequences. Participants were also told that each game sequence was
played by two entirely new people to remove any perceived continuation of the players’
behaviors across game sequences. Furthermore, the blue and yellow players’ behaviors
were not consistent between game sequences to prevent participants from associating
a certain color with a certain level of cooperativeness.

At the end of the experiment, participants were asked to fill out a demographic
survey indicating their gender, age, political affiliation, and their thought process
when evaluating the cooperativeness of each player.

To view a video of a participant completing the attribution experiment, please go
to https://youtu.be/hR-R1FdQ_m4 (first set of game sequences) or https://youtu.
be/07UsiZ6DvLY (second set of game sequences).
3.2 Software Architecture

Because these experiments needed to be run on MTurk, they were written in Javascript and HTML/CSS using the psiTurk framework. The basic structure consisted of a Game class that contained all the functionality that had to be uniform across all games, such as the game rules and the game UI, and a Player class that contained the state for each player. Separate subclasses were created to have games with specific functionality, such as being able to play back an entire game sequence and pause at certain points.

The participants' evaluations of each player per step per round were recorded, and after the participant completed the final demographic survey, all of this data was sent to the psiTurk server to be used for later analysis.

All of the data analysis was done in Python. NumPy, SciPy, and pandas were used to manipulate the data, and Matplotlib and seaborn were used for plotting and visualizing the data.

3.3 Results and Analysis

We collected 52 responses for the first set of game sequences and 51 responses for the second set of game sequences. All of the participants' evaluations per player per step per game sequence, called judgment points, were scaled to be between 0 and 1 and averaged to represent the sample mean human belief of the level of cooperativeness at that judgment point. The average standard error across all evaluations per judgment point was 0.0269, and the 95% confidence interval was $(\bar{x} - 0.0527, \bar{x} + 0.0527)$. This is quite narrow and means that the sample mean $\bar{x}$ should be fairly representative of the population mean.

The sample means were compared to the belief values generated by the CCAgent when observing the same judgment points. However, we noticed that the sample means per judgment point did not encompass the entire range from 0 to 1 - the minimum mean was found to be 0.186, and the maximum mean 0.877. The model
beliefs $P(C)$ were scaled to have the same max and min as the experimental values, by performing $P(C)(0.877 - 0.186) + 0.186$ for all values. These bounds could reflect the idea that humans are never fully certain how cooperative or competitive a player is acting. Additionally, because we are modeling the "average" player, the bounds allow for us to account for noise across people.

The experimental and model beliefs of cooperativeness were plotted against each other as line graphs to get a high level view of fit, with one line per player and one graph per game sequence. Examples can be seen in Figures 3-1. The experimental and model values per judgments point were also plotted against each other to visualize the level of correlation between the two sets of values, an example of which can be seen in Figure 3-4.

3.3.1 Parameter Optimization

As seen in Section 2.3.2.1, CC-Agent depends on two free parameters, beta $\beta$ and decay $d$. In order to find the set of parameters that would make our model best fit the human judgments, the model was optimized using SciPy’s optimize method to minimize the root-mean-square error (RMSE) between the experimental means and model beliefs per judgment point.

3.3.1.1 Single Beta and Decay

The optimized fit using $\beta = 4.04$ and $d = 0.159$ can be seen in Figures 3-1. The fit was generally decent, but there were some values that are repeatedly fit poorly. Take games 10-14, for example, which were double fence games. The blue player was always the one that got stuck at the fence, so its only movement was to continue forward toward the goal. As a result, the model had no indication of how cooperative blue was being, so its value hovered around 0.5 for the entire game sequence. However, the human values tended to be higher, hovering around 0.7 for the entire game sequence.

Another instance of this type of ill-fit is in games 23-28, which were PDF games. On the first step both players went for the center and collided, and the blue player won
the collision. At this point, both the human and model beliefs of cooperation were very low. On the next step, the yellow player moved toward its outer goal. Regardless of what the blue player did, this was viewed as a weak signal of cooperation by humans, and the belief always rose to about 0.6. On the contrary, the model still viewed this as competitive-leaning, and the value hovered around 0.4.

In both of these examples, the human and model beliefs followed the same trajectories, but the human values tended to be slightly higher. This indicated that, when there was uncertainty of whether a player was behaving cooperatively or competitively, humans generally tended to see them as more cooperative. Additionally, it seemed that the human mental model might have been transitioning from competition to cooperation at a separate, higher rate than it was transitioning from cooperation to competition.

3.3.1.2 Asymmetric Decays

To represent the asymmetric transition probabilities, the decay parameter $d$ was split into two separate values, $d_a$ and $d_b$, so now the transition matrix from step 0 to step 1 is:

$$
\begin{bmatrix}
P(C_1) \\
P(C'_1)
\end{bmatrix} \propto
\begin{bmatrix}
1 - d_a & d_b \\
d_a & 1 - d_b
\end{bmatrix}
\begin{bmatrix}
P(C_0) \\
P(C'_0)
\end{bmatrix}
$$

(3.1)

which multiplies out to

$$
P(C_1) \propto (1 - d_a)P(C_0) + d_bP(C'_0)
$$

(3.2)

$$
P(C'_1) \propto d_aP(C_0) + (1 - d_b)P(C'_0)
$$

(3.3)

To reason about how the two separate values of decay affect the transition probabilities, let’s focus on the first element in the resulting matrix, $P(C_1)$. We know that $P(C_1)$ is greater than $P(C_0)$ for low values of $P(C_0)$ and is lower than $P(C_0)$ for high values of $P(C_0)$. This is because the decay "decays" values towards the middle. To find the boundary value where $P(C_1)$ is no longer greater than $P(C_0)$, we set
Figure 3-1: Plots of each game sequence showing the human (solid) and model (dotted) values (using a single $\beta$ and $d$) per judgment point per player. (The yellow player is displayed in orange for better visibility.) Games 0-4 are prisoner’s dilemma games, games 5-9 are four-way games, games 10-14 are double fence games, and games 15-28 are PDF games.
\[ C = P(C_1) = P(C_0), \] which gives us

\[ C = (1 - d_a)C + d_b(1 - C) \quad (3.4) \]

Solving for \( C \), we find that \( C = d_b/(d_a + d_b) \), which we will call the cooperative decay ratio, or CDR.

When there was a single decay, the CDR was always \( d/(d + d) = 0.5 \). This makes sense, as decay pulls values that are further from the middle towards the middle, so values above 0.5 will decrease towards 0.5 and values below 0.5 will increase towards 0.5.

With separate decay values, the CDR is no longer 0.5. We can think of the CDR as the new "middle" that values are pulled towards during the transition. For values of \( C_0 < \text{CDR} \), \( C_0 \rightarrow C_1 \) will increase to be closer to the CDR. For values of \( C_0 > \text{CDR} \), \( C_0 \rightarrow C_1 \) will decrease to be further from the CDR. If \( C_0 = \text{CDR} \), then there is no change from \( C_0 \) to \( C_1 \).

By setting \( d_a \) and \( d_b \), we can set the CDR and dictate how "cooperative" the transition from \( C_0 \) to \( C_1 \) is. If the CDR is high, transitions from \( C_0 \) to \( C_1 \) will increase in value more often than they will decrease in value, which means that more values will increase to be more cooperative after the transition. If the CDR is low, transitions from \( C_0 \) to \( C_1 \) will decrease in value more often than they will increase in value, which means that more values will decrease, or become more competitive, after the transition.

If we want our model to be more cooperative-leaning in its observations, the CDR should be higher, so \( d_b \) should be higher and \( d_a \) should be lower. This means that, inversely, if we want our model to be more competitive-leaning, the CDR should be lower, meaning \( d_b \) should be lower and \( d_a \) should be higher. Setting \( C' = 1 - C \), we find that the decay ratio for \( C' \) is \( d_a/(d_a + d_b) \), which follows the previous statement.

After running the optimization using asymmetric decays, we found \( d_a = 0.086 \) and \( d_b = 0.307 \), and the scatter plot of the values can be seen in Figure 3-4. \( d_a \) is lower and \( d_b \) is higher when compared with the original \( d \) of 0.159. This gives us
Figure 3-2: A comparison between the optimized single decay and optimized asymmetric decays. The dotted line shows the transition from $C_0$ to $C_1$ using a single decay $d = 0.159$, and the solid line shows the transition using the asymmetric decays $d_a = 0.0867$ and $d_b = 0.307$. The $C_1$ for asymmetric decays is higher, showing that the judgments end up being more cooperative-leaning.

a cooperative decay ratio of $0.307/(0.086 + 0.307) = 0.775$. This means that after applying the decay, $P(C_1)$ will generally end up increasing unless it is already quite high (and even at that point, it doesn’t decrease by much). Figure 3-2 shows how the asymmetric decay values affect the updates to $P(C_1)$ and $P(C'_1)$ at different values of $P(C_0)$ compared to the single decay value $d = 0.159$. This also qualitatively reflects what we saw from the human attributions when unclear situations were generally perceived to be more cooperative.

Comparing the human and model data points in Figure 3-4, $R$ increased from 0.899 to 0.938, $R^2$ increased from 0.809 to 0.879, and the RMSE decreased from 0.107 to 0.075. By comparing Figures 3-1 and 3-3, we can see that the poorly-fit lines in games 10-14 and 23-28 are also fit much more closely, without the fit being negatively affected in the other games.
Figure 3-3: Plots of each game sequence showing the human (solid) and model (dotted) values (using a single $\beta$ and asymmetric $d$) per judgment point per player. The fit compared to Figure 3-1 is improved, indicating that these parameters are reflective of the human mental model.
Figure 3-4: A comparison of the human versus model values per judgment point using a single decay \( d \) (left) and asymmetric \( d_s \) (right). The correlation using a single \( d \) was \( R = 0.899, R^2 = 0.809, \text{RMSE} = 0.107 \). The correlation using asymmetric \( d_s \) was \( R = 0.938, R^2 = 0.879, \text{RMSE} = 0.0748 \).

<table>
<thead>
<tr>
<th></th>
<th>( \beta_a )</th>
<th>( d_a )</th>
<th>( \beta_b )</th>
<th>( d_b )</th>
<th>( R )</th>
<th>( R^2 )</th>
<th>\text{RMSE}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 beta, 1 decay</td>
<td>4.04</td>
<td>0.159</td>
<td>0</td>
<td>0</td>
<td>0.899</td>
<td>0.809</td>
<td>0.107</td>
</tr>
<tr>
<td>1 beta, 2 decays</td>
<td>2.42</td>
<td>0.0867</td>
<td>-</td>
<td>0.307</td>
<td>0.938</td>
<td>0.879</td>
<td>0.0748</td>
</tr>
<tr>
<td>2 betas, 2 decays</td>
<td>3.84</td>
<td>0.0899</td>
<td>1.55</td>
<td>0.287</td>
<td>0.942</td>
<td>0.887</td>
<td>0.0708</td>
</tr>
</tbody>
</table>

Table 3.1: A comparison of the optimized values from fitting the CCAgent model with different set of free parameters. Using asymmetric decays had a marked improvement in the fit, but using asymmetric \( \beta_s \) did not.

### 3.3.1.3 Asymmetric Betas and Decays

We also tried fitting the model using separate \( \beta \) values for cooperative and competitive planning, along with the asymmetric decay values. However, the high number of free parameters caused the optimization to take an exceedingly long time to run, without much improvement in the RMSE. A full comparison between the three can be seen in Table 3.1.

### 3.3.2 Model Comparison

The CCAgent uses two low-level planners, the CoopAgent and the CompeteAgent, for policy generation. To validate that the high-level planner using both the co-
operate and compete modes of planning was indeed the most representative of the human mental model, the CCAgent was compared against two lesioned models. The Cooperate-Random Agent used the CoopAgent and RandomAgent, and the Random-Compete Agent used the RandomAgent and CompeteAgent (see Section 2.3.1 for a detailed explanation of the different low-level planners).

Each agent’s parameters $\beta$, $d_a$, and $d_b$ were optimized by minimizing the RMSE between the experiment and model values as in Section 3.3.1. The full results of the optimized parameters can be see in Table 3.2. Even though these lesioned models were optimized, the fit was still evidently worse - this can be seen through the $R$, $R^2$, and RMSE values in Table 3.2 and the scatter plots in Figure 3-5.

Through these comparisons, we can see that the high-level planner must use both cooperative and competitive planning in order to attribute cooperative and competitive behavior in a manner similar to humans. Of the models used in the attribution experiment, the CCAgent is the most representative of the mental model humans use when attributing cooperative intention and behavior.
Table 3.2: A comparison of the optimized values using the CC-Agent, Cooperate-Random Agent, and Random-Compete Agent and how well each model fit to the experimental data. Clearly, the CC-Agent outperformed both lesioned models.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \beta_a )</th>
<th>( d_a )</th>
<th>( d_b )</th>
<th>( R )</th>
<th>( R^2 )</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate-Compete (CC) Agent</td>
<td>2.42</td>
<td>0.0867</td>
<td>0.307</td>
<td>0.938</td>
<td>0.879</td>
<td>0.0748</td>
</tr>
<tr>
<td>Cooperate-Random Agent</td>
<td>4.66</td>
<td>0.215</td>
<td>0.0781</td>
<td>0.820</td>
<td>0.673</td>
<td>0.120</td>
</tr>
<tr>
<td>Random-Compete Agent</td>
<td>3.90</td>
<td>0.0290</td>
<td>0.463</td>
<td>0.770</td>
<td>0.593</td>
<td>0.139</td>
</tr>
</tbody>
</table>

3.4 Discussion

Examining the line graphs in Figure 3-3 and the flowcharts in Appendix A, we can see people generally attributed behavior to the players in a way that we expected and that people were able to make meaningful judgments based on just a single state-action sequence. Games 0-4 were sequences played on the prisoner’s dilemma game described in Section 2.2.1 and games 5-9 were sequences played on the the four-way game described in Section 2.2.2. It is clear that participants understood which movements were cooperative and competitive, rating highly when a player waited or both players reached their goals at the same time, and rating lowly during collisions or when a single player reached their goal first.

Games 4 and 9 in particular show instances of miscoordinated cooperation. In game 4, the players collide in the center square, which looks like competitive behavior. In game 9, both players wait a turn before the intersection and then both try to move into the intersection on the next turn. In both games, the collision is seen as quite competitive, even in game 9 when both players had waited on the previous turn. This indicates that, attribution-wise, any signs of disharmony look quite competitive, even if it was a mistake of coordination. Eventually the players recover and end up in their goals at the same time, and both are seen as cooperative again. This reveals that seemingly competitive movements in the middle, which can be chalked up to miscoordination, do not affect the overall assessment that both players were behaving cooperatively. This reinforces the statement from Section 1.2.2 where miscoordination is simply a possible component of cooperation and not necessarily a detractor.
Games 10-28 all involved fences, which resulted in final outcomes that were incongruous to the actual intention of the player. However, the participants were able to judge the behavior based on the intention - a competitive outcome did not take away from cooperative intent, and vice versa. This is particularly noticeable in games 15-28, which were PDF games. As seen in Figures A-4 and A-5, the fences resulted in 2-3 different outcomes per identical game sequence. However, looking at the line graphs in Figure 3-3, we can see that for each identical game sequence (disregarding the final outcome), the participants’ judgments were very similar. This demonstrates that the human mental model does indeed look at the entire sequence of movements when assessing cooperative behavior and not just at the final outcome. Therefore, it is important that our model is able to assess the entire sequence and make judgments at each point rather than purely looking at the final outcome.

3.5 Future Optimizations

There are a few steps that can be taken to further improve the fit of the model.

There are a lot of judgment points that are equivalent in terms of the states and actions leading up to that point. It would be useful to average judgments with duplicate histories to decrease the noise and redundancy of the data. This would also likely improve the fit of the model to the data.

We made the assumption that people’s priors were 0.5, but in reality people could come in predisposed with a belief of how cooperative people generally are. The prior could potentially be pulled out as a free parameter and optimized upon as well.

Finally, we can test the robustness of the model across different types of games. First, we could re-run the attribution experiment with completely new game sequences. Then, we could use the currently optimized model to make judgments on the new game sequences. By comparing these new model predictions with the new experimental data, we can test how good of a predictor the model is on games that were not used for its optimization. We can also see if the model overfit to the previous experimental data and iterate on which free parameters should be used for
optimization.
Chapter 4

Two-Player Experiment

We also ran a two-player experiment where participants played 15 repeated rounds of the prisoner’s dilemma with fences (PDF) game detailed in Section 2.2.1 against another participant. This game combined both the features of a social dilemma and tested for intention versus outcome. The purpose of this experiment was to gain insight on how actual humans would play this game, if it aligned with our expectations of what would happen, and if our model would be able to fit these human-selected movements. This experiment was also run on Amazon Mechanical Turk (MTurk) with 92 participants, with 24 pairs of participants completing all 15 rounds of the game.

4.1 Experimental Design

The two-player experiment began with a tutorial that taught users the game rules described in Section 2.1 through a series of interactive slides. Users had to select and make moves on the grid in order to complete the tutorial and get familiar with how to play the game. To describe the collisions, the tutorial used the same playback mechanism as the attribution experiment in Chapter 3. Users were also able to navigate back and forth throughout tutorial slides in case they wanted to reread or retry anything. See https://youtu.be/ZEHFBzfoI7s for a video of the two player experiment tutorial.
After completing the tutorial, the participant would wait for up to 5 minutes to be paired with another participant. If another participant could not be found, the participant would be notified, and the experiment would end there (the participant would still be compensated for their time). If a match was found, the two participants would be paired up, and a sound would play to notify both participants that the experiment had begun.

The participants were told that they would be playing a game against the same player for 10 to 20 rounds total. The exact number of rounds (15) was withheld so that the participants could not use backwards induction to compute a competitive strategy [14]. Participants were incentivized to score points by being paid a 1 cent bonus per point they received (on top of a $1 payment for completing the experiment). In each round, the participants were given 30 seconds to make each move. Both participants played as the blue circle to remove any biases that different colors might have introduced. After both users submitted moves, the UI would animate to display the selected actions and the eventual state the participants ended up in. If a participant failed to make a move within 30 seconds, they would be disconnected, and the experiment would be over. Only the data collected from pairs where both participants successfully completed all 15 rounds were used for analysis.

At the end of 15 rounds, the participants were directed to a demographics survey where they were asked for their gender, age, and political affiliation. They were also asked to rate how cooperative the other player seemed, how much they would want to play with the same player again, and how "human" the other player seemed, and describe their overall thought process when making the moves.

To view a video of a participant completing the two player experiment, please go to [https://youtu.be/8KBc2G446aU](https://youtu.be/8KBc2G446aU).
4.2 Software Architecture

4.2.1 Front End

The two-player experiment used the same Javascript architecture as described in Section 3.2 to display the front end, but a separate subclass inheriting from Game was created so that participants could control the player in the game and select moves. The psiTurk package was used to run the experiments on MTurk.

Each client kept track of all of the states and actions that the participant selected for each round as well as how long it took for each participant to select a move. After completing all the rounds and the final demographic survey, all the data was sent to the psiTurk server to be used for later analysis.

A one-player version of the game was also created where a user could play against an agent using a single predefined policy. This version of the game was used for the tutorial when, to learn certain rules, the participant had to play against a hard-coded player that moved in a specific way. This functionality could also potentially be used to have people play against our agents to assess if our agents can pass the Turing Test and pass as a human player.

4.2.2 Back End

Because each trial of the experiment now involved multiple people, a Node.js server was used to keep track of clients, and Socket.IO was used to communicate between the server and its clients. The server stored the games and their metadata, such as the configuration, the goals and fences, and the starting state of each player.

When a client connects to the server, the server gives the client a unique socket ID and holds it in a "waiting room" until a second client connects. If no second client connects within 5 minutes, the first client is disconnected. If a second client does connect, the server pairs the two by putting them in a Socket.IO room together. Each room has a unique ID based off of the sockets’ IDs, and two maps are used to bidirectionally map each room to the sockets that it contains. Though these maps
are shared between all clients, the unique room and socket IDs guarantee that the no messages will be passed to the wrong clients. The server also assigns one participant to be "player 1" and the other to be "player 2" for each pair. Once the participants are paired up, the server sends the game metadata to the clients so that the web interface can properly display the game.

Each participant then selects their next action, which is sent to the server along with their current state. The server uses a maps mapping from socket IDs to submitted moves to keep track of whether or not two clients in a room have both submitted moves. When it receives a client’s move, it checks the map to see if its partner has also submitted a move. If not, it will send a waiting message back to the client. If the server does not receive a move within 30 seconds, the client that did not submit a move is disconnected, and their partner is sent to the final demographic survey. Otherwise, if the server has received moves from both clients, it will then calculate what the next state is for each player. The next state is generally the same as the selected action, except when a collision occurs or a fence is crossed. In these cases, the server probabilistically determines what the next state is. The joint next state is sent to each client, which then display the animation from the current state to the next state in the UI. The server does not store any history of states or actions - the games are designed such that, to determine the next state, the server only needs the current state and next actions. Figure 4-1 shows the basic relationship between the server and the client.

4.2.3 Data Analysis

All of the data analysis was done in Python. NumPy, SciPy, and pandas were used to manipulate the data, and Matplotlib and seaborn were used for plotting and visualizing the data.
Figure 4-1: The relationship between the server and its clients, designed such that minimal information needs to be passed back and forth. When a player makes a move on step $i$, the client will send its current state and the selected action to the server. Once the server receives the message from both players, it will calculate the next joint-state and send it back to each client.

### 4.3 Results and Analysis

The experiment was run on 92 participants total, but only the data collected from pairs that completed all 15 rounds were used for analysis. This resulted in 24 pairs of complete data. The data was incredibly heterogenous, with pairs’ behavior ranging from only cooperating to only competing, with everything in between.

A web user interface was built to visualize the experimental data. In the UI, you could select any pair and view played-back animations of all 15 of their game sequences as well as a graph per game sequence that showed observer CCAgents’ beliefs of cooperativeness per participant per movement. The graph provided a quicker visualization of the participants’ behavior without necessarily having to watch the entire game sequence. The UI containing the 24 pairs of data can be seen at [http://web.mit.edu/zlily/www/meng/twoplayer-data/](http://web.mit.edu/zlily/www/meng/twoplayer-data/)
Table 4.1: A table showing which free parameters each model depended on.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\beta_a$</th>
<th>$d_a$</th>
<th>$\beta_b$</th>
<th>$d_b$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAgent</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>RandomAgent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoopAgent</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompAgent</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RandomCCAgent</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutcomeAgent</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: A comparison between using different sets of free parameters. As in Section 3.3.1, a single $\beta$ and asymmetric decays was found to improve the fit and also run within a reasonable amount of time.

### 4.3.1 Parameter Optimization

The parameters were optimized by using SciPy’s optimize method to minimize the negative log likelihood of the data (the calculation of this value is explained in Section 4.3.2). As in Section 3.3.1, we tried using a single $\beta$ and decay, a single $\beta$ and asymmetric decays, and both asymmetric $\beta$s and decays. In the attribution experiment, the model’s observations were bounded by the minimum and maximum mean judgment from the data. To mimic that behavior and account for noise, the bounds $b$ used to scale $P(C)$ and $P(C')$ for policy weighing in Equation 2.10 was also pulled out as a free parameter, where $0 \leq b \leq 0.5$. Table 4.1 shows which parameters each agent depended upon. We found that a single $\beta$ and asymmetric decays were able to improve the fit without taking excessively long to run, just as we had found in Section 3.3.1. The various parameter combinations with CCAgent can be seen in Table 4.2.

### 4.3.2 Metrics of Comparison

The next step was to determine which agent model best fit the human mental model. We wanted to find how accurately our model could predict the next selected action, given a certain set of states and actions. To do so, each participant’s 15 game se-
sequences were treated as one long sequence, which we will call a player’s data $D_p$, defined by $(s_0, s_1, ..., s_{n+1})$ and $(a_0, a_1, ..., a_n)$ for $n$ total moves. Because they were playing the same game against the same player, it follows that the posterior probabilities from the previous round would become the priors for the next round, which is why stitching together game sequences is valid.

We compared 6 different optimized models, CCAgent, RandomAgent, CoopAgent, CompeteAgent, RandomCCAgent, and OutcomeAgent (see Section 2.3 for a full description of the models and Table 4.3 for each model’s optimized parameters).

<table>
<thead>
<tr>
<th></th>
<th>$\beta$</th>
<th>$d_a$</th>
<th>$d_b$</th>
<th>$b$</th>
<th>ln(likelihood)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAgent</td>
<td>3.95</td>
<td>0.0953</td>
<td>0.0001</td>
<td>0.269</td>
<td>.1051.07</td>
</tr>
<tr>
<td>RandomAgent</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-2320.36</td>
</tr>
<tr>
<td>CoopAgent</td>
<td>2.06</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-1814.23</td>
</tr>
<tr>
<td>CompeteAgent</td>
<td>0.93</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-1112.32</td>
</tr>
<tr>
<td>RandomCCAgent</td>
<td>3.69</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-1119.93</td>
</tr>
<tr>
<td>OutcomeAgent</td>
<td>3.70</td>
<td>-</td>
<td>0.438</td>
<td>-</td>
<td>-1113.79</td>
</tr>
</tbody>
</table>

Table 4.3: The optimized parameters for each model. Both CoopAgent and CompeteAgent have very low $\beta$s, which mean they are quite noisy models. In comparison, the high-level agents have higher $\beta$s, indicating they are able to fit to the two player data without needing to introduce much noise.

For each step $i$ from 0 to $n$ in each $D_p$ (where $p$ represents each player), each model $M$ would "play" the game using the HMM in Figure 2-5 with the predefined states $(s_0, s_1, ..., s_{i+1})$ and actions $(a_0, a_1, ..., a_i)$ up to step $i$. After the final action $a_i$, the model would then update its belief as described in Section 2.3.2.1 and generate a policy $\pi^M_{i+1}(a|s)$ that contains the probabilities of all next actions $a$ given state $s$. Because $\pi^M_{i+1}$ was generated using all the previous states and actions, it encoded the entire history up to and including step $i$. $\pi^M_{i+1}$ was then used to find the likelihood $P^M(a = a_{i+1}|s = s_{i+1})$ of the next predefined action $a_{i+1}$ from the player’s data $D_p$. If the agent was close in modeling the human mental model, then $P^M(a = a_{i+1}|s = s_{i+1})$ would be high. Inversely, if the agent’s policy is nothing like the human mental model, then $P^M(a = a_{i+1}|s = s_{i+1})$ would be low.

To calculate how likely the data per player $D_p$ was given each model $M$, we
multiplied all of the likelihoods per player, formalized by:

\[ P(D_p|M) = \prod_{i=1}^{n} P^M(a = a_i | s = s_i) \quad (4.1) \]

We start with \( i = 1 \) to skip the likelihood generated by our preset prior of 0.5 in order to not bias the data.

To find how well each model \( M \) performed given the data, we used Bayes’ theorem:

\[ P(M|D_p) = \frac{P(D_p|M)P(M)}{P(D_p)} = \frac{P(D_p|M)P(M)}{\sum_{M=\text{all\_models}} P(D_p|M)P(M)} \quad (4.2) \]

Though we do not know \( P(M) \), we assume a uniform prior across all models, which means that all \( P(M|D_p) \) per model \( M \) is factored by the same constant \( \frac{1}{\text{number\_of\_models}} \).

Figure 4-2 shows a graphical representation of the model likelihood for each player, and it’s clear that the CCAgent was the most representative. Appendix B contains graphs that provide an in depth insight of the behavior per pair of participants and how well each model represented each individual participant.

To find the likelihood of the entire dataset \( D \) for each model \( M \), all the likelihoods per participant were multiplied, or:

\[ P(D|M) = \prod_{p=1}^{P} P(D_p|M) \quad (4.3) \]

Where \( P \) is the total number of participants (48 for this experiment specifically).

To find how well each model \( M \) fit the entire data \( D \), or the model likelihood \( P(M|D) \), we used Equation 4.2 substituting \( D_p \) (the data per player) for \( D \). We used this model likelihood to compare how representative each model was of the data as a whole. The leftmost plots in Figure 4-3 shows a comparison of the data likelihoods \( \ln(P(D|M)) \) (the natural log was used because the original values were too small to plot) and model likelihoods \( P(M|D) \) per model. For this analysis, we considered a higher data likelihood, or a log data likelihood closer to 0, to mean a better fit.
Figure 4-2: A plot of each model’s model likelihood as a stacked bar for each participant, sorted by how representative CCAgent is of that participant. This provides an overview of how the different models captured the human mental model at a granular, per player level. CCAgent is the dominant model, occupying the most area in this graph.
of the experimental data. The high-level models, CCAgent, RandomCCAgent, and OutcomeAgent all generally outperformed the low-level models, RandomAgent and CoopAgent, with the exception of CompeteAgent which outperformed both Random-CCAgent and OutcomeAgent. However, referencing Table 4.3, CompeteAgent had an extremely low $\beta$ value, meaning that the model was quite noisy, which could explain why it was able to capture more than the other two models. However, looking at the model likelihoods, CCAgent was still the best-performing model, as we had anticipated.

Another interesting metric to explore was how "correct" each model was in predicting the next action. To find the average probability of "correctness", the likelihood per action was summed up and divided by the total number of actions. A bar graph comparing each model can be see in Figure 4-4. For most of the models, the numbers are quite similar - that is because for most of the games, there are only a few specific states where the cooperative and competitive policies actually differ in next action probabilities. For many states, the most likely thing to do is to go towards the goal, regardless of strategy employed. Because RandomAgent weighs all next actions equally, it makes sense that its average probability of correctness is simply the reciprocal of the average number of next actions, which checks out to equal between 3-4 number of next actions per state. The average probability of correctness follows the
Figure 4-4: The average likelihood of selecting the participant’s next action for each model, a metric of "correctness." CCAgent is a hair higher than CompeteAgent, RandomCCAgent, and OutcomeAgent, but that can be attributed to the fact that at certain states in the game, there is only one possible, rational next movement, and that is captured by all four of these models.

same general distribution as the log likelihoods from 4-3, with a higher average correctness corresponding to a log likelihood closer to 0. A potential optimization would be to identify the states where the models actually predict differently and specifically use those states to measure the correctness of the predictions.

4.3.3 Model Comparison

We wanted to further validate specific aspects of CCAgent. As described in Section 2.3.2.1, the CCAgent is special for a variety of reasons: 1) it is a high-level planner that uses two separate low-level planning policies, 2) it makes observations after each movement to update its belief state of the other player’s behavior, and 3) to get an accurate assessment of the other player’s behavior, it assesses every movement, not just the final state. To validate these aspects of the CCAgent, three specific agent
Figure 4-5: A closer look at specific model comparisons. The top row shows the log data likelihoods for each model (the colors correspond to Figures 4-2 and 4-4), and the bottom row shows the model likelihoods. From left to right, the comparisons show CCAgent vs the low-level models, CCAgent vs RandomCCAgent, and CCAgent vs OutcomeAgent. It is evident that CCAgent is the best fit to the experimental data, especially when comparing the model likelihoods.

comparisons were conducted: 1) CCAgent versus the low-level planners (Cooperate Agent, Compete Agent, and Random Agent), 2) CCAgent versus RandomCCAgent, and 3) CCAgent versus OutcomeAgent. In each of these comparisons, CCAgent was by far the most representative of the data, as seen in Figure 4-5. This demonstrates that the extra computations the CCAgent performs are necessary steps in representing the human mental model.

4.4 Discussion

The data collected from participants playing against each other was highly variable, which can be see from the line graphs in Appendix E.

The purpose of running the PDF game for 15 rounds was to both see how people would perform in an iterated prisoner’s dilemma situation and if, with the fences, participants could still reach a state of robust cooperation despite the outcome not necessarily reflecting the intention. Overall, there were 11 pairs that reached some
form of robust cooperation by at least round 8 of 15, which is a little less than half of the pairs. Though there were some pairs that played the game as we had expected, there were also many that did not.

Because of the incentivization to earn points, participants could have been more focused on receiving a high score per round than strategizing about how to earn more points over the long term. Therefore, they might have defaulted to competing on every round. Some people started out cooperative, but whenever they saw that they did not make it into their goal, they began competing, even if their opponent was acting cooperatively. Therefore, we can reason that in the population, especially under these conditions, some people’s mental models of cooperation are actually best captured by the OutcomeAgent. It also appeared that some players didn’t care about earning points and just wanted to get through the experiment as quickly as possible. Furthermore, many participants did repeatedly try to cooperate, but their partner would not comply.

However, despite all of the variance, the CCAgent was still able to do a good job of fitting the experimental data, far outshining the other agents. This reinforces the fact that a model that can make observations, update its belief, and plan accordingly is indeed necessary to represent the vast flexibility of the human mind.

4.5 Future Optimizations

When analyzing the model likelihoods of the models for individual players, it was evident that CoopAgent and CompeteAgent fit quite poorly, even for players that were clear cooperators or competitors. This is likely due to the low $\beta$s, which meant that models were noisier and thus more random than behavior-specific, as they were originally intended to be. This was a result of using agents that could only do one type of planning to fit very diverse data. Noise was introduced in the model to handle the behavior that was outside of its planning capabilities, which caused the CoopAgent and CompeteAgent to be mediocre fits of everything rather than fitting a few specific players well.
To account for this, a future step could be to run the optimization of each model on each individual player to find out which model best represents each individual player rather than the average player overall. We could also sum up the total number of players best represented by each model and compare that with the overall distribution to see if the model that fits the highest number of individual players also fits the average player well. This may also reveal a dichotomy between the kind of mental model used for observing behavior versus actually behaving. We may want a flexible model that can read many different types of behavior, but it could be that humans are more ingrained in their ways and prefer to stick with a more specific, narrow model when selecting their own actions. By having the optimized model per player, fine grained comparisons between different agents such as the ones from Section 4.3.3 could also be done on a player level. We can also use this information to visualize the model likelihood for each step in the game to help gain a better understanding of the relationship between specific actions and which models they are best captured by.

A possible iteration of this experiment to have participants play a different game each round, but the games would still all have the same underlying social dilemma structure and payoff matrix. For specific games, there could be certain behavioral heuristics in place that map to cooperation and competition, undercutting any actual need for planning. For example, in the prisoner’s dilemma game, going to the outside goals can always be seen as cooperative and going to the inside goals competitive. Having a different game each round could do away with any behavioral heuristics that might be learned from repeatedly playing the same game and instead test if learned strategies between players can be abstracted across different games. Having a different game per round could also prevent the CCAgent model from overfitting specifically to how people are playing one single game and also demonstrate our model’s flexibility and diverse capabilities that the lesioned models do not possess.
Chapter 5

Conclusion and Future Work

Our goal was to mathematically represent the mental model humans use in problems of cooperation in order to better understand the mechanisms at play when humans attribute behavior and execute actions in dual-agent interactions. Using Bayesian modeling and reinforcement learning, we built the CCAgent, a high-level planner that can infer abstract social intentions and use those inferences to weigh two low-level modes of planning, cooperative planning and competitive planning.

To validate this hierarchical model, we ran two experiments, the attribution experiment and the two player experiment. In the first experiment, participants watched played-back sequences of stochastic grid games and evaluated how cooperative each player seemed. In the second experiment, participants played these stochastic grid games against each other. We used the data collected from both of these experiments to gain more insight on how humans inferred social goals and acted in problems of cooperation. We also compared our model with lesioned models and found that the model that used both modes of planning and inferred social intentions far outperformed the rest in terms of fitting the experimental data.

Though this work has been illuminating, there are still many interesting directions to take this research to both fine-tune the current model and explore other aspects of cooperation in dual-agent interactions.

One next step would be to run an experiment where participants play the games against the newly optimized CCAgent. As mentioned in Section 3.2, there is already
code in place for a participant to play stochastic grid games against an agent using a predefined policy. However, further work is needed to build an agent in Javascript that can observe and update the policy after each action. This experiment would serve as a Turing Test to see how human-like our model is and if it could fool participants into thinking that they were playing against a human player.

A different angle to approach the two experiments is to use the attribution experiment as the training set to optimize the model and the two player experiment the test set to validate the model. After running parameter optimizations to fit the model to the data collected from the attribution experiment, the same model with the same parameters would then be used to predict next actions in the two player experiment. This would test the robustness of the model’s fit and also whether or not the model used for attribution translates to the model used to execute actions. It could be that attribution and response, when associated with different sets of stakes, may not be modeled in the same way.

Another interesting area to explore is if some people are systematically better attributers than others and if that affects how people play the game. This would tie together the attribution and two-player experiments. Participants would first make a series of judgments on predefined game sequences, followed by actually playing the game against another participant. People’s "accuracy of judgment" can be found by taking the RMSE between all of an individual participant’s judgments and the sample mean judgments. Figure 5-1 shows this metric for the data collected in Section 3 and there is clearly a non-trivial spread. Then, we would want to explore if there exists a relationship between a low RMSE, which means the participant judged closely to the sample mean, and game-play behavior. Specifically, we are interested in if the participant would be a robust cooperator, which means that they and their partner would defer to a cooperative strategy in repeated rounds. Currently, in the data collected in 4, 11 of 24 pairs exhibited some degree of robust cooperation. If the two are correlated, that would corroborate the idea that, to be a better cooperator, one must first understand how to read and assess another’s behavior, which can be seen as a form of emotional intelligence.
Figure 5-1: A histogram showing the RMSE between each participant's judgments and the sample mean judgments in the attribution experiment, or their "accuracy of judgment." The distribution is right-skewed, with the highest density between 0.2 and 0.3, showing that there exists a range of accuracy. This could potentially be used to determine if there's a correlation between how accurately people make judgments and how they play the grid games.

Finally, it would be interesting to explore if the CCAgent could infer behaviors beyond just cooperation and competition, such as the behaviors discussed in Section 1.2.4. By analyzing the two player experimental data in Appendix B, we found various possible high-level social intentions the participants seemed to be exhibiting, such as teaching, punishing, and forgiving.
Appendix A

Attribution Experiment Data

The following flowcharts show all of the game sequences from the attribution experiment as well as the specific judgment points and the game state at that judgment point. For each judgment point, the associated bar graph shows how the human beliefs of cooperation (colored bars) compare with the model beliefs (black bars) using the CCAgent with parameters optimized in Section 3.3.1.
Figure A-1: This flowchart shows all 5 game sequences for the prisoner’s dilemma game. The sequences from top to bottom correspond to games 0, 3, 1, 4, and 2 from the experiment.
Figure A-2: This flowchart shows all 5 game sequences for the four-way game. The sequences from top to bottom correspond to games 5, 7, 6, 8, and 9 from the experiment. Game 9 is notable because there is a high degree of miscoordination between the two players before they eventually reach the cooperative outcome.
Figure A-3: This flowchart shows all 5 game sequences for the double fence game. They correspond to games 10, 11, 12, 13, and 14 from the experiment.
Figure A-4: This flowchart shows the first 8 game sequences for the prisoner’s dilemma with fences game, corresponding to games 15-22 from the experiment. Each game sequence from [A-1] now has 3 possible outcomes: both players reach the goals, only blue reaches the goal, and only yellow reaches the goal. This resulted in 14 total game sequences (one was removed due to symmetry).
Figure A-5: This flowchart shows the last 6 game sequences for the prisoner's dilemma with fences game, corresponding to games 23-28 from the experiment.
Appendix B

Two Player Experiment Data

The graphs below show the complete data collected from the two player experiment. For each pair of participants, there is a line graph that shows the observed beliefs of cooperativeness for each player at each move using the CCAgent with the optimized parameters from Section 4.3.1. The vertical lines mark the end of each round, and the color signifies the color of the player that "won" that round, or black if the two players jointly reached their goals. There is also a bar graph that shows how representative each model was (using the model likelihood as calculated in Section 4.3.2) of each participant in the pair.
Bibliography


