TÉCNICO
LISBOA

Coup Agents: Agents that play a deception game

Filipe Luís Felício Fortes

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Ana Maria Severino de Almeida e Paiva

Examination Committee

Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. Ana Maria Severino de Almeida e Paiva
Member of the Committee: Prof. Francisco António Chaves Saraiva de Melo

November 2015

I dedicate this work to my family.

Acknowledgments

Many people have helped to make this work possible. Firstly I thank my thesis advisor, Prof. Ana Paiva, for giving me the opportunity to work with amazing people and making me believe that we truly can accomplish great things. Thanks to Tiago for the constant help with Thalamus and making that EMYS work, thanks to Eugenio for always being up to help and for making EMYS just randomly talk, I had a blast. Thanks to both Patricia and Sofia for being the best psychologists ever, for helping with the experiments and the analysis of their results. Thanks to Filipa and Nuno for all the help and laughs we had together. Thanks to Rui for dealing with my randomness throughout the days and the crazy ideas of going to work at four in the morning, which is still hard to believe that it was done.

Thanks to all the GAIPS team for providing a great environment of work and development. Thanks to all the participants of our experiments that helped contributing to this work. Thanks to Instituto Superior Técnico for providing me with the opportunity to learn and grow as a person.

Thanks to my cats for sometimes disrupting my work, I obviously needed a break those times. Thanks finally to all my family for the constant support throughout the years.

Resumo

Este trabalho investiga como construir um agente que, implementado numa entidade robótica, é capaz de jogar um jogo social de decepção, chamado COUP, ao mesmo nível de humanos. Para que tal seja possível, começámos por primeiro definir o problema em geral, seguido pela definição da solução a esse problema. Produzimos depois um algoritmo que implementámos no nosso agente de forma a tomar as decisões para o jogo do COUP. Este algoritmo é baseado na minimização de arrependimento contrafactual. Seguidamente, implementámos um sistema completo baseado na solução mencionada anteriormente com o objectivo de testarmos a nossa hipótese. Este sistema incluí a arquitectura do nosso agente, a interface do jogo e uma parte relativa ao robot que usámos para representar o nosso agente no mundo real, o EMYS. Para provarmos a nossa hipótese, criámos uma experiência com quatro diferentes condições, uma condição de grupo e uma individual, e uma condição de mentira, onde o agente era capaz de mentir, e uma condição de verdade, onde o agente só poderia agir de forma verdadeira. Em todas as condições o nosso agente foi capaz de jogar ao mesmo nível dos humanos, com excepção da condição individual em que podia mentir, onde foi levemente melhor jogador que os seus oponentes humanos. Isto prova que é realmente possível um agente artificial jogar um jogo de decepção ao mesmo nível de um humano. Finalmente, este documento também apresenta algumas sugestões para trabalho que pode ser realizado no futuro com o objectivo de melhorar este projecto em geral.

Palavras-chave: Decepção, Robot Social, Coup, Jogos de Tabuleiro, Interação Humano-Robot

Abstract

This work investigates how to build an agent that, by being implemented in a robotic entity, is capable of playing a deceptive social game called COUP at the same level of humans. To be able to do so, we first start by defining the overall problem followed by how our solution should be. We then produce an algorithm to make the decisions for the COUP game, which is based on minimizing counterfactual regret. We also implement a whole system based on the before mentioned solution that we defined, so that we can test our hypothesis. This system includes our agent's architecture, the game interface and EMYS, which is the robot we use to physically represent our agent. To prove our hypothesis we devise an experiment with four different conditions, a group and individual condition, and a lie and truth condition, where our agent is allowed to lie, lie condition, and where it is only allowed to act truthfully, truth condition. In every condition, our agent is capable of playing at the same level of humans, with the exception of the individual lie condition, where he is slightly better at playing the Coup game than its human opponents. This proves that it is indeed possible for an artificial agent to play a deception game at the same level of humans. Finally, this document also provides some suggestions for future work to be done in order to improve this overall project.

Keywords: Deception, Social Robot, Coup, Board Games, Human-Robot Interaction

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deception has been used by various types of life forms to achieve different objectives[1], for example, chameleons use deception as a defense mechanism. While usually carrying a negative connotation, many studies show that it appears everyday in our lives[2], ranging from little lies, such as telling someone that he/she can have our dessert because we are full, when we are not, so that the person would not feel guilt for eating it, to bigger, more dangerous lies, that is a lie about infidelity.

Since deception is so commonly used among humans, it should be taken into consideration when building a robotic agent capable of social interaction. But that is not the sole reason that it should be considered, if we go into the most general definition of the word, "a false communication that tends to benefit the communicator"[1], we can see that by adding deception into the architecture of such robotic agent, we open a whole new world of possible outcomes for such entity. An example of this would be a situation where a robotic agent tells a criminal that he is looking for the keys of the safe, while in reality is waiting for the police to reach his position.

However, it is not always easy to find such situations where one can lie multiple times without having social repercussions. A game constitutes the perfect option for studies about deception, because not only it allows a rich environment for deception, but also provides situations where one can lie without being affected by it on his/her social life.

Coup is a turn-based social game, that can played by two to six people, where one uses deception to maintain his/her influence (face down cards) while making other players lose theirs. The deception used in this game is mainly towards what are the cards that each player has face down, making it a great game to build a deceptive robotic agent upon, since communication options are limited but still allows for a great deal of deception.

Using the Coup game, this thesis addresses the general problem of creating deceptive behaviors in a robot or an agent. More concretely, we will address the following question: "Is it possible to create an agent that, by implementing it on a social robot, is capable to play the Coup game at the same level of humans?". Which in turn, we predict that it is indeed possible to do so.

To answer the before mentioned question and to actually know if our prediction is indeed true, we have built an agent architecture that takes into account the possibility of deception and uses a decision making algorithm to produce actions for the Coup game. We have also built a complete system that our agent can use to both play the Coup game against human players and interact with them through the use of a social robot.

The rest of this thesis is divided into the following chapters: Chapter 2 will present a background on topics related to this thesis but that are not that common in the Computer Science area, Chapter 3 is a compilation of the most important works done in this area, providing a strong initial step for our work, Chapter 4 is used to describe in a more general way what our problem actually is and how can we define a solution for it, next is Chapter 5 where we will show the algorithm that we have developed that is capable of using deception to produce actions that will give the most advantage to our agent in every step of the game, Chapter 6 will then present our implementation of the solution defined before, after that Chapter 7 will present the experiments that we have done with the purpose to answer the question that this works addresses, finally, we will present in Chapter 8 the conclusions that we can take from this work, which will include what we achieved and some future work to continue on this intriguing work topic.

Chapter 2

Background

2.1 Deception

Deception can be a mean to an end but it usually carries a negative connotation among humans. Nevertheless, the animal kingdom is filled with all kinds of deception, ranging from mimicry and camouflage, to the more commonly seen feigning death. All these mechanisms, actively or passively used by animals, have been developed during the evolution of the species and is what allows a big part of the animal kingdom to survive against their predators.

This shows that deception, at least when used in the animal kingdom, represents an evolutionary advantage for the deceiver. Further, some researchers even point out that deception is a strong potential indicator of theory of mind[3] and social intelligence[4].

So what is deception? Along this work we will be using the definition for deception provided by Bond and Robinson, and they describe it as "a false communication that tends to benefit the communicator"[1]. This definition allows for all kinds of deceptions that would be seen in biology.

When applied directly to the human being, deception can have various types. These types include: (1) intentional and accidental deception, depending if the person deceiving is conscious of it or not; (2) active and passive deception, active when the deceiver provides the information, in some way, to deceive, or passive when the deceiver simply lets the deceived keep believing something that is false; (3) implicit and explicit deception, being implicit when it is used as behavioral communication, for example, appearing confident while having a bad hand in poker, or explicit, such as telling a lie; and finally (4) strategic deception, for instance, when a poker player checks instead of raising the bet, so that the other players will think that his hand is not as strong as it really is.

One important aspect of deception is to know when someone is trying to deceive, or still, how to deceive without being "caught". Paul Ekman, an American psychologist who has dedicated his life to the study of emotions and deception detection, believes that lie detection can be performed through the observation of facial expressions, in a way to know which emotions is the person feeling, and then comparing them with the emotions that the person is trying to show through speech acts[5]. He is also helping developing a computer program that does exactly that, through the use of computer image analysis[6].

Other ways of detecting deception pass by detecting the signals usually sent by the deceiver when deceiving. These include making facial expressions that differ from what the person is trying to say that he/she is feeling, as mentioned before; the deceiver's body language and even the volume of his/her voice and the pauses throughout the speech.

2.2 Board Games

A board game, in its most general definition, is a game that involves the placing or moving of some kind of objects on a pre-marked surface called "board", according to a given set of rules.

Board games date back to ancient times and, by the depiction in ancient drawings and carving, they present some of earliest evidence of humans playing[7]. They first started by being made out of the most common materials, like wood and stone, but as they got more popularity and continuously became a bigger part of the culture of different civilizations, game sets built from more exquisite materials, as ivory, started to being build for the pleasure of the ruling entities, such as kings and pharaohs.

Since those ancient times, board games have had an evolution, the older games, such as the "Mancala"[1] type games, have been developed into more variants, and new games were created. These new games come in a complete range of categories, going from economic simulation games, like the famous Monopoly game[2], where players go against each other, to more cooperative games, such as Pandemic[3].

The real magic to these kinds of games is that the players, not only are having fun when playing the game, but are also learning how to deal with situations that are not so common in their normal lives and gaining skills that would be much harder if learnt in another way. Studies show that by playing linearly arranged board games, children have improved their spatial numerical understanding[8], but that is not all, other studies also show that playing board games can also augment the rationality of a player, while reducing their feeling of depression[9].

Board games can also be a way of escaping reality and the normal day-to-day life, it is a way to explore fantasies while not having those affect the way a person lives. A person can lead a life of honesty, making it unacceptable to lie during the day-to-day life. By playing a deception game, this person can be rewarded for clever lying and experience what deception really is, without the need to change their lifestyle.

Deception board games encourage the players to use deception to achieve their aims and always have an element of hidden information in them. They are usually linked to party games as they encourage social interaction, and most of the deception types require some sort of interaction between people. The game Werewolf and its variant Mafia are one of the most known deception games. It is a game where there are two parties, the werewolves and the townspeople, with differing objectives: the werewolves want to kill the townspeople and the townspeople the werewolves. The catch in this game is that the townspeople don't know who the werewolves are, so the werewolves have the objective of blending in with them, using deception to do so.

Another game that resembles the Werewolf game is The Resistance game, where the werewolves are the spies and the townspeople are the resistance. The mechanics are slightly different so that no player elimination happens during the game. Coup is a game from The Resistance's family, that provides a completely different gameplay, while still allowing a great deal of deception.

[1]http://en.wikipedia.org/wiki/Mancala
[2]http://en.wikipedia.org/wiki/Monopoly_(game)
[3]http://en.wikipedia.org/wiki/Pandemic_(board_game)

2.3 COUP

Coup is a social board game that revolves around secret identities, deduction and deception. The game is played by 2 to 6 people and contains 15 character cards (3 each of Duke, Assassin, Captain, Ambassador, Contessa) and a currency used to execute certain actions, called coins.

Initially all the character cards are shuffled and then 2 are dealt to each player. Players can always look at their cards but must keep them face down in front of them. The remaining cards will be kept in the middle of the play area as the Court deck. Each player also receives 2 coins which they have to maintain visible so that other players know exactly how much anyone has. Remaining coins are put in the middle of the play area as the Treasury.

The face down cards of each player represent their influence. Every time a player loses an influence, they have to turn over and reveal one of their face down cards. When losing influence, a player has always to choose which of their own cards they wish to reveal.

When a player has lost all their influence, they are exiled and out of the game and will have to return all their coins to the Treasury. This leads to the goal of the game: *To eliminate the influence of all other players and be the last survivor!*. This does not necessarily mean that the winning player has to be the one to eliminate all the other players' influence, but he has to at least eliminate the last influence from the remaining player not counting with himself.

In terms of game play, the game is played in turns in clockwise order, where each turn a player has to choose one and only one action. After the action is chosen, other players have an opportunity to challenge or counteract that action, and if such action is not challenged or counteracted, it automatically succeeds. Challenges are resolved first before any action or counteraction is resolved.

There are a total of 7 actions that can be played, where 4 are character actions, which require the influence of specific character cards, and the other 3 are general actions that have no character requirements to be played. If a player chooses to play a character action, that player must claim that the required character is one of their face down characters, which can be the truth or a bluff. The player does not need to reveal any of their face down cards, unless they are challenged. If no challenge is made, the player automatically succeeds in using the action.

The general actions available are:

- **Income**: The player takes 1 coin from the Treasury.
- **Foreign Aid**: The player takes 2 coins from the Treasury. (*Can be blocked by the Duke*)
- **Coup**: The player pays 7 coins to the Treasury and launches a Coup against another player. That player immediately loses an influence. A coup is always successful. If a player starts his/her turn with 10 (or more) coins, he is required to use the Coup action.

The character actions, where if challenged a player must show they influence the relevant character, are:

- **Duke - Tax**: The player takes 3 coins from the Treasury.
- **Assassin - Assassinate**: The player pays 3 coins to the Treasury and launches an assassination against another player. If successful, that player loses an influence. (*Can be blocked by the Contessa*)

- **Captain - Steal**: The player takes 2 coins from another player. If the target player has only 1 coin, take only 1 coin. (*Can be blocked by the Ambassador or the Captain*)
- **Ambassador - Exchange**: The player exchanges cards with the Court. First the player takes 2 random cards from the Court deck and then chooses which, if any, to exchange with his/her face down cards. Then returns two cards to the Court deck.

In terms of counteractions, they operated like character actions. Players may claim to influence any of the characters (having them face down) and use their abilities to counteract another player, which may be the truth or a bluff. Like the character actions, a player does not need to show any cards unless challenged, and if the counteraction is not challenged, it automatically succeeds. If an action is successfully counteracted, the action fails but any coins paid as the cost of the action remain spent.

The counteractions available are:

- **Duke - Counteraction**: Any player claiming the Duke may counteract and block a player attempting to collect foreign aid. The player trying to gain foreign aid receives no coins that turn.
- **Contessa - Blocks Assassination**: The player who is being assassinated may claim the Contessa and counteract to block the assassination. The assassination fails but the fee paid by the player for the assassin remains spent.
- **Ambassador/Captain - Blocks Stealing**: The player who is being stolen from may claim either the Ambassador ir the Captain and counteract to block the steal- The player trying to steal receives no coins that turn.

As mentioned before, it is possible to challenge both the character actions and the counteractions. The challenges can be issued by any other player regardless of whether they are the involved in a action. Once an action or counteraction is declared other players must be given an opportunity to challenge and once the play continues, challenges cannot be retro-actively issued. If a player is challenged they must prove they had the required influence by showing the relevant character is one of their face down cards. If they are not able to prove it, or do not wish to, they lose the challenge, otherwise the challenger loses. Whoever loses the challenge, immediately loses an influence. If a player wins a challenge by showing the relevant character card, they first return that card to the Court deck, re-shuffle the Court deck and take a random replacement card, then the action or counteraction is resolved. If an action is successfully challenged the entire action fails and any coins paid as the cost of the action are returned to the player.

Finally, and taking into account the well defined rules explained before, we can say that Coup provides one of the best blends of deception and board games, while still providing a simple rule set and well defined actions, making it a great game to base our work on.

Chapter 3

Related Work

This section has the objective to show some of the work that has already been done in the area of agents with deception. While they do not focus directly on using such deception for social board games, which is our objective, they are the closest to it in today's state-of-the-art.

We first start with three cases that implement deception on virtual agents, we then focus on a case that has been used on robotic agents, and finally, a case that provides a test-bed for deceptive agents.

3.1 MOUTH-OF-TRUTH

MOUTH OF TRUTH is a tool, developed by Carofiglio, Castelfranchi and de Rosis[10], that is capable of simulating how a communicative act may be constructed, based on taking advantage of the uncertainty of a target, through the use of four different factors.

Letting S denote the Sender (potential deceiver), R the Receiver and let $B_S p_i$ and $G_S p_i$ denote, respectively, the belief and goal of S concerning p_i, the deception object, while $\neg BW_S p_i$ denotes the ignorance of S towards p_i. For R, similar notations were used. To denote second-order beliefs and goals, they use a combination of the items defined before. An example of such combination is $B_S B_R p_i$ which denotes the belief of S towards the belief of R concerning p_i. The problem considered by the tool may be modeled in the way that follows.

Given (a) a domain; (b) a domain reasoning model of S, that consists on the beliefs of S about the domain, named OWN-S and represented in a bayesian network; (c) a set T containing the domain facts that R knows, as seen by S; (d) a deception object p_i that represents the goal to which S wants to change the belief of R in, plan a communicative act that permits S to achieve some goal regarding R's belief of p_i (e.g. $G_S \equiv (G_S B_R p_i)$) through the use of some deception medium q, which is also a fact.

The tool, to reach such communicative act in the end, goes through a complete process, first it builds a model of the mental state of R, called IM-S-R, based on the input it receives regarding what R says it knows. This model is an overlay of OWN-S with a smaller number of nodes and both models have the same probabilities for the nodes they share. After having such model built, the truth values from the set T will be propagated so that S is enabled to see the probability that R would give to a given fact p. Finally S will choose the deception candidate medium q based on four different factors[11].

The four factors mentioned before are the following:

(a) *efficacy* in achieving the goal G_S, calculated as the *impact* that the deception medium q will have on the deception object p, which in turn is computed by performing two different calculations, one that calculates the degree of achievement of the goal G_S towards R's belief in p: $Imp^1(q,p) = 1 - |G_S B_R P(p) - B_S B_R P(p|q)|$, where $P(p)$ is the probability of A believing in p and $P(p|q)$ the probability of A believing in p given q, and other that focus on the change in R's belief towards p that is caused by believing in q: $Imp^2(q,p) = B_S B_R P(p|q) - B_S B_R P(q)$;

(b) *plausibility* of q to R, that represents how likely is R of getting convinced about q after having it communicated by S. To implement this, two different functions will be used: one used to represent the *local plausibility* as the difference between what S presumes to be R's belief towards q and what S is communicating: $Plaus^1_{S,R}(q) = 1 - |B_S B_R P(q) - B_S P(q)|$, and another used to represent the *global plausibility* as the overall compatibility of the difference between the beliefs created by knowing the new fact and the previous beliefs: $Plaus^2_{S,R}(q) = 1 - \frac{1}{w}(\sum_{l=1,\dots w} |B_S B_R P(q_l) - B_S P(q_l)|)$;

(c) *safety* of the communication act, measured by the risk of S getting caught in the attempt of deceiving R. To calculate the risk, various factors must enter into account: (1) how reasonable, for R, is that S actually believes in what it says; (2) what are the chances of R understanding that, by believing in the fact, S will get some advantage from it;

(d) *reliability* to R regarding the source of information that S evoked with the purpose of justifying its statement. The credibility of such a fact being mentioned by the entity stated by S is calculated as: $Cred(q, IS) = P(q|Say(IS, q))$, where the reliability of the information source IS is defined as: $Rel(IS, q) = P(Say(IS, q)|q)$. S is able to provide multiple information source as a way to strongly support his communication, or just to shake R's ideas by mixing both reliable and unreliable information sources.

After having computed the four factors for each candidate, it will select one or more *good* candidates based on a strategy. That strategy may be formalized as a continuous selection of subsets of candidates that have their *impact*, *safety*, *plausibility* above a certain threshold, each threshold defined for each factor, then ordered multiple times from best to worse on each factor and finally, if an information source exists so that the credibility of the candidate may increase.

This tool is especially good for a User to try different strategies, defined by the thresholds and input information given to the System, and see how different, or similar, will the System try to deceive a target, when compared to humans. While being strongly related to our work in terms of making a comparison between a human deceiving and an agent doing so as well, it is currently not being used by our work, since our main objective is to use deception in a social environment, instead of in an artificial system.

3.2 GOLEM

GOLEM is a system, developed by Castelfranchi, Falcone and de Rosis[12], based on a multi-agent blocks world with the objective of studying both the interactions and attitudes between two agents with different social attitudes and personalities, when delegating and adopting tasks from each other, and the reasoning behind those delegations and adoptions, such as not being able to do a specific task, or having a lazy personality.

GOLEM's multi-agent world only contains two agents, and while it looks rather limited by having just the minimum of interacting agents to make it a multi-agent world, it gains immensely by allowing the two agents to be completely different, both by having different goals, that can me competing, and by having different kinds of personalities.

The goals in GOLEM are quite simple, since it is on a blocks world, and are limited to making some type of construction with the limited resources of the world. As initially formulated by the authors, there exist four big blocks and four small blocks in the world and competing goals (e.g. building a bell-tower, four overlapping big blocks with a small block on top of them, and building a small-tower, four overlapping small blocks). Each agent has a different goal and tries to achieve it.

Personalities of the agents is what makes GOLEM such a rich environment, as they are much more diverse. Both agents have personalities for when delegating tasks, from "Lazy Agent", that will always delegate tasks if the other agent is capable of performing them, to "Never-Delegating" which only achieves tasks if they can be performed by him; and when adopting tasks, from "Hyper-Cooperative" that always tries to help, to "Non-Helper" which never helps. The adopting personalities are a little more diversified than the delegating personalities, allowing a agent, for example, to hypothesize the higher order goal of the delegating agent and help him towards that, "Overhelper" personality.

Each agent is also limited in terms of abilities. What this means is that each agent may only be capable of performing a limited set of actions on the domain state. An example would be that agent A can only move small blocks and agent B is only able to move big blocks. If they have goals which include moving both types of blocks, they will have to cooperate to achieve them, which can lead to a conflict of interest if the goals are competing.

As mentioned before, there may exist two type of conflicts in GOLEM, (1) conflict between the plans and goals on the domain that the agents have and (2) conflict between social goals, such as delegating and refusing tasks. These conflicts is what triggers the various types of deceptions in GOLEM.

GOLEM acts as a game where initially it starts by selecting a "world", which is composed by the definition of each agent's mental state (goals plus personalities), the setting of the initial domain state and the decision of which agent moves first. After that, both agents will introduce themselves by stating their personalities and abilities, which they can lie about, or simply give partially incorrect or imprecise descriptions. Finally, both agents build an image of their partner's mental state. The running of this simulation is then played in turns, where each turn an agent can perform some action on the domain or not, and perform a "communicative act" according to a defined protocol.

The most interesting kind of deception currently present in GOLEM is a motive-based deception that appears when an agent is in a situation that if the other agent knows that p, it will not adopt the request made by him, where p may be (1) the capabilities of the agent, or (2) his personality, or even (3) his goals and plans. Starting by (1) deception about capabilities, a good example to represent this situation is having agent A, with personality "lazy", trying to delegate some action that he can do to agent B, with personality "supplier", which will only help if the delegating agent can not perform such action. For A to be successful in delegating the task, he must hide his ability to perform the action, so that B believes that A is incapable of doing it and consequently helping him by accepting the delegation and performing it. A can achieve this by saying explicitly that he is not able to perform the action or try some more complex strategies, such as inducing B that he is "delegating-by-necessity", only delegates if not able to perform the action, and delegate said action. Another of the content-based cases is (2) deception about personality, in which an agent can make an action that is not congruent with his personality, making personalities not rigid, but tendencies, so that the other agent may presume wrong personalities and thus allowing the initial agent to take advantage of this in the same way represented on the previous example. The last case is (3) deception about goals and plans, where an agent may declare that his goals is different than the real one and perform actions congruent with it, so that the other agent won't know that, for example, their goals are competing, and thus making him help, when he would rather not.

In his current form, GOLEM does not allow strategic lies, i.e. deception through speech acts, as agents do not reason about the reasoning of the other agent in response to their speech acts, thus not allowing to use such speech acts or protocols for deception.

3.3 Deception Planner

Deception Planner, developed by David Christian for his master thesis[13], is an implementation of a model of strategic deception, i.e. it attempts to deceive in order to achieve or enable some final goal.

To better explain how the deception planner works, consider the following example:

Bob is looking to buy a new lcd television for his home. When he finally gets to the store and starts looking for the ideal one, he is approached by Carl. Carl knows, by looking at the price tags of the TVs that Bob is considering, that Bob has a generous amount of money and so, because of that, Carl wants to lead Bob to some dark alley and rob him.

Which statements, that may include lies, can Carl present to Bob as to provide himself a good opportunity to achieve his goal?

The problem represented in the above example is the type of problem that the deception planner tries to solve. To do so, it first needs some input of the problem, that is, a set of ulterior goals (in this example, the goals that Carl wants Bob to achieve), the current world state and the model of target agent, which include observation rules (Carl's model of Bob). In the end, it will provide, if possible, a set of facts and a set with negation of facts (linking with the example, "The televisions from that store usually brake fast" and "Carl knows a store with better televisions and some good discounts that is near the one they are").

The deception planner is a modified version of the LPG (Local Search for Planning Graphs) planner, for the reason that it is a local search planner, the heuristic is relatively informed and is able to do plan repair. Like the LPG planner, it treats the planning problem as an exploration of a search space, where each node is a partial or complete plan. The plan repair done in the deception planner is almost identical to the generic one, it will remove some of the actions with most cost from the plan, so that it select new actions with less cost.

To be able to find a plan that fulfills the needed conditions, the deception planner uses a complex heuristic:

$$\begin{aligned} H_{DP}(P,a) = (1 &+ \lambda_m^a * CostToFixMutexes(P,a) \\ &+ \lambda_p^a * CostToFixOpen(P,a)) \\ &+ v_{ug} CostToAchieveUltGoals(p,a) \\ &+ v_{lie} CostToFixLies(P,a) \ . \end{aligned}$$

While both the CostToFixMutexes and the CostToFixOpen are used in the generic LPG planner, the deception planner adds the CostToAchieveUltGoals and CostToFixLies. Starting with the part of the heuristic that is generic, both λ_m^a and λ_p^a are stochastic weights that are calculated dynamically and take into account how hard is it to solve a particular precondition taking into account the planning problem involved. The CostToFixMutexes is self-explanatory as it is the cost of reasserting a condition p due to a mutex (mutual exclusive actions) and the CostToFixOpen is an estimation of the cost of achieving all open preconditions of the new plan, which also takes into consideration the achievement of the deceiver's ulterior goals.

CostToAchieveUltGoals (CUG) is an estimation of how many steps are needed from the current plan to a plan that achieves all ulterior goals. It takes into consideration that an ulterior goal can only be achieved if it is relevant to achieving the planning goals and that all ulterior goals must be achievable. If some ulterior goal is not achievable in some plan, that plan will have an infinite CUG. It also takes into consideration the possibility of time points where all ulterior goals are asserted at the same time, giving to those plans a lower CUG.

The final part of the heuristic is the CostToFixLies (CFL), which is 0 if the planner has told no lies that can be observable, through the target agent's observation rules, before the achievement of the ulterior goals or an estimate of how many steps have to be added so that the lies are kept from being observed.

The last part of the deception planner is the negation of competing plans. A competing plan is a plan that given the information provided to the target agent, with the objective of him creating a plan that achieves the ulterior goals, there is still a better plan that doesn't include the achievement of the ulterior goals. To negate such a plan, the deception planner tries to find lies about beliefs that the target agent may have, so that those competing plans are no longer viable. If a competing plan can't be countered, the planner will have to find another candidate plan.

While being difficult to implement directly into our work, the deception planner is a tool that could increase our agent's social capabilities so that it could profit more from playing deceptively. As so, it is something we are considering to implement in the future.

3.4 Study about Deception in Robots by Wagner and Arkin

Wagner and Arkin, with the use of interdependence theory and game theory as a way to understand deception from a robotic point of view, have developed some algorithms that try to solve two types of problems: (1) when to deceive and (2) how to deceive[14].

To understand how the algorithms work, first we need to take into consideration how represented the interactions between the deceiver, the agent that deceives following the deception definition of Bond and Robin, "a false communication that tends to benefit the communicator"[15], and the mark, agent that may be deceived and always tries to maximize his outcome. To represent the interactions, the authors used outcome matrices that consist of (1) a finite set N of individuals that are interacting, (2) a set of actions A^i that cannot be empty for each of the $i \in n$ individuals and finally, (3) the utility generated from each combination of actions for each individual.

Another thing that must be defined is the mental model of both the deceptive agent and the mark. To produce such models, it is used Wagner's interact-and-update algorithm, which can populate outcome matrices, but also build mental models that consist of three different types of information: (1) a set of features that allow the recognition of the individual (hair color, age, height), (2) an action model containing a set of actions that are available to the individual, (3) an utility function which has information of the outcome for each combination of actions.

Knowing when to deceive is an important topic when talking about deception, since not all situations provide advantages to a potential deceiver (e.g. teammates cooperating). To find which situations provide the need for deception, the authors have developed an algorithm that takes as input both the deceptive agent model and the mark's model and gives, as output, if deception should be used or not. The algorithm first uses Wagner's interact-and-update algorithm to produce the true matrix, which is the outcome matrix that represents the actual outcome obtained by both individuals gather that no false communication as occurred, and then use the interdependence space algorithm to obtain the values for interdependence and conflict. If those values are within some interval, then the situation can be said to warrant deception.

The other algorithm developed is for the agent to act deceptively and consists of various steps. The first step is to check if the situation warrants deception by using the algorithm described before. All following steps depend on if deception is warranted. In the next step, the algorithm will create an induced matrix, which represents the situation that the mark will believe is true after getting the information from the false communication, after that, it will select the best false communication from a set of false communications known to the deceptive agent, and finally, it will interact with the mark by producing the false communication and selecting the action that maximizes his outcome.

By experimenting with these algorithms, through the use of different conditions, Wagner and Arkin obtained results that lead to three main conclusions: (1) the outcome acquired by an agent that is able to recognize when to deceive, will be greater than that of an agent that does not; (2) most of the situations do not warrant deception; and (3) by only partially deceiving another agent, the amount of outcome obtained will be relatively lesser than if fully deceived. These lead to the main conclusion that an agent that knows when and how to deceive will acquire a greater amount of outcome, resulting in an advantage, when compared to an agent that does not, which in turn, is exactly what we assume in our work.

3.5 Study about Deception in Agents by Ward and Hexmoor

Ward and Hexmoor have implemented a test-bed with agents that may use deception in order to collaborate and communicate. In their implementation, agents have (1) different personalities, ranging from benevolent to selfish, (2) various trustworthiness levels, (3) a certain reactiveness to deception depending on the agent, (4) been designed in BDI paradigm[16] and (5) implemented a possible-world semantics[17].

The simulated environment consists of a two dimensional grid where each square contains blocks in a stack. The goal of the agents is to push block such that four our more adjacent squares have the same number of blocks and with that, gain ownership of those squares for its own team. Agents are also restricted in terms of what blocks they can push, they can only push blocks that are in a square which has a lower elevation than the one they are. They can also plant solar panels on the squares their team has ownership and can then harvest them when they have charged.

The agents' intentions consist of a plan that is selected through the use of depth first search[18] in a state space. The utility function for a given state is:

$$\begin{aligned} UTIL(S) = \frac{1}{4}[W_1 * UTILmove(S) + W_2 * UTILdig(S) \\ + W_3 * UTILplant(S) \\ + W_4 * UTILharvest(S)] \; . \end{aligned}$$

Where each W_i is a weight between 0 and 1.0, depending on the agent's strategy.

Since the reward obtained through working on individual plans is shared between the group, some agents may use deception with the objective of working less. For that to be possible, the authors have modeled deception into three different devices, (1) deceit about goals, (2) deception about beliefs, and (3) passive deception or withholding of information.

For an agent B to communicate a deception device d to agent A, the value of the deception utility function $Deceive(d, B, A)$ must be ≥ 0.5. The deception utility function is defined as:

$$\begin{aligned} Deceive(d, B, A) = (1 - trustworthiness(B)) * efficacy(d, Bel(B, Bel(A))) \\ * plausibility(d, Bel(B, Bel(A))) \; . \end{aligned}$$

It is needed to note that $Bel(B, Bel(A))$ show B's belief towards A's beliefs and that each function used by the deception utility function returns a value that between 0.0 and 1.0.

By experimenting with two different scenarios, the authors concluded that deceit used inside a group can bring positive effects, such has tricking an inactive agent into traveling to unleveled areas, making him more productive, but also negative effects, some agents got distracted from completing their almost completed tasks.

3.6 Study about Robots Deceiving Humans by Terada and Ito

Terada and Ito have developed an experiment to prove if robots can indeed deceive humans. While they do not provide a tool of some sort towards deception, like the majority of related works, their work has value when we take into consideration the true meaning of deception and try to apply it to a robot[19].

To prove that robots can deceive humans, the authors focus on deception as a cue for deception attribution and base their experiment on proving that a human can treat a robot as an intentional entity. They take the design stance so that, if a robot is treated as a designed entity by the human, an unexpected behavior is considered as an error and thus the human will feel deceived. This is a strong indicator that the robot is being treated as an intentional entity.

The experiment is based on a slightly modified version of "Daruma-san go Koranda", which is a Japanese game similar to the English game Red Light/Green Light.

The game consists of two phases: (1) the robot is facing the wall and says “Daruma-san go Koranda”, while the person is walking towards the robot, and then turns around, and (2) tries to detect if the person is moving or not and turns to face the wall again. These phases are repeated until either the robot detects the person moving or the person is able to touch a button on the robot’s head surface.

To be able to reach some conclusions, they used two different experimental conditions, (1) the deception condition, where the robot adopted a “slow normal behavior” which consisted on slow chanting of the syllables and a big turn around time, and then, when the person could reach the robot in the next turn, it would adopt a “fast deception behavior”, accelerating the chanting of words and turn around; and (2) the control condition, where it always produces the “slow normal behavior”. It’s important to state that, in the deception condition, when adopting the deception behavior, the robot will always signalize that he caught the person moving, even if that is not true.

The conclusions that can be taken from this experiment come from the answer to a questionnaire answered by the participants, in which the deception condition group felt more outwitted than the control condition group, while the control decision group felt that the robot was more predictable than what the deception condition group felt.

By taking into account that the feeling of being deceived is a strong indicator of being taken as an intention entity, it can be concluded that the humans in the deception condition group took that stance.

3.7 Mindreading Agents

João Dias et al. developed a model for a mindreading agent that supports N levels of Theory of Mind and is capable of carrying out deceptive behaviors[20].

Their agent’s Theory of Mind is based on the Mindreading model of Baron-Cohen[21] and follows the ST (Simulation-Theory) of Meyer et al.[22], which claims that one should represent others the same way they would simulate themselves in the same situation.

The Theory of Mind is composed of three different components: (1) Theory of Mind Mechanism, that represents and stores others’ mental states, which consist on Models describing the beliefs of a specific agent that the agent knows; (2) Eye Direction Detector (EDD) that determines who sees what; and (3) Shared Attention Model (SAM) which constructs higher level relations between entities. Each Model of Other is a simplification of the agent’s own model, so that it can be updated using the same mechanisms as it’s own.

In order to determine how and when the Models of Other should be updated, given a perception, EDD and SAM provide two mechanisms. One of the mechanisms checks if a target agent is within a defined radius of the perception received, and if so, it will also apply that perception to the Model of Other that represents that agent’s beliefs. The second mechanism uses some domain specific rules that provide restrictions over the perceptual mechanisms of the agents.

To be able to use their information about the world and other, and to behave in a deceptive manner, the agent’s model includes a Deliberation and Means-End reasoning component that uses a continuous planner. This allows the creation and execution of plans of actions to achieve the agent’s desired goal states.

Finally, they built an experiment to test if agents with two levels of ToM were more effective than agents with only one level of ToM, when playing a deceptive game called Werewolf. The results showed that the 2-level ToM agent version won more games than the 1-level ToM version, proving that having it is advantageous for an agent to possess two levels of ToM when playing a deceptive game as opposed to only one level.

3.8 Summary

To compare the various works studied, we produced a table that shows the differences between each of them, while still providing significant information towards our work.

Table 3.1 shows the different types of deception allowed in each of the works, being those the ones mentioned before, the levels of theory of mind available, if they use the target's mental model when trying to deceive and finally, if they were implemented on a virtual or robotic/physical agent.

Related Work	Deception Allowed	Levels of ToM	Uses target's mental model	Type of agent
MOUTH-OF-TRUTH[10]	Intentional, Active, Explicit, Strategic	One	Yes	Virtual
GOLEM[12]	Intentional, Accidental, Active, Passive, Explicit, Implicit,	One	Yes	Virtual
Deception Planner[13]	Intentional, Active, Explicit	One	Yes	Virtual
Deception in Robots[14]	Intentional, Active, Explicit, Implicit	Zero	No	Robotic
Towards Deception in Agents[17]	Intentional, Active, Passive, Explicit	Zero	No	Virtual
Can a Robot Deceive Humans[19]	Intentional, Active, Implicit	Zero	No	Robotic
Mindreading Agents[20]	Intentional, Active, Exlicit	N-levels	Yes	Virtual

Table 3.1: Comparison of the different architectures from the related work

MOUTH-OF-TRUTH takes deception as a communicative act that is based on the calculated value of the efficacy, plausibility, safety and reliability of a certain deception medium, while taking into account the uncertainty of a deception target given its mental model.

While this kind of approach would be meaningful to apply to our architecture, it lacks a stronger modulation of the target's mental model and only allows deception through communicative acts as opposed to our need for deception through actions.

While GOLEM allows more kinds of deception than any of the other related works, it still does not allow deception through speech acts, which may have value if implemented in our work. Nevertheless, GOLEM still provides a relatively better mental model modulation that may be similar to what we need, and it also provides an overall good basis for our modulation of deception.

The Deception Planner is one of the best tools, from the studied work, at doing what it does, which is selecting a set of statements to influence a target's mind in a way that its plan of actions contains actions that are in the deceiver's best interests.

In our case, we want a specific opponent to perform an action from a specific set of actions, but it is very difficult to model the target's mind in the way required by the deception planner, as the environment is in constant change.

The approach taken in Deception in Robots is based on both game theory and interdependence theory, and while it may be a very generic approach, it may provide significant value to our work. Since our needs towards which action we should take can be correctly modelled as an outcome matrix, by adding deception to it and trying to deduce the induced matrix, the outcome matrix that the deception target will produce, we can come up with a good model for our deception needs.

It should also be mentioned that knowing in which situation we should deceive is crucial to our work. Deception in Robots provides an algorithm for this.

Towards Deception in Agents, while being a simple test-bed for agents that can use deception, it provides important insight towards deception.

The way they model deception, not only takes into account the efficacy and plausibility of something being said, it also considers how trustworthy is the agent for the target agent, which is something that is not as explicit in any other of the related works, as it is here. Trustworthiness is an important factor for our work as we deal directly with humans.

While not being a tool for deception, Can a Robot Deceive Humans provides something that is needed for our work, which is to actually know if a human would feel being deceived by a robot. Their approach is quite simple and makes some bold assumptions. Nevertheless, it provides a strong indication that robots can indeed deceive humans, which is a good assumption to have in our work.

The work done by João Dias et al. is strongly connected to ours and provides us with a solid prove, given this thesis subject, that having two levels of theory of mind gives an advantage to an agent capable of deceiving, over only one level of theory of mind. Their model for the mindreading agent, or a slight variation towards our goals, should be used in this thesis as a way to model the humans beliefs.

Chapter 4

Problem and Solution Definition

In this chapter we start by giving a detailed description of the problems that are inherent to the problem that this work solves, and provide an overview of the most important aspects to take into consideration for the solution that is presented after. We then proceed to describe the various components of the solution devised to solve the problem.

4.1 Problem

This section will describe the various problems that are inherent to this work's final objective and is divided into categories of problems that have to be dealt with to produce a system that allows for the testing of our work's hypothesis.

4.1.1 Controllable lying

Since the agent does not possess the ability of a human to "feel" the possibility to lie without being detected nor is it aware of the differences between lying and not lying in terms of behavioral aspects, it is not able to reproduce the changes normally seen on humans.

It has to get a way to simulate these changes as a way to make humans try to doubt it.

The agent needs to be aware of the drawbacks of lying, since without it, it would perform a variety of actions that would make sense in terms of the value they would have in some particular moment and not because those are the actions that he is able to do. By doing so, the agent could be constantly lying and by doing that, it would keep losing indefinitely, as it is much easier to catch when the agent is lying.

Making the agent capable of all the things mentioned before would be extremely complex if the objective were to make it as generalized as possible, since most of them are not even fully comprehended on how they work with humans.

4.1.2 Representation in the real world

To be able to simulate a human player in terms of behaviour and to give the feeling that the agent is not completely virtual (give feeling of familiarity to the humans towards the agent), we need some kind of physical entity to represent our virtual agent.

Having such entity would allow the virtual agent to interact with the physical world, making it capable of creating a more solid relationship with its fellow human opponents.

Another problem that we have to take into consideration is the fact that the agent will have to be linked to the physical entity in order to ”act” through it, which in itself is a problem of communication between our agent and the physical entity that will represent it.

The physical entity would need to be positioned in a way that a human would be, in regard to the place where the game will run. This puts a restriction in both the physical entity used to represent our agent and the place where the game will be played.

We also need to consider that our agent has to be capable of using the physical entity capabilities in order to take advantage of said interactions. This puts a requirement on the physical entity that we will use of being expressive enough to be used by our agent.

4.1.3 Human and agent game interfaces

To be able to test an agent that is capable of deception for its own advantage while playing the board game COUP, we have to have an environment where both the virtual agent and the physical human can play the game simultaneously.

The problem is with the interface. If we have a completely physical game, like the traditional board games, the human players will have the same experience that they are used to, but the virtual agent will not only have a different experience, but he will not even be able to play the game, which will make this work impossible to achieve. Having the game completely virtual, where the game is running as a software and the interface for the players to play is a monitor and a mouse, just like a traditional computer game, will make the human player experience completely inferior when compared with the physical version of the game. The human players will not be able to speak with each other face to face, nor have the ability to physically touch the components of the game (coins, cards).

To make this work the best possible, an equilibrium of the two has to exist. Something that feels like the traditional physical game and that provides some type of interface for the virtual agent, so that the agent can perceive what is happening in the game and at the same time, perform actions on it.

4.1.4 Synchronicity between actions and interactions

To be able to effectively demonstrate some similarities to the way that humans behave, so that it is possible, as mentioned before, to deceive them, the virtual agent and the physical entity that represents the agent have to be linked so that when the virtual agents decides some action towards the game, the physical entity will represent how the agent would behave while doing that action. This implies that the agent is capable of sending commands to the physical entity, or that the physical entity is capable of recognizing the actions made by the agent and acted in conformity to that.

To specify a little more, the actions and the physical behaviour may be needed to run at specifically different times. An example of this would be when the agent wants to deceive its human opponents by making them believe that it does not have the card required to do the action that it is doing, and to do so, it tries to look unsure by representing it through behaviour using the physical entity and after some time playing the action. This requirement furthers the need to make the agent in control of the physical entity, so that the human players truly feel that the agent that they are playing is the physical entity and so that the agent have more control its capacity to deceive its opponents.

4.1.5 Lie detection capabilities

Being an artificial entity, our agent will not have the same capabilities of a human in terms of lie detection. Humans subconsciously detect out of order patterns on other humans. These can be changes in facial expressions, changes in vocal intensity, stuttering, changes in body expressions. A virtual agent is not capable of detecting these. It would require some good cameras capable of detecting those changes and software that could determine if those expressions were coming out of the ordinary. This is not possible with the equipment we have available to us.

For the voice, a powerful voice detecting and analysis software would be needed. Still, with all this technology, it would still be a complex problem to detect if a human would be lying or not, since even nowadays there are humans capable of bypassing all machines made to detect lying.

This part is relevant so the agent knows when to challenge.

4.1.6 Play at the same level of humans

Finally and mainly, the most significant problem that we have to solve is to make our agent capable of playing at the same level of humans. To do so would imply that, not only is our agent capable of detecting lies at the same level of humans, but also make decisions towards what actions it should take, including actions that it should not be able to do considering the character cards it has, or in other words, perform actions that require lying. This problem is not as linear as we are putting it, since by having a very strong decision making algorithm, our agent can have a weaker lie detection system and still balance things out in a way that it is has capable of winning the game as humans are.

Since this is the main objective of this work, this problem should be the one to take the most attention and time spent.

4.2 Solution

In this section we describe the overall solution to the problem of the Coup Player. We not only mention all the components relevant to the solution, but also describe some of them in more depth in a way to show how they were implemented in this work.

4.2.1 Overall System

The system that we devised to solve the problem is divided into three big parts:

Digital Tabletop

The first thing we have to take into consideration so that our work is possible is the fact that a virtual agent will not use the same interfaces as a human player would use. Not only that, but the human players should be able to use the physical pieces of game, such as cards and coins, as pieces in our merged environment, so that their experience is the same as with the traditional board game and with that the results from our experiences will not be affected by that. To do so, we need something that is capable to detect physical objects and distinguish them, and at the same time provide that information to our virtual agent. The actions used in the game, must be both possible through physical touching some options, for the humans, and through a virtual interface so that the virtual agent can interact with the game.

To meet all these requirements, we came to the conclusion that a digital tabletop would be our best bet, since it is capable of running the game as a software, while displaying it for the humans through a touchable digital display, much like the one used for smartphones and tablets, but the size of a table, and displaying it through a virtual interface for the virtual agent to receive messages of what is happening to the game state and send commands to execute actions unto the game. By having a physical touchable display and being capable of detecting and distinguishing objects that are on top of it, the digital tabletop provides a seemingly traditional board game experience for the human players.

However, the problem of compatibility between the solution for the different interfaces and the software that is able to run on it still remains. Since most of the digital tabletops commercially available run the same operating systems as the ones that personal computers run, the game can be implemented in Unity[1], which not only is a software platform used by a great community of game developers, but also has support for a very big selection of platforms, making our implementation of the game testable on multiple environments. It also provides the capacity to use both interfaces that we need, the one for the virtual agent and the one for the human players, putting it as one of our main building blocks for our work.

Robot

As mentioned in the problem definition section, we need a physical entity that can represent the virtual agent in the real world, so that the agent can interact with the human opponents. To do so, some capabilities are required from that physical entity, such as the ability to use facial expressions, since those are the expressions most reliable read by humans, to communicate its possible internal emotional state, a way to demonstrate what the agent is effectively focusing upon, much like how humans show their focus through the directionality of their eye sight, or where they are looking at. The physical entity would also need equipment to recognize whom it is actually looking at, equipment to be able to express the virtual agent thoughts through voice and a way to listen to key utterances from the human players.

The solution we came up for the problem of the physical entity is to use a robot with all those capabilities. The robot has the form of a human head, which is capable of expressing a complete range of human emotional and intensity of such emotions and mobile in the same way a human head his, so that it can change the directionality of its focus, making the humans recognize to whom the agent is looking at or directing its interaction to. To meet the requirement of recognizing the target of interaction, the robot has a camera attached to itself, so that it can map the

[1]`http://unity3d.com`

presence of each human in terms of directionality of view and then focus a specific human based on its objective of interaction. Another of the problems mentioned before was the need to make the agent able to express itself through voice output, for that, the robot must have a speaker also attached to it, so that directionality of interaction still exists. Finally and as a way to make the virtual agent capable of detecting the human voice, the robot needs to be connected to microphones, which are best used if attached to each human player, as the voice recognition and distinction will be much more easier to do and complex sound analysis software is not required.

The robot must also run a software to as to give an API for the agents to call and with that perform actions unto the robot. Those actions include all of the previous mentioned necessities, such as performing a facial expression, saying something, or simply moving the head to some specific direction.

With all this functions available to the virtual agent to express itself in the real world and interact with the humans, it is now provided with the much needed elements to achieve a deceptive behaviour and successfully deceive its human opponents. The only real challenge is how to coordinate all these output mechanisms to act in a way that humans do consider the actions of the virtual agent to be different from what they really are, thus making themselves be deceived by our virtual agent.

4.2.2 Agent's Architecture

The last part is the most important one for our work, since it is the very core of our virtual agent. The agent's architecture part is responsible for all of the logic and processes that happen with the our agent, which include the agent-game communication and the agent-robot communication, this means that the architecture takes into consideration both the robot and the digital table interfaces to link them to the virtual agent.

The agent's architecture will receive data from the robot component, such as the voice of the human opponents, and will need to be able to process it so that it can discover some key utterances and from there came to conclusions regarding the current cards of that player. It will also be able to receive the actions that are being played in the digital tabletop, so that it can update its own version of the game state. Finally and being it the most important step, it will, from all the information provided, compute the best approaches for the agent to take.

All this leads to an architecture composed of six main components, each one with different objectives to them, but all used to produce the best actions for the agent to take, be it a game action, an interaction with one or more of the human players, or even both of them at the same time. The components devised to solve the problem of action and interaction selection are:

- **Perception Receiver** It's the component responsible for receiving perceptions, both from the game and from the robot, for example when an interaction as finished.

- **Memory** Contains both the memory of the agent, which holds all the perceptions received, and its theory of mind, which takes into consideration the agent's memory and tries to derive what the opponents are thinking, including the probability of having each card and the probability of making a certain action.

- **Theory of Mind** Takes into consideration the agent's memory and tries to derive what the opponents are thinking, including the probability of having each card and the probability of making a certain action.

- **Decision Making Algorithm** Algorithm that uses the theory of mind component and a modification of the regret minimization algorithm to produce the next action for the agent to play
- **Action/Interaction Producer** This component is responsible for the decision of how the robot used to represent the agent's physical presence should act, it takes into consideration the action that the agent is going to make. It is also the connection that outputs messages to both the robot and the digital table so that actions can be played in both of them.

This agent's architecture is just an overview of the components used on it, since in Chapter 6, we will present not only a more complete definition of what each component does, but also how we implemented each one of them, so that the agent is able to play the game against humans.

Chapter 5

The Decision Making Algorithm

In this chapter we describe how the decision making algorithm works in detail. The decision making algorithm takes as input the current state of the game and the agent's theory of mind module to produce the next action for the agent to play. We first start to describe the standard algorithm on which the decision making algorithm is based on, which is the regret minimization algorithm in games with incomplete information. We then describe the necessary modifications to this algorithm in order to take into consideration some aspects that are game-dependent.

5.1 Regret Minimization

The decision making algorithm used in this work is the regret minimization algorithm that minimizes regret through the minimization of counterfactual regret [23], with a different utility function. The regret minimization algorithm not only supports games with incomplete information, which is the case of COUP, but also works fairly well in extensive games.

This algorithm was chosen as the starting point for its way of dealing with extensive games with incomplete information, as mentioned before, and for its amazing success in Poker, which is a similar game to COUP, in the way that a player, to be successful, will most likely have to lie.

We will start to present the definition of a finite extensive game with incomplete information and then describe how the Regret Minimization algorithm takes advantage of the Counterfactual regret in order to produce actions that minimize regret.

5.1.1 Extensive games with incomplete information

An extensive game is a game that is able to provide a general, but compact model for multiagent interaction, which in itself is capable, most of the times, of representing interactions with a sequential nature.

Giving a more general definition, an extensive game is composed by a game tree just like in any perfect information game, with every non-terminal game state being associated to a player chosen action, while each of the terminal states has an associated payoff for each of the playing participants. The greatest difference between these two is the fact that there is a constraint on the game states, where some of them can not be distinguished by the controlling player. By having a group of states that ca not be distinguished, we attribute to this set the name of information set.

In the game of Coup, as an example, whenever the first player starts to make the first action in the game, no matter what the cards his opponents have, if he has the same cards as in another game, from his point of view, the information set is exactly the same.

We now present a formal definition and notation for what a finite extensive game with imperfect information is.

Definition [24] *a finite extensive game with imperfect information has the following components:*

- *A finite set N of* ***players****.*
- *A finite set H of sequences, the possible* ***histories*** *of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the terminal histories (those which are not a prefix of any other sequences) $A(h) = \{a : (h, a) \in H\}$ are the actions available after a nonterminal history $h \in H$.*
- *A function P that assigns to each nonterminal history (each member of $H \setminus Z$) a member of $N \cup \{c\}$. P is the* ***player function****. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$ then chance determines the action taken after history h.*
- *A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.*
- *For each player $i \in N$ a partition $\mathcal{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$ for any $h \in I_i$. $\mathcal{I}_i$ is the* ***information partition*** *of player i; a set $I_i \in \mathcal{I}_i$ is an* ***information set*** *of player i.*
- *For each player $i \in N$ a utility function u_i from the terminal states Z to the reals* ***R****. If $N = \{1, 2\}$ and $u_1 = -u2$, it is a* ***zero-sum extensive game****. Define $\Delta_{u,i} = max_z u_i(z) - min_z u_i(z)$ to be the range of utilities to player i.*

Taking that definition in mind we can also define what a strategy is. A strategy of player i σ_i in what we have defined as an extensive game, is a function that receives an information set and attributes to it an action from $A(I_i)$. A strategy profile σ consists in the aggregation of a strategy of each player, σ_1, σ_2,..., with σ_{-i} referring to all the strategies in σ except σ_i.

Let $\pi^\sigma(h)$ be the probability of history h happening by all players choosing actions as stated by σ. $\pi^\sigma = \Pi_{i\in N\cup\{c\}}\pi_i^\sigma(h)$ is then decomposable into every player's contribution for this probability. Consequently, $\pi_i^\sigma(h)$ is the exact probability of player i, for all histories h' that are a proper prefix of h, taking the corresponding action in h when player i is playing as stated by σ. For $I \subseteq H$, define $\pi^\sigma(I) = \sum_{h\in I}\pi^\sigma(h)$ as being the probability that a particular information set give σ is reached, with $\pi_i^\sigma(I)$ and $\pi_{-i}^\sigma(I)$ being defined in a similar way.

The value of a strategy profile for a player i is the expected payoff of the terminal node that is reached as the result, $u_i(\sigma) = \sum_{h\in Z} u_i(h)\pi^\sigma(h)$.

5.1.2 Regret Minimization

To define the concept of regret, we need to consider playing an extensive game on a repeated way. Letting σ_i^t be the strategy used by player i on round t. We can calculate the average overall regret at time T of a player i with:

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \sum_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t)) \tag{5.1}$$

Where σ_i^* is the strategy that produces the most utility, when considering that all other players play according to strategy σ_{-i}^t.

By calculating the average regret, we know how much utility the player lost for using strategy σ_i^t instead of the optimal strategy σ_i^*. Knowing this and using an algorithm that selects σ_i^t for player i that minimizes regret, or in other words, makes player i's average overall regret go to zero, as t goes to infinity (regardless of the sequence σ_{-i}^t), we can conclude that the algorithm will tend to select the optimal strategy σ_i^*, as t tends to infinity.

5.1.3 Counterfactual Regret

The fundamental idea of [23] approach is to decompose the overall regret into individual regret terms that can be added, which will then make them able to be minimized independently. They even introduce a new regret concept for extensive game to which they called counterfactual regret and is defined on an individual information set.

Starting by considering that one singular information set $I \in \mathcal{I}_i$ and player i's choices that are made based in that particular information set. They then define $u_i(\sigma, h)$ to be the expected utility given that the history h is reached and then all players play using strategy σ. Counterfactual utility $u_i(\sigma, I)$ is the expected utility given that information set I is reached, while all players are using the strategy σ with the exception of player i which will play to reach I. By taking into account that $\pi^\sigma(h, h')$ is the probability of reaching history h' starting from history h, then:

$$u_i(\sigma, I) = \frac{\sum_{h \in I, h' \in Z} \pi_{-i}^\sigma(h) \pi^\sigma(h, h') u_i(h')}{\pi_{-i}^\sigma(I)} \tag{5.2}$$

By considering that for all $a \in A(I)$, $\sigma|_{I \to a}$ is the strategy profile that is completely the same as σ, with the sole difference that when in information set I, player i always chooses the action a, Zinkevich et al. define immediate counterfactual regret as being:

$$R_{i,imm}^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I)) \tag{5.3}$$

By intuition, this is the player's regret from the decisions it would take by continuing to employ its current strategy, instead of using an action a that would maximize its counterfactual utility at round t, with an additional weighting term that considers the probability of other players playing to reach information set I. Since only the part where regret is positive matters, let $R_{i,imm}^{T,+}(I) = max(R_{i,imm}^T(I), 0)$ be defined as the positive part of immediate counterfactual regret.

With all this, Zinkevich et al. came to the most important key result. That $R_i^T \leq \sum_{I \in \mathcal{I}_i} R_{i,imm}^{T,+}(I)$.

The proof is presented in the appendix of [23]. And so we actually know that by minimizing immediate counterfactual regret, we can minimize the overall regret.

The main feature of immediate counterfactual regret is that, by having control over $\sigma_i(I)$, we can minimize it. For this, Blackwell's algorithm for approachability is perfect to minimize the regret in an independent way for each information set. In particular, Zinkevich et al. maintain that for all $I \in \mathcal{I}_i$, for all $a \in A(I)$:

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I)) \tag{5.4}$$

Define $R_i^{T+1}(I)(a) = \max(R_i^T(I,a), 0)$, then the strategy for time $T+1$ is:

$$\sigma_i^{T+1}(I)(a) = \begin{cases} \frac{R_i^{t,+}(I,a)}{\sum_{a\in A(I)} R_i^{t,+}(I,a)} & \text{if } \sum_{a\in A(I)} R_i^{t,+}(I,a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \tag{5.5}$$

In other words, actions are selected in proportion to the amount of positive counterfactual regret for not playing that action. If there is no action that produces a positive counterfactual regret, then the action is selected randomly.

5.2 The Decision Making Algorithm

In this section we will present the decision making algorithm that we created based on the algorithm mentioned in the section above.

Starting with why we should use counterfactual regret minimization as our leading variable to decide what actions to take, we then present the modifications that had to be done to the original counterfactual regret minimization algorithm, so that it could be used by us, and the justification for doing so. By modifying the algorithm, it may lead to the modification of some important features that the algorithm provided. Because of that, we then explain what are the implications of having the algorithm modified in the way that we did.

Finally, we present how the algorithm actually works and what it does with the information it receives in every single step to produce the final action that our agent will use.

5.2.1 Why use counterfactual regret minimization

Coup is a game where the information available is incomplete, which leads to the need of making actions and decisions without having all the relevant information to get the best result out of the actions and decisions available at that moment. This will produce regret in a player that makes an action and then, afterwards, gets the remaining information that was missing and realises that if he would have chosen another action, he would have gotten a better result. By minimizing the counterfactual regret and, in turn, the overall regret of the agent, we make it possible for the agent to have a decision making algorithm that thrives in an environment where information is not perfect.

The game also includes the concept of bluff, where a player does an action when he does not have the required character card to do so. Actions that require bluff can be calculated the same way as actions that do not, by calculating the regret they may generate. Taking this into account and by minimizing its overall regret, the player is able to decide if bluffing is the right thing to do in a particular moment, by also considering the actions that have been performed before. This will make him able to play to win in the long run instead of trying to maximize each single action utility, which would potentially increase the risk of being catched bluffing and losing, or instead of trying to minimize risk, losing by never getting an edge over the game.

As mentioned before, to calculate regret, the player will also take into consideration the previously played actions. This makes the continuity of bluffing more stable, since the regret generated from maintaining a bluff is considerably less than the regret the player would get from starting a bluff. By having this property of being able to maintain a bluff, the player is more capable of wining, as being able to maintain a bluff is a considerable advantage in the game of Coup.

Bluff is a big component of the Coup game, and not only must the players be able to maintain it throughout the game, they too need to know on when to capitalize on it. What this means is that the players should know when to stop bluffing and go for the actions they can actually make. Players that did not know that the other agent was previously bluffing, will get the impression that he is bluffing now and consequently challenge the player, resulting on them losing a card. The player that capitalized on the bluff will get a new card, making it easier to start bluffing again, and the challenging player will lose the card. By minimizing regret, not only makes maintaining a bluff easy, but also makes the capitalization of it even easier, as the player that is bluffing will be committed to the bluff for a good amount of time that will lead other players to believe more in him.

5.2.2 Modifications made and why

When calculating the utility of an information set given a strategy using $u_i(\sigma, I) = \frac{\sum_{h\in I, h'\in Z} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(h,h')u_i(h')}{\pi^{\sigma}_{-i}(I)}$, we changed it so we never take into consideration specific histories, but just continue to deal with information sets, as the abstractions are still needed at this level.

We then came up with the following counterfactual utility $u_i(\sigma, I)$ function:

$$u_i(\sigma, I) = \sum_{I'\in(I,\sigma), I''\in Z} \pi^{\sigma}_{-i}(I, I')\pi^{\sigma}(I', I'')u_i(I'') \tag{5.6}$$

Where $\pi^{\sigma}_{-i}(I, I')$ is the probability of state I' being the outcome of the current state I given that the player plays accordingly to σ and $\pi^{\sigma}(I', I'')$ is what we have called the $potentialToWin$ which returns the estimated probability to win of the player. Finally, we only take into consideration the outcomes that make the player victorious so, $u_i(I'') = 1$ if the player is victorious and $u_i(I'') = -1$ if not. This allows us to remove a great amount of possibilities that need to be calculated.

In a way to simplify the function, but still reproducing the same expected results, we went a little further and modified $\pi^{\sigma}(I', I'')u_i(I'')$ into $potentialToWin(I', \sigma)$, which removes all the need to have an utility function for each terminal state, since the ones where the player loses would account to 0 utility and the ones where he wins account to 1 utility. This $potentialToWin(I', \sigma)$ function calculates the potential of the player to win the current game and returns a value between -1, game is already lost, to 1, game is already won.

We had to incorporate this change into the counterfactual utility function, since not only, as we mentioned earlier, there is only one terminal information set that has relevant utility for the player, but also for the fact that calculating all the possible combinations of plays from a certain point of the game until a terminal state would be close to impossible, since that a Coup game may go indefinitely (e.g. 2 players constantly playing the action exchange) and while the game from an outside point of view may look exactly the same, from each individual's point of view the game may have completely changed on every round that goes by, and this does not take into consideration that each play gives information to the other players. So, removing the abstraction of the information sets and dealing directly with histories, and calculating the probability to reach a terminal state from a given history are both parts of the utility function that we had to modify to comply with the needs of our player.

The final counterfactual utility $u_i(\sigma, I)$ function used by our decision making algorithm is:

$$u_i(\sigma, I) = \sum_{I' \in (I,\sigma), I'' \in Z} \pi^{\sigma}_{-i}(I, I') potentialToWin(I', \sigma) \tag{5.7}$$

Algorithm 1 CalculatePotentialToWin

```
procedure POTENTIALTOWIN($playerNum, I, \sigma$)
    $totalPotential \leftarrow 0$
    $agentPlayer \leftarrow I.getPlayer(playerNum)$
    if $agentPlayer.isPlaying()$ then return 0
    else
        $gameWon \leftarrow true$
        for all $player$ in $I.getPlayers()$ do
            if $player \neq agentPlayer \wedge player.isPlaying()$ then
                $gameWon \leftarrow false$
            end if
        end for
        if $gameWon$ then return 1
        end if
    end if

    for all $player$ in $I.getPlayers()$ do                    ▷ Calculate other players' potential
        if $player \neq agentPlayer \wedge player.isPlaying()$ then
            $attackPlayerProb \leftarrow probOfAttacking(player, agentPlayer, I, \sigma)$
            $getCoinsPotential \leftarrow calcCoinsPotential(player, I, \sigma)$
            $playerCoinsToKillPotential \leftarrow coinsToKillPotential(player, agentPlayer, I, \sigma)$

            $playerPotential \leftarrow player.numberActiveCards() * attackPlayerProb * (player.numberCoins() + getCoinsPotential) / playerCoinsToKillPotential$
            $totalPotential \leftarrow totalPotential + playerPotential$
        end if
    end for

    $agentCoinsPotential \leftarrow coinsPotential(agentPlayer, I, \sigma)$    ▷ Start calculating agent's potential
    $agentCoinsToKillPotential \leftarrow coinsToKillPotential(agentPlayer, I, \sigma)$    ▷ Is the average between all possible targets
    $agentPotential \leftarrow agentPlayer.numberActiveCards() * (agentPlayer.numberCoins() + agentCoinsPotential) / agentCoinsToKillPotential$
    $totalPotential \leftarrow totalPotential + agentPotential$

    return $agentPotential / totalPotential$
end procedure
```

5.2.3 Implications of the modified utility function

The modifications made to the algorithm only affect the way that the probability to a terminal state is calculated and with that the utility of the final state. In other words, the change only modifies how $\pi^{\sigma}(h, h')u_i(h')$ is calcu-

Algorithm 2 CalculateCoinsPotential

```
procedure COINSPOTENTIAL($player, I, \sigma$)
    $taxProb \leftarrow \sigma.getActionProb(tax, player) * calcActionSuccessProb(tax, player, I, \sigma)$
    $foreignAidProb \leftarrow \sigma.getActionProb(foreignAid, player) * calcActionSuccessProb(foreignAid, player, I, \sigma)$
    $incomeProb \leftarrow \sigma.getActionProb(income, player) * calcActionSuccessProb(income, player, I, \sigma)$
    $stealProb \leftarrow \sigma.getActionProb(steal, player) * calcActionSuccessProb(steal, player, I, \sigma)$ ▷ Is the average between all possible targets

    $gettingStolenProb \leftarrow 0$
    for all $opponent$ in $I.getPlayers()$ do
        if $opponent \neq player \land opponent.isPlaying()$ then
            $stealPlayerProb \leftarrow \sigma.getActionProb(steal, opponent, player) * calcActionSuccessProb(steal, opponent, player, I, \sigma)$
            $gettingStolenProb \leftarrow gettingStolenProb + stealPlayerProb$
        end if
    end for

    $coinsPotential \leftarrow 3 * taxProb + 2 * foreignAidProb + 1 * incomeProb + 2 * stealProb - 2 * gettingStolenProb$
    return $coinsPotential$
end procedure
```

Algorithm 3 CalculateCoinsToKillPotential

```
procedure COINSTOKILLPOTENTIAL($player, targetI, \sigma$)
    $assassinateProb \leftarrow \sigma.getActionProb(assassinate, player, target) * calcActionSuccessProb(assassinate, player, target, I, \sigma)$
    $assassinateSuccesProb \leftarrow calcActionSuccessProb(assassinate, player, target, I, \sigma)$
    $coupProb \leftarrow \sigma.getActionProb(coup, player, target)$
    $combinedProb \leftarrow assassinateProb + coupProb$

    $assassinateRelProb \leftarrow assassinateProb \; / \; combinedProb$
    $coupRelProb \leftarrow coupProb \; / \; combinedProb$

    $coinsToKill \leftarrow 3 * assassinateRelProb / assassinateSuccessProb + 7 * coupProb$
    return $coinsToKill$
end procedure
```

lated, but since $potentialToWin(I', \sigma)$ still provides the same results expected from the generic algorithm, values between -1, if the game is lost, and 1, if the game is won, we can conclude that the our modified version of the algorithm will still provide the same properties as the original algorithm.

Taking into consideration that our modified version of the algorithm produces the same type of results as the generic one on the calculations to reach a terminal state, we still need to consider the fact that those calculations may be producing wrong results given a certain state. This is something that has a certain degree of difficulty as the game is mathematically complex, nevertheless, our approach tries to match reality by considering a large amount of game characteristics and mechanics, and including them into the calculations.

Changing some of the components may give a feeling of losing the capabilities of the overall counterfactual utility minimizer algorithm, but the fact that our modified version already includes, as mentioned before, some of the game characteristics into its calculations, make the algorithm more specific, but not less precise.

5.2.4 Flow of information throughout the algorithm

How does the algorithm actually produce an action based on all this? Is the question that we will be answering here, going through the various steps throughout the algorithm and presenting the transformations done to the information that entered it.

The decision making algorithm takes the current information about the other players directly from the Theory of Mind component, that will be explained later in the Chapter 6, and the current state of the game, which is taken from the information processed in the Memory component of our agent.

For each action that is available to our agent in the current state of the game, it will generate all possible outcomes that may happen when the action is indeed used, it will also generate for each of these outcomes their corresponding probability, that is based on the information that our agent has stored in its Theory of Mind component.

After having the the possible outcomes and corresponding probabilities for the available actions, the algorithm will calculate, for each of the outcomes, the value they have towards winning the game, which is done in $potentialToWin(I', \sigma)$, it then multiplies that value with the outcomes corresponding probability. By adding all the outcomes product between value and probability, for a specific action, the value of said action is calculated.

By multiplying each action's value with its corresponding probability in the previous agent's strategy, the algorithm is able to produce the value relatively to that previous strategy. This will be done for all past strategies.

Having the value of past strategies towards the current state of the game, the algorithm will proceed to calculate the counterfactual regret of each action by summing the difference between the action's value and a past strategy's value, multiplied by the probability of reaching the current state of the game using that past strategy. By summing all these value for each past strategy and then dividing by the number of past strategies, the algorithm is capable of effectively calculate the counterfactual regret for each action.

A new strategy is then produced based on the percentage of counterfactual regret that each action gives, when considering the total amount of counterfactual regret of all actions summed. After having the new strategy defined, the agent will then play the action with the most probability of being played, in other words, the action that produces the least amount of counterfactual regret.

All this process and calculations is done in $GenerateNextStrategy(I', \sigma)$.

The process mentioned before is only used when a decision involves an action, challenge or counteraction. When the decision involves losing a card or changing cards, the algorithm uses a different approach.

The different approach that the algorithm takes to make a decision that involves losing or changing cards, is very similar to the way it makes a decision for a normal action, but since losing or exchanging cards is something that is difficult to include in the strategies, the calculations are only done taking into account the value of the outcomes of losing or changing certain cards.

After calculating the value of the outcome for each possible action of losing or changing cards, the algorithm will select the action which provides the most value for our agent, or in other words, that provides the biggest potential to reach a favorable end game state.

All these calculations to make a decision that involves changing or losing cards is done in $generateNextCardAction(I', \sigma)$

Algorithm 4 GenerateNextStrategy

```
procedure GENERATENEXTSTRATEGY($playerNum, I, \sigma$)
    $availableActions \leftarrow I.getAvailableActions(playerNum)$
    $regretForNotPlayingActions$                ▷ HashTable<action, regret> — regret starts at 0

    for all $\sigma^t$ in $\sigma$ do
        $generalStratUtility \leftarrow 0$                ▷ $u_i(\sigma^t, I)$
        $actionsUtility$                ▷ HashTable<action, utility>
        $probReachCurrState \leftarrow \sigma^t.probOtherReachCurrState(playerNum, I)$                ▷ $\pi^{\sigma^t}_{-i}(I)$

        for all $action$ in $availableActions$ do
            $utility \leftarrow 0$
            $possibleOutcomes \leftarrow I.generatePossibleOutcomes(action)$

            for all $outcome$ in $possibleOutcomes$ do
                $utility \leftarrow utility + outcome.getProb() * potentialToWin(playerNum, outcome, \sigma)$
            end for

            $generalStratUtility \leftarrow generalStratUtility + utility * \sigma^t.getActionProb(action)$
            $actionsUtility.put(action, utility)$
        end for

        for all $action$ in $availableActions$ do
            $currActionRegret \leftarrow regretForNotPlayingActions.get(action)$
            $currActionUtility \leftarrow actionsUtility.get(action)$
            $regretForNotPlayingActions.put(action, actionUtility.get(action) - generalStratUtility)$
        end for
    end for

    $totalRegret \leftarrow 0$

    for all $action$ in $availableActions$ do
        $positiveRegretForNotPlaying \leftarrow max(actionsRegretForNotPlaying.get(action), 0)$
        $actionsRegretForNotPlaying.put(action, positiveRegretForNotPlaying)$
        $totalRegret \leftarrow totalRegret + positiveRegretForNotPlaying$
    end for

    $evenlyProb \leftarrow 1/availableActions.number()$
    $newStrategy$                ▷ $\sigma^{T+1}$

    for all $action$ in $availableActions$ do
        if $totalRegret \neq 0$ then
            $newStrategy.setActionProb(action, actionsRegretForNotPlaying.get(action))$
        else
            $newStrategy.setActionProb(action, evenlyProb)$                ▷ Same probability for each action
        end if
    end for

    return $newStrategy$
end procedure
```

Algorithm 5 GenerateNextCardAction

```
procedure GENERATENEXTCARDACTION($playerNum, I, \sigma$)
    $availableActions \leftarrow I.getAvailableCardActions(playerNum)$
    $bestUtility \leftarrow -2$                                    ▷ Utilities go from -1 to 1
    $bestAction$                                                   ▷ Will hold the best available action

    for all $action$ in $availableActions$ do
        $outcome \leftarrow I.generateCardOutcome(action)$
        $actionUtility \leftarrow potentialToWin(playerNum, outcome, \sigma)$
        if $actionUtility > bestUtility$ then
            $bestUtility \leftarrow actionUtility$
            $bestAction \leftarrow action$
        end if
    end for

    return $bestAction$
end procedure
```

Chapter 6

Implementation

In this chapter we present how we actually implemented the more general solution that we defined in Chapter 4. We also mention how everything works from an individual standpoint and how everything works together to produce exactly what we want.

Starting with an overall view of our implementation, the overall system is presented and then how the communication between the different components is made, after that, we continue by presenting our Agent's Architecture, which include its various components. Every single component is thoroughly explained and its inputs and outputs are mentioned. Then comes the part related to the Coup game that is played by both our agent and the human players. Finally, we present how our agent interacts with its human opponents, as well as how the system works to produce such interactions.

6.1 Overall System

In this section we show how the overall systems functions, going through a little explanation for the different components that compose it.

This system was built with the purpose of testing our hypothesis and for that, we had to take into account a great number of things. Starting with the central piece of the system, which is Thalamus, this components objective is to receive messages from the different components and send them to the components that are expecting to receive it. To do so, Thalamus is composed by a scheduler integrated with a MOM (Message-oriented middleware) [**?**], which allows for it to have asynchronous and abstract sides of communication, while still supporting synchronously distributed behaviours, which in our case are used to make our robot communicate. The Thalamus scheduler also allows the use of synchronized actions and events that are originated from the same behavioural language as the behaviours used to communicate. This provides the option of sending and receiving events, which is a good way to send information essentially between the game and our agent.

Our agent architecture, which is basically the core of our agent, will decide which actions and interactions should our agent perform, depending on the various situations that it will perceive. It is directly linked to a thalamus bridge so that it can convert the messages from Thalamus and filter them depending on which ones it wants to receive.

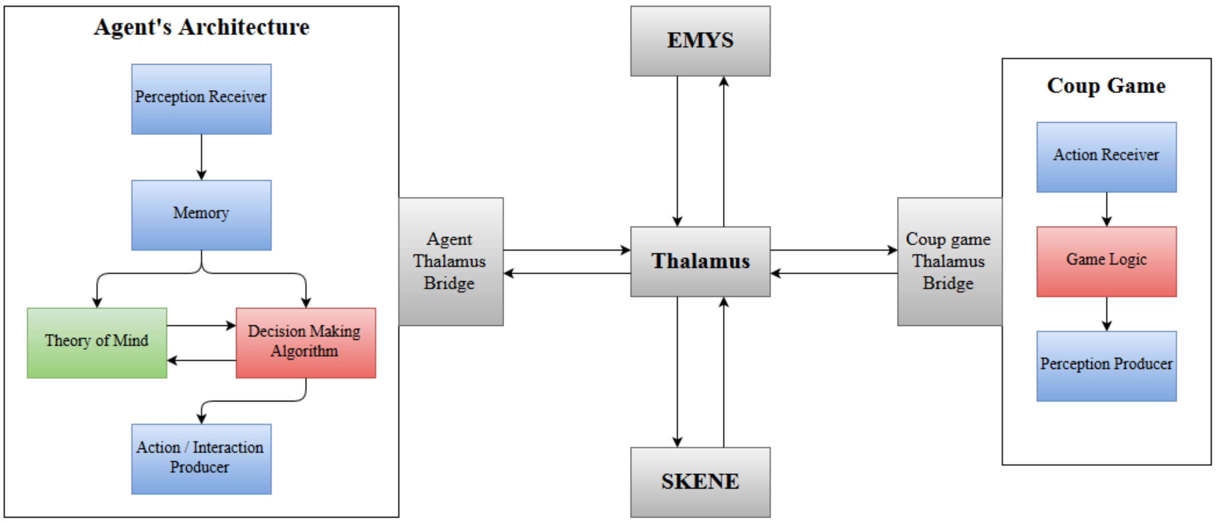

Figure 6.1: Our Overall System

Similar to our agent architecture, the Coup Game component also has a thalamus bridge for the same exact reasons as the architecture. The coup game will send the perceptions that our agent will receive, as well as receive actions from our agent to apply into the game, and consequently show the human players how our agent as acted in the game.

The EMYS component represented in our overall system, not only includes the physical robot that will interact as the agent with the human players, but also the software beyond it that processes the messages received from thalamus and will then how the robotic head will move.

The SKENE component is only used as part of the interaction towards the human players, nevertheless, it is used as a translator that receives Skene Utterances, which will be mentioned later what they are, from our agent architecture with the objective to process them and send events directly to EMYS, which will then produce the interactions relatively to those Skene Utterances received.

Finally, our overall system can be seen in Figure 6.1.

6.2 Agent's Architecture

The base of our agent's architecture is FAtiMA (Fearnot AffecTIve Mind Architecture). FAtiMA is an Agent Architecture that is capable of planning by taking into consideration the use of emotions and personality, influencing the agent's behavior accordingly [25]. To do so, FAtiMA uses Appraisal Derivation and Affect Derivation to produce an emotional state in the agent, and then use that state to influence its believes and plan of action. Not only that, but FAtiMA is also composed by multiple components that easily integrate with the rest of our system, by providing an extendable interface for perception receiving and action production.

While we do not currently use the emotional component of FAtiMA, our Agent's Architecture was initially built upon it to provide for an easier way to integrate with it in the future. As mentioned earlier, we mainly use FAtiMA for the seamless integration with the other main components of our system.

Regarding our Agent's Architecture, it is divided in different modules, where every single one of them has a

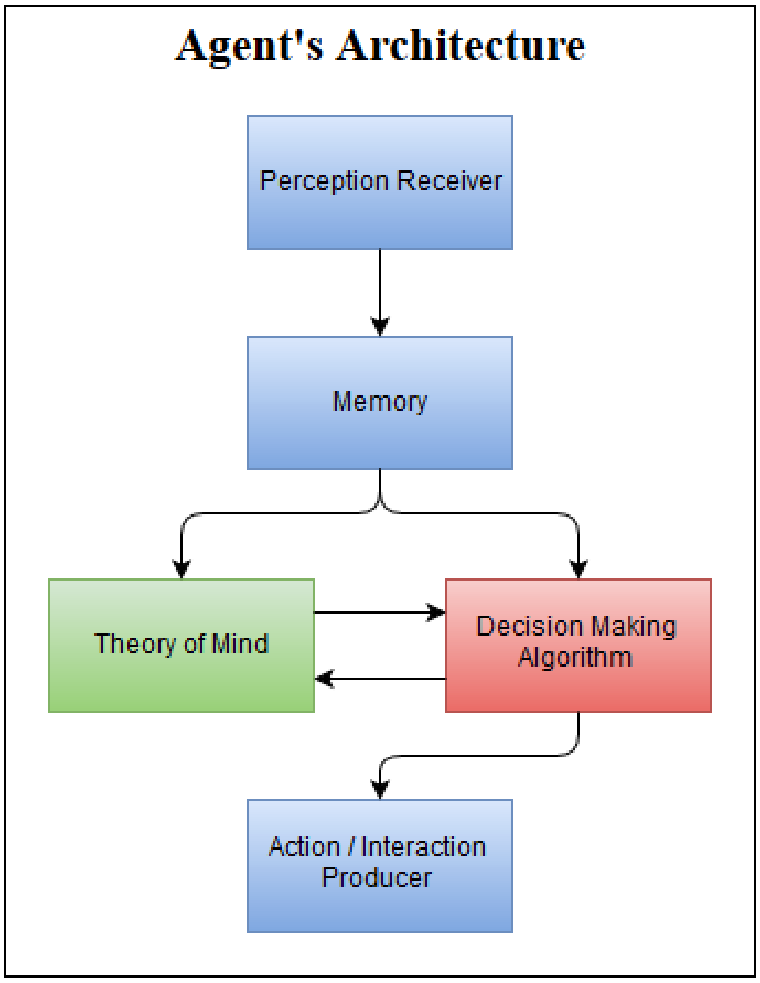

Figure 6.2: Our Agent's Architecture

certain input that depends on the output generated by the previous module, and produces an output to provide for the next module.

The modules or components used in this architecture are the Perception Receiver component, the Memory component, the Theory of Mind component, the Decision Making Algorithm component and finally, the Action/Interaction producer.

All the before mentioned components and our overall Agent Architecture can be seen in Figure 6.2.

6.2.1 Perception Receiver

In the perception receiver module, the grand objective is to receive the various perceptions that are meant for the agent, be those perceptions events from the game, which could be actions from the players, including the agent, and events to make the agent play, or events sent from the other components associated with all the system, for example, when a the robot as stopped animating or a speech as ended.

This component is already incorporated in the FAtiMA architecture that we are using as a basis for our agent's architecture. It's sole purpose is to serve as an interface to receive information from the overall system, it then sends it along to the next component as an event object, so that the processing of it is easier. Since it is a component already developed, we just had to specify which types of events we want to deal with, as to filter the information that the agent's architecture is receiving.

Each event is composed of a subject, a target, a name for the event and finally, a list of parameters which, contrary to the other fields, can take any kind of of value.

6.2.2 Memory

This component holds all the events and game states that the agent perceived, which does not include information created by the agent, such as, its knowledge about the other players.

In here, the events are applied to the game state so that it changes towards the current game state. While always keeping the current game state up to date, this component also keeps a history of all the events, so that the agent can keep the order of events as an input into its calculations, which is needed since equal game states can be reached from different ways.

The information stored here is ordered sequentially and organized towards the player that performed the action given by the event. With this, not only the agent possesses information on how the game developed towards the current state, but also has the information organized in a way that it is possible to know, with ease, which player did which action, allowing a much easier job for the components that have to deal with this kind of information. The information is then sent to two different components that deal with it in two different ways, which are the Theory of Mind component and the Decision Making Algorithm component.

6.2.3 Theory of Mind

Regarding the Theory of Mind component, the degree of specification given here will be much lower than its actual full form, which will be developed later. To give an overview of what it does, we first have to make a little introduction to what it is.

Theory of mind is the capacity to attribute various mental states, beliefs, desires, knowledge, to oneself as well as to understand that others do exactly the same. In our case, this will deal with how our agent perceives itself and how it perceives the human players. Increasing the complexity of a theory of mind is done by incrementing its levels. An example of theory of mind of level three would be what the agent A thinks of what agent B thinks of what another agent, let's call it C, thinks. This module uses this model of how to perceive the other agents and himself.

What it does is receive the information from the Memory component about what has happened and what is the game state, and calculates the probability of each player having a certain card and the probability of doing a certain action. This information is then given to the Decision Making Algorithm component. These probabilities mentioned before towards the agent itself, are not calculated through this component, but are received from the Decision Making Algorithm component and kept here so that the same type of information is kept in the same module.

The current version that we have in our agent architecture is a much simpler one that only takes into account events received from the memory about actions that have a great impact in our agent's way of playing. One example of this would be having our agent perform an action that requires the Duke character card, while not having it, and then be challenged on it. This would lead to our agent losing a card. If we did not update our theory of mind component, the agent could possibly redo this play even though its opponents know for sure that he does not have a Duke character card now. So what our simple version of the Theory of Mind component does is to change all opponents' challenge probability to 100% for any action that our agent could use that required the Duke character card, making it impossible for our agent to redo such action, since it knows for a fact that he will lose another card with 100% certainty.

6.2.4 Decision Making Algorithm

The Decision Making Algorithm is the module where the actions performed by the agent are generated, this also includes the interactions done, that require the robot, to interact with the other players in the real world.

An overview of what this component does, since a more detailed explanation was given before in the Chapter 5, is presented here so that the an understanding of the flow of information inside the agent is created.

The main inputs are the current game state and the sequence of events of each player, which are both provided by the Memory component, and the probabilities of the players having certain cards and performing certain actions in the future, which is given by the Theory of Mind component. With all this information and the use of a modified version, towards a more domain specific version, the agent is able to produce the next action that it will make unto the current game.

This information is then sent to the final component of the agent's architecture, so that an action and a possible interaction can be sent towards the game.

6.2.5 Action/Interaction Producer

In this final component, the information received, as mentioned before, is the action produced by the Decision Making Algorithm component. This action is then processed alongside with what agent knows about the game to produce a possible interaction that will be sent to the robot in a way to give the agent a social presence and with that be capable of producing behaviour that is deceptive.

Another responsibility of this component is to send both the new action and the interaction to Thalamus, the center component of the overall implementation, which will then send the action and interaction to the systems that are meant to receive them.

6.3 Digital Tabletop and Coup Unity Game

To make a user interface where multiple human players and our virtual agent can play, we need some kind of device that is able to run software and allow the usage of this software (game) by various players. For all the players to be able to play together on the same platform, this needs to be big enough so that all players are able to play on the same device.

The solution for this, as mentioned before on the Problem/Solution chapter, is a digital tabletop that is big enough to allow multiple players and is able to run software.

Our virtual game of Coup, that is used to allow both physical and virtual players, is built using Unity as it provides the biggest spectrum of operating systems deployments and is a good platform to build an interactive game like Coup.

The game had to be built in a way that allows the modification of its own state through the use of a virtual interface as well as a physical interface. For the virtual interface we used a mechanism that receives events from a central messaging system called Thalamus. All the events received had to be defined through an interface to allow the defined events to be received from Thalamus unto the the Unity game. These events would then be applied unto the game state, modifying it. Things to take into consideration here would be the fact that a certain event could have some king of delay getting into to the Unity game, be it from a delay on the network, or during the process of such event, and that by then, the current phase of the game would not allow such type of action. To solve this problem a logic barrier had to be built so that the events received would only be applied if the current game phase would allow it. Another problem with this kind of interface is the fact that multiple events can be received in a short amount of time, if one of these events gets delayed somehow, the order of all events will be abnormal, making events, that do not make sense to be applied on the current game phase, appear before the ones allowed. In order to solve this problem, a queue of events had to be created so that the events would not be discarded. This queue would then be run through to see if any of the events could be applied on a particular moment, if so, the event would be removed, the current game state would change based on the event, and then the process continued, in other words, the queue is ran through as many times as needed to empty it or until no more events can be processed, in which occasion, the remaining events are discarded.

The way for the agent to interact with the game has already been mentioned, but the information that the agent sends and receives still needs to be defined. For every action or modification of the game state, the Coup Unity game sends an event to the central messaging system, Thalamus, that, when applied to the previous game state, produces the current game state. This kind of event is very similar to the ones received by the game and have to be defined as an interface of Thalamus, so that the agent can indeed receive them.

Regarding the interface that allows the humans to interact with the game, it is graphically represented through the digital tabletop. Each of the players, including our virtual player, will have a determined position where all their information is. This information includes their cards, that are hidden by default, a button that shows or hides the cards, depending on their current state, hidden or shown, a number that represents their current number of coins and finally, a list of actions, much like the original summary card, that not only allows the players to do the actions, but also provides all the information they need in terms of actions, counteractions and which characters are needed for any of those. Using any action that requires another player as a target will create a new list of valid players to target, so the acting player can choose who to use his action on. All actions and counteractions that require one of the characters to use, will prompt all other, still playing, players if they want to challenge the action, the same will also be done when an action can be countered, asking the players that are allowed to counter it, if they want to do so.

All this provides the players with the means to know the current game state, including our virtual agent's player information, as though it was a human player, while allowing a similar experience as the original board game.

6.4 Robot EMYS

In terms of the before mentioned physical representation of our agent, we had the requirement of using a robot that would be capable of expressing the thoughts of our agent and with that, interact with the human players. The robot was also required to resemble a human being, or some kind of conscious being, so that intentionality of interaction and directionality of interaction were present. Something that should not be missed is that the more it resembles a human being, the easier will it be for the human players to see our agent as one of their own and, consequently, put more trust on it.

Our chosen robot to fulfill these requirements was EMYS (EMotive headY System) (Figure 6.3), which is an emotive robotic head designed and built within the EU FP7 LIREC project[1]. This head is composed of three discs and equipped with a pair of eyes and eyelids that are movable. Everything is mounted on a movable neck[26].

The head is capable of speech through the use of a speaker, and is able to produce prerecorded or synthesized voices. To perceive the environment, it is equipped with a colour CMOS camera Logitech Sphere AF[2], Kinect sensor[3] and a lapel microphone. All this allows both speech recognition and speaking, visual perception of the environment, eye-tracking of humans and objects, and even the establishment and maintaining of eye-contact with humans and expressing emotions. While the robotic head allows for all the things mentioned before, we will not be using the camera, nor the Kinect sensor, since it have a great increase in terms of our agent's complexity, and would not improve that much the possibility to meet this work's objective, so we consider it to go a little beyond our work's scope.

By using an already developed system to perform interactions through EMYS, we are able to easily and effectively make our agent interact with the human players, making this interaction as complex or simple as we want, do to its easiness of use.

The component that produces the interactions is called Skene, which is a semi-autonomous behaviour planner capable of semi-automated behaviour [27]. To produce such semi-automated behaviours, Skene takes as input a high-level behaviour description language that was developed by a team that included non-technical partners from psychology, which is called Skene Utterances, and perception information, such as target locations. An example of a Skene Utterance would be "¡gaze(player0)¿ You are great player, but he is not ¡glance(player1)¿", where the robot would look to player 0, say "You are great player, but he is not" and then momentarily look to player 1 and then back to player 0. The output of Skene consists on both the scheduling of BML (Behaviour Markup Language) and non-BML actions (such as sounds or application commands), which are then sent to the EMYS component so that it will interpret them and in turn make the actual robotic head move and reproduce sound according to the Skene Utterance.

Our underlying agent architecture, FAtiMA, has the advantage of already having a full functioning emotive component that, based on some appraisal variables defined by us, is capable of producing emotions which in turn are sent as Skene Utterances to Skene and consequently being represented in our agent depending on the conditions of the current game state, or we can even generalize it more and have emotions that reflect the state of our agent towards a whole session of games. By having this feature at our disposal, the possibility of using these emotions

[1]http://www.lirec.eu
[2]http://www.logitech.com/en-us/support/webcams/quickcam-sphere-af?osid=s14&bit=32
[3]http://www.sxbox.com/Kinect

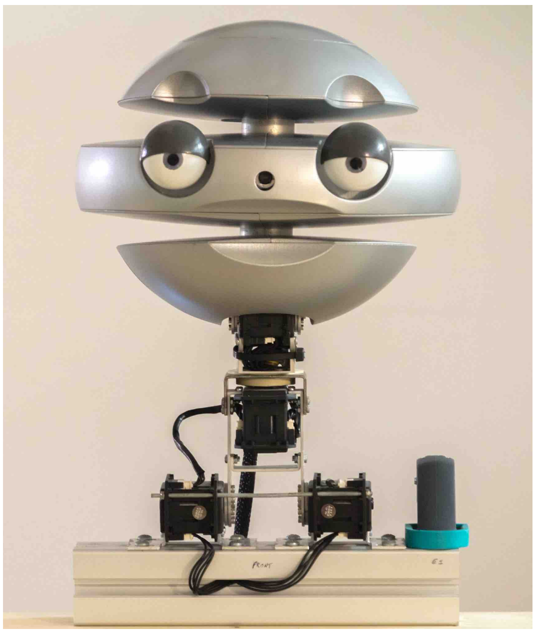

Figure 6.3: EMYS - EMotive headY System.

to emulate a more human interaction with the agent's human opponents exists. The problem with this feature, that we could use, is the fact that since the game that our agent plays is a deception game and the objective of this work being to capitalize on a deceitful behaviour, showing what our agent feels depending of the game situation would put it at an incredible disadvantage in this kind of game, since its opponents would be able to read it like an open book and know exactly what our agent was trying to do. Nevertheless, we could still use this in a way that the agent would only present emotions with the sole purpose to deceive, in other words, the agent would show, through EMYS, emotions that would not necessarily represent what a human in its game position would truly feel. While being a big advantage to have in our system, the complexity of building such emotive system, which would also require a stronger theory of mind component than the one that currently we have, makes it go a little beyond the scope of this work for the time being. It surely is something to take into account for future work, as the reproduction of emotions on EMYS is easily done through the use of the already defined architecture.

Chapter 7

Experiments

In order to test our virtual agent so that we can prove that our hypothesis is correct, we have designed a use-centered study with people playing the Coup game against our virtual agent.

The aim of the experiment, as mentioned before, is to see if our virtual coup agent is capable of playing at the same level of human beings, which include not only deceiving them, but also a small amount of discovering when the humans are lying. We also wanted to measure how the human players would react to being deceived by our agent, as it is physically represented by a robot, which in itself breaks the *sincerity assumption* [28] which is an assumption usually made when studying agent communication, that the agents involved in communication are telling the truth and acting without a deceptive intention.

The equipment we used to perform this experiment was:

- **A MultiTaction Ultra Thin Bezel Display**[1], which is a 55" display unit with interactive multiuser LCD, capable of tracking an unlimited amount of touch points, including hands, fingers, fingertips, 2D markers and real-life objects, with object recognition, as our digital tabletop;
- **An EMYS** (EMotive headY System), as our robot that physically represents our agent and interacts with the human players;
- **Three Lavalier microphones** to record the human player voices;
- **Four cameras** for filming, where one had the sole objective to record the interaction of the agent with the human players, and the other three focused on the behaviour of each of the human players;
- **One Coup game set** to explain the game to participants that did not know how to play it, or did not remember it so well.

In terms of the risks of this experiment, since we are trying to study people behaviours towards our agent and how it affects the game, we want the conditions to be as close as possible to an environment where they can freely play and act as though they were not being watched. While one of the cameras was easily seen inside the room, it had the advantage of being put at a high enough height that the normal person eyesight would never meet the camera. All other cameras were camouflaged so that the players would not be aware of them, even if they were

[1]http://www.multitaction.com/products/displays/ultra-thin-bezel/

looking for them. The microphones on the other hand were explicitly asked to be worn by the participants, which by doing so may slightly alter the spoken communication from the human players.

Some of these mentioned aspects have to be taken into consideration when analysing the results obtained on this study, as the players' behavioural aspects may slightly differ from the reality, even though the conditions were as closely as possible to those of a natural environment.

7.1 Methodology

7.1.1 Sample

Regarding the sample, a total of 57 university students took part of this study, where 38 were male and 19 female, with ages ranging from 19 to 29. Participants were randomly allocated to a type of group, either they played with two more human players and our agent, or played one on one with our agent. At the same time, participants were randomly allocated to one of two different conditions: the lie condition, where our agent was using the decision making algorithm with all available actions, and a no lie condition, where the agent would be constrained to using only actions that did not require having certain cards, or the actions and counteractions that he possessed the card needed. In both conditions, the capacity of challenging other players bluffs were equal and did not differ from one another. It was a 2x2 design with variance in the type of group the participant was part of and the way our agent played.

All participants signed a consent form so that all the information they provided could be used in this study.

7.1.2 Procedure

Upon arrival, participants were allocated to just one of the types of group (playing individually against our agent or in a group with other two human players and the agent) and conditions (the lie condition, the no lie condition). Participants were not aware that the lie/no lie condition existed, so their initial perception of our agent did not differ between those two conditions.

They started by filling a pre questionnaire without supervision and then, after finishing it, the game Coup was explained using the original board game, so that all the players had at least a basic understanding of the game and were capable of playing it. After the explanation, participants were then guided to the digital tabletop where EMYS, the physical representation of our agent, was already mounted on its place, taking into account the in game positions, and then attached a microphone to themselves.

In the group condition, where three participants and the agent played, the number of Coup games played by the participants was three, while in the individual condition, participants played five games. The first game of the series in both group conditions was supervised by one of the people that directed the experiment, this was done so that any doubt towards the game rules or the game interface was easily clarified and with that the game experience would be better and the results would not include mistakes due to the lack of understanding of how to play.

After the game session, participants were taken to a different room and then filled a pos questionnaire, without being supervised, in which the questions were in regard on how they felt about the interaction with our agent, more specifically, through the use of the EMYS robot as our agent physical representation.

The participants were then thanked for their participation in our experiment and contributing for our study, and were gifted with a coupon to get a free ticket for any movie in a certain group of cinemas.

7.1.3 Measures

To understand how trust would be perceived by a human player regarding a robot, by playing the deceptive game of Coup, two questionnaires were presented.

Before the game session, participants responded to the Big Five Questionnaire [29] to ascertain the participant personality type (validated for the Portuguese population by Lima and Castro 2009), followed by an interpersonal trust scale, the Multidimensional Trust Scale [30] to see the level of trust that the participants had in themselves and others. For this scale, only the dimensions of Self and Others to ascertain the global score of trust were used, leaving the Environment dimension out due to its low internal consistency value. This was done before the explanation and playing of the game, so that the participants awareness towards the real objective of the questionnaire was not known beforehand and their answers were as sincere as possible.

After the interaction with our agent, by playing the game, the participants filled another questionnaire, this time being the Godspeed Questionnaire [31], in order to understand if the perception of the robot changed regarding the condition that they were allocated to. For this, a Credibility scale was also applied (taken only the Trustworthiness dimension from Ohanian, 1990) in order to understand if participants felt when our agent was being dishonest (all this was answered in a 5-point Likert scale) [32]. A Trust scale specific for Human-Robot Interaction was then used to perceive the level of trust the participants had on our agent, through its physical representation with EMYS [33].

In the last questionnaire, participants also answered directly to what they thought of our agent. They were asked on what level they would put the Coup playing skill of our agent, how much they thought that the agent lied, and how well it lied. All this questions were answered by using a 5-point scale.

Other measures that we used to understand how well our agent was capable of deceiving and ultimately winning the game against human players were the percentage of victories that the agent achieved, the number of times that he got challenged and won, and finally, the number of challenges he got per session. The first measure was used to know if in reality, an agent that has the capability of lying is indeed more beneficial than an agent that only tells the truth, the second measure to know if the agent was capable of successfully deceiving its human opponents, the last measure was needed to know if the participants were getting more doubtful of the agent or not.

7.2 Results

Beginning with the first questionnaire that the participants answered, we focused primarily on the Multidimensional Trust Scale, that had the objective of seeing the level of trust that the participants had in themselves and others. This would indicate if the level of trust is similar among our sample in the different conditions that we had defined for our experiment. These results are presented in Table 7.1

In terms of the results we have gotten from the second questionnaire, we have separated them between the group types, since the results heavily differ from the sample that were attributed to the group condition (three

	Lie		Truth	
Trust Value	Mean	Std. Deviation	Mean	Std. Deviation
Individual	77,20	8,27	80,53	6,91
Group	77,73	8,21	77,00	6,30

Table 7.1: Trust value mean for the participants in both group types and conditions in a scale of 0 to 120

	Lie		Truth	
Measure	Mean	Std. Deviation	Mean	Std. Deviation
Anthropomorphism	3,07	0,60	2.96	0,56
Animacy	3,39	0,31	3,16	0,57
Likeability	3,56	0,51	3,29	0,84
Perceived Intelligence	3,85	0,62	3,65	0,82
Credibility	2,61	0,72	2,20	0,68

Table 7.2: Individual condition mean and standard deviation for the different dimensions extracted from the Godspeed Questionnaire plus the Credibility scale, in a 5-point Likert scale

participants played with our agent) to the sample that was included in the individual condition (one on one against our agent).

We will start by presenting the results from the Godspeed Questionnaire, in Table 7.2 for the Individual group type condition and in Table 7.3 for the Group group type condition, that was used to understand if the perception of the robot changed regarding the condition that the participants were allocated to. Another thing that should not be missed regarding the results from the Credibility scale, is that the higher the value, the more dishonest the behaviour of our agent was perceived.

The results regarding the Trust scale specific for Human-Robot Interaction are presented in Table 7.4 for the Individual group type condition and in same table for the Group group type condition. These results represent the percentage of trust that the participants had in our agent.

Another measure that we considered was how well the participants thought that our agent played, and for that we asked them exactly that question on the second questionnaire to which they answered in a 5-point Likert scale The mean of the score given by the participants in the Individual condition is presented in Table 7.5 and the mean of the score relative to the participants in the Group condition is also presented in the same table.

Having the results based on the perception of the participants towards how well the agent played gives us an idea on how well he played, but to really get a precise value on that department, we had to the real percentage of games that our agent won. The results that show that were obtained from the logs produced while the participants were playing and give us the percentages precisely. The percentage of victories that our agent got against a single opponent (Individual condition) is presented in Table 7.6, as well as the percentage of victories it had against a

	Lie		Truth	
Measure	Mean	Std. Deviation	Mean	Std. Deviation
Anthropomorphism	2,89	0,64	2,80	0,81
Animacy	2,91	0,84	2,83	0,71
Likeability	3,08	0,62	3,08	0,45
Perceived Intelligence	3,67	0,58	3,72	0,35
Credibility	2,47	0,78	2,57	0,98

Table 7.3: Group condition mean and standard deviation for the different dimensions extracted from the Godspeed Questionnaire plus the Credibility scale, in a 5-point Likert scale

	Lie		Truth	
Trust Score	Mean	Std. Deviation	Mean	Std. Deviation
Individual	67,43%	10,43%	59,86%	14,35%
Group	59,78%	11,59%	60,04%	12,60%

Table 7.4: Mean trust score and standard deviation for the Individual group type condition and Group group type condition, in percentages

	Lie		Truth	
Played Well	Mean	Std. Deviation	Mean	Std. Deviation
Individual	4,07	0,92	3,67	1,67
Group	3,80	1,60	4,00	0,73

Table 7.5: Mean score given by the participants in both conditions to how well our agent played in a 5-point Likert scale

group of opponents (Group condition).

To know if by having an agent that only tells the truth, participants would pickup on it and reduce the amount of times they challenged our agent, we captured every single time that a challenged happened against our agent for both the lying and the truth condition. The results for this are given as number of times that the agent got challenged per session, which in the Individual condition were five games, while being only three games for the Group condition. These are presented in Table 7.7.

Already having the number of times that our agent got challenged, we went even further and also retrieved the amount of times, from those that he got challenged, that he won, in other words, the amount of times that when he got challenged, he actually had the required card to do so. We have presented the average amount of times that the agent got challenged and won per session in Table 7.8. This will give us information towards the fact that our agent can actually capitalize on lying and with that still make his opponents try to challenge him when he is telling the truth.

7.3 Discussion

Going into the analysis of the results we got, we will be starting from the very beginning. The results we got regarding the Multidimensional Trust Scale had the objective to prove that our participants, having been attributed to different group types and conditions, were still a good sample in terms of their initial trust value.

As is presented in Table 7.1, the participants from both group type conditions have very similar values for the trust value, not only that, but within both group types, even for the different lying conditions (lie and truth) the means are practically identical, appearing only a small discrepancy between the Lie condition and the Truth conditions for the Individual group type condition, where they differ for only a value of 3, which in a total of 120 possible points is 2,5%. Making a statistical analysis, we came to the conclusion that the difference is not

	Lie		Truth	
Victory	Mean	Std. Deviation	Mean	Std. Deviation
Individual	56,00%	3,50%	49,00%	7,40%
Group	23,40%	9,10%	25,00%	8,40%

Table 7.6: Percentage of victories our agent had against participants in the Individual condition and in the Group condition

	Lie		Truth	
Times Challenged	Mean	Std. Deviation	Mean	Std. Deviation
Individual	6,00	2,30	4,00	2,20
Group	5,40	2,13	1,25	1,71

Table 7.7: Number of times our agent got challenged on average per session against the participants in both conditions

	Lie		Truth	
Challenged Won	Mean	Std. Deviation	Mean	Std. Deviation
Individual	3,87	1,30	4,00	2,20
Group	4,60	1,92	1,25	1,71

Table 7.8: Average number of times, per session, that our agent got challenged and made his opponents lose a card from it, in both conditions

statistically significant. This was needed to be tested so that our next tests and results would not be influenced by preconceptions from our participants and with that make the results as real as possible for a sample that is as constant as possible in terms of trust in themselves and others.

Continuing with the results obtained from the Godspeed Questionnaire, where the main objective was to see if the perception towards our agent and the way it acted and interacted would change depending on the different lying conditions, neither of them had a significant difference between these two conditions, nevertheless, the results have significant meaning to them to be reported, as they show in a more global scale the real difference between the perceptions of the participants towards both the lying conditions (lie and truth) and both the group type conditions (individual and group).

To begin and to show the biggest difference, we can see that all of the measures of perceptions that we captured from the participants have an increase from the Group condition to the Individual condition, being the Likeability measure the one with the biggest increase for the Lie condition, an increase of perception from 3,08 to 3,56, or in other words, and increase of 10% in a 5-point Likert scale, which in itself represents that participants perceived our agent to be 10% more likeable in the Individual condition than in the Group condition. From the differences in the group type conditions, we can deduce that by being in a group of more people, the spotlight of interaction is more easily removed from our agent, which then gets a lower score in terms of how well it was perceived by the participants. Other conclusion that we can reach based on these results is that, given that the game is played by multiple players and since the game also has in it player exclusion, by being removed earlier from the game, our agent's interaction, and consequently how it is perceived by the participants, diminishes as he will no longer take turns and perform actions. Since in the Individual condition if our agent loses the game ends and he continues to play in the next new game with its opponent, without having time lost where he could not actively participate in the game.

Still in the results from the Godspeed Questionnaire, we can see differences between the Lie and Truth conditions. Focusing primarily in the Individual condition, as the differences are more significative, we can see that the participants in the Lie conditions perceived our agent as being more Anthropomorphic, Animate, Likeable, Intelligent and more dishonest based on the Credibility score, since the more score it has in the Credibility measure, the more dishonest its behaviour was. Taking this into account, we can come to the conclusions that by being capable of deceiving its fellow opponents, these perceived it as being more similar to the average human being and with that thought of it as more human, giving a better averaged score to every single measure in terms of this questionnaire.

In terms of the results obtained from the Trust scale specific for Human-Robot Interaction, we got a very similar percentage of truth in the Group condition for both Lie and Truth conditions, with around 60% trust from the participants towards our agent, this may be due to the fact that its interaction on this group type condition did not have the most impact on the participants, so they felt the same towards our agent int both the lying conditions. More interesting, even though not big enough to be scientific significant, is the difference between the Lie and Truth conditions in the Individual group type condition, having an increase of 7,53% of trust from the Truth condition to the Lie condition. This increase is really interesting since the participants knew at one point or another that our agent was capable of deceiving, so why would they attribute an higher score of trust to a deceiving agent? The reason to that is most likely the same for the increased interaction perception from the participants in this condition, the Individual Lie condition, which is the fact that the participants more easily identify themselves with our agent and consequently build a more trustful image of it, perceiving it as more human.

Going into the results obtained from the more direct questions asked, the ones that only needed direct processing to build the results, we will start with how well the participants perceived our agent of playing. This measure was score with a 5-point Likert scale, so the opinion of each participant may differ slightly even if they give the same score as another participant, nevertheless, the results obtained here are a little different from the results obtained in other of various variables we used, in the sense that the conclusions that we can take from them are completely different based on the group type condition. In the Individual condition, we can see an 8% increase on the perception towards how well our agent played when comparing from the Truth condition to the Lie condition, going from an average score of 3,67 to 4,07. On the other hand, in the Group condition, we can clearly see that the average score given has decreased from the Truth condition to the Lie condition, going from an average score of 4,00 to 3,80, which represents a 4% decrease. This small decrease comes most likely from the fact that when playing in a group, sometimes a good strategy is one where the player will not try to get ahead in the beginning of the game, or in other words, will not be a potential threat in the beginning, making itself go unnoticed and avoid being the target of the other players, and since our agent in the Truth condition tends to play in a more conservative way, as its actions are limited, its quite plausible for that to be the reason of such decrease in the perception of how well it played when going from the Truth to the Lie condition.

Going a little deeper towards how well our agent played in the different conditions, we can see the results we got from the in game victories itself. Looking at Table 7.6, we can see that in terms of the Individual group type condition, in the Truth condition our agent won almost half of the games, with a 49,00% win ratio. This percentage receives a considerate increase that, while not being scientifically significant, contributes enormously to our perception of how well the agent played. The increase is of 7,00% to reach a much better win ratio of 56%, which in other words means that our agent is actually more capable of winning at the game of Coup than a human player. In terms of the Group condition, the results are a little exchanged for both the Lie and Truth condition, as have been shown in the last measures, being it an average of 23,40% win ratio for the Lie condition and an average of 25,00% win ratio for the Truth condition.

The conclusions we can take from the win ratio for the different conditions is that in all of them, our agent is capable of playing at least on the same level of a human player. In the Lie condition for the Individual group type condition, our agent can even play slightly better than a human, while in all other conditions, specifically in the Truth condition for the same group type condition, it won practically half of the games it played, where there was only one opponent, so the human players got the other half of games won, an in both the Lie and Truth conditions for the Group group type condition, the percentage of games won is close, on average, to one quarter of games won, which is actually the same average win ratio for each of the human players, since there are more three player when not counting with our agent. These are quite good results that we achieved, for both our lying and truthful agent, since they both use our Decision Making algorithm, we can prove that it is indeed possible to make an agent, that by implementing it on a social robot, is capable of playing at the same level of humans.

To see how having both the Truth and Lie conditions would influence the amount of times that our agent got challenged, we counted all of the times where a challenge was presented to our agent. The results, as mentioned in the above subsection, can be seen in the Table 7.7, and as can be seen, the amount of times that our agent gets challenged increases significantly from the Truth condition to the Lie condition, with a 4 times increase in the Group condition. This is due to the fact that in the Group condition is easier to make actions that may seem deceiving and with that, participants will label our agent as being capable of deception and then start to challenge it. Regarding the Individual condition, the increase while not being as significative as the one in the Group condition, is still a 50% increase. This leads us to conclude that the participants in general, knew that our agent in the Lie condition was more deceitful than our agent in the Truth condition, and with that, they challenged it more.

Finally, the last results that we analyzed were the average number of times per session that our agent when challenged, had the card to actually produce the action that he got challenged on. The results given in the Table 7.8 show exactly that, with our agent having relatively the same amount of times that it won in the Individual condition when challenged, but having a significant increase from the Truth to the Lie condition in the Group condition. As mentioned before, having more people that detect that it is capable of deceiving, will make the amount of challenges he receives increase, and by capitalizing on it, by starting to play a little more truthfully, our agent is capable of winning a great amount of challenges that it receives. This on the other side can have some repercussions, as the other players start to lose their cards, they will start to consider our agent as a potential threat, since it was our agent that made them lose their cards and consequently being close to losing, justifying why the percentage of won games for our agent in the Group Lie condition is inferior to the one of the Group Truth condition.

7.4 Conclusions

With this experiment we were able to take a number of conclusions about our agent's effectiveness on playing the deception game that is Coup, how its interaction was perceived by the participants and how much trust they put on it.

We also came to the conclusion that our agent is indeed capable of playing at the same level of humans and even play slightly better than them if used with the capability of deception and playing against a single opponent. This proves that our initial hypothesis is correct, which in itself is a great achievement.

Finally, something that we could have certainly have taken a more appropriate consideration would be the way that our agent interacted with its human opponents. While it was not the main focus of this work, it would have provided us with more data and more conclusion towards the differences of an agent that deceives and interacts taking that into account and an agent that only acts and interacts truthfully. The way we had arranged our interaction was a more general approach that worked in both lying conditions, which made us lose a bit of potential towards a better interaction.

Chapter 8

Conclusions

In this document we started by presenting some background on Deception, Board Games and the COUP game, with an extensive explanation on how the game works, in order to provide some comprehension towards some topics that are not usually directly linked to the computer science area. We then presented some of the most important works that relate to our own work's state if the art and consequently contribute to the ideas developed here. These works range from agents that can act deceptively in a purely digital environment, to agents that were implemented on robotic entities and were capable of deception. Other works that were focused simply tried to prove that a robot can actually deceive a human being and not just another robot.

We proceeded to define both the problem that surrounds our hypothesis and the way to solve it, just focusing on the components needed to do so in a more general and abstract way. After having the problem and solution defined, we presented the decision making algorithm that would be the base of the action selection done by our agent, before that, we introduced the algorithm that was the base for our algorithm and why minimizing counterfactual regret would help us solve our problem and successfully prove our hypothesis. The presentation of our agent architecture plus all other components needed to make the experiences to prove our hypothesis was done in a chapter specifically for just the implementation of our solution. In this chapter, the general solution that we presented before was specified into a real solution that we could effectively use in the real world.

The experiment done was then presented and included all the information regarding the sample we used, all its procedure, the variables that we used to measure different things, the results we obtained from analyzing the data acquired during it, a discussion towards the results we got, taking some conclusions from most of the results and finally concluding with what was the most interesting result we got from the experiment and what could we have done better to obtain more significant results towards the interaction between our agent and the participants.

In the next couple sections we will be concluding this work with the achievements we reached, based on the results and this work's objective, and finally we will present some suggestions for future work that can be done on this work's topic.

8.1 Achievements

Our initial objective for this work, and as presented in the introduction, was to address the following question: "Is it possible to create an agent that, by implementing it on a social robot, is capable to play the Coup game at the same level of humans?". Being this our grand objective, we have developed an entire system, including a powerful decision making algorithm, that would then be able to prove it.

Going through the results we got from our experiment, we were capable of coming to a considerable numbers of conclusions that provided a great amount of insight towards the use of deception by an agent that is physically represented by a robot and that intends to communicate with human beings. One of those conclusions was that humans actually consider our agent more trustful when he is capable of deception, which is due to the fact that they more easily identify our agent with themselves and consequently deposit more trust on it. This should be something to take into consideration when building agents that are required to interact with humans and that also require a significant amount of trust.

The result that was expected to give the most feedback to the way our work was going was definitely the percentage of games won by our agent against the human players. As already mentioned before, our agent was capable of achieving a 50% win ratio against a single human opponent and a 25% win ratio against three other human players, which in itself means that our agent was playing at the same level of humans. This completely proves our hypothesis, which is the greatest achievement that this work could get, a successfully proven hypothesis. Not only could our agent play at the same level of humans, but by using the unrestricted decision making algorithm, the one that is capable of using actions that were lies, our agents was capable of playing at a slightly higher level than its human opponents.

This proves that artificial intelligence is capable of producing behaviours that are mostly linked to the human behaviour and are not commonly associated with the agent world, which is a nice step up for the world of artificial intelligence.

Still, it is little disappointing that we could not reach results, regarding the differences between our agent that could lie and our agent that could not, at the point that they were big enough to be considerable for scientific purposes. Nevertheless, our initial objective was met and a substantial amount of new ideas was developed so that future experiments and developments can achieve the results that we could not now.

8.2 Future Work

Having reached our objective for this work does not mean that this research project has reached its end. There is still a big amount of possible additions and developments that this research can take to reach new levels of discoveries. In this section we will list the multiple directions and potential developments that would make this project benefit from.

8.2.1 Different interactions for the different agents

One of the things that we could have improved during our testing phase was the way our agent interacted with its human opponents. Since the main focus of this work was to see if it was possible for an agent, that being represented physically by a robot, was capable of playing at the same level of humans, we tended to put more effort and time into the development of the Decision making algorithm than into other things, which in turn made us neglect a little the interaction that our agent would make.

The interaction we used for our agent was a more general one, in other words, the interaction was built so that both our agent in the Lie condition and our agent in the Truth condition would be capable of using it without problems, this in time led to a more truthfully interaction than we would have wanted. This would be good if we only had the agent in the Truth condition as the interaction, while still being able to be improved, only allowed for truthful utterances. But since we also had an agent capable of deceiving through actions, we should make its interaction capable of deception too.

To allow for a more specific interaction for our deceiving agent, a good approach for it would be to use the Deception Planner mentioned in Chapter 3 Related Work, since the objective of the Deception Planner is to create utterances that would change the way that our target thinks in a way that would benefit the user of the Deception Planner. By adding it to the way our agent interacts and by coordinating it with the way our agent plays, we could have an agent much more capable than the one we currently have, making it even possible for it to win against experienced players in deception games.

8.2.2 Better game User Interface

During our experiments, one issue with all our system was the user interface that we presented to the human participants. While our agent is not influenced by the graphical representation of the Coup game on the digital tabletop, its human counterparts certainly are.

To make the user interface more appealing to the human eye and more user friendly, we have compiled a number of possible improvements to it and will list them:

- **More colourful interface** - The interface we used, while having some colours to help identifying the different cards in the game, was still mainly white in its entirety, making it look a bit monotone and giving the illusion that it was a prototype of the final version of it. To increase the overall experience of the game for the human players, we can certainly add some background colours to our interface.

- **Flashy indicators** - One of the suggestions that the participants made towards our game interface was that sometimes they did not really know what happened within the game, or if it was their turn or not. To make it easier for the players to capture this information, flashy indicators of turn and who has played and who can act can be added to our Coup game user interface.

- **Bigger delays between phases** - A problem that the participants had with our user interface was that the delay between some of the phases was non existent, which in turn made some participants choose a wrong action. An example of this is when there is an action that can be blocked, if a player presses almost at the same time that he wants to block the action as another player, the player that clicks on it secondly will be acting to challenge the block instead of making the block himself. To make this situation much harder to happen, adding a delay of at least 1 second between this kind of phases would be ideal.

- **Easier to choose action buttons** - Since our buttons to perform the actions were really close together, some of the participants wrongly chose the action that they wanted, making an action that they did not choose in the first place. To improve the selection of actions, bigger, more separate buttons for each action can be added, and if space is lacking in the user interface to accommodate this change, a confirmation button can be added so that the player is completely sure that the action he will be making is the action he actually wants.

These are some of the possible improvements that can be done to our game interface and there are probably more ways to increase the user experience while playing with our agent.

8.2.3 Stronger Theory of Mind component

One of the things that made our agent a little lacking was the way it perceived its human opponents. The theory of mind component was still existent in our agent architecture and our decision making algorithm was highly dependant on the values produced by it, still, since this component was just a simple version that would update what the agent knew about its opponents with simple rules, such as, when the agent got challenged on a action that used a particular action and from that lose a card, he would not make an action that would require such character again until at least he had exchanged cards, as it does not make sense to make.

This improvement to this project is already being developed and should be something to look forward upon, as the improvement it will provide should make our agent capable of reasoning about how its opponents play, predicting not only the actions they will make, but also the character cards they are using. This along with a decision making algorithm that minimizes regret, should make our agent a very capable Coup player.

8.2.4 Increased Bluffs

Our current player is lacking some of the features of the more experienced players, which is the capacity to make bluffs that instead of increasing the agent's current value in the game, for example, by using the Tax action that provides him with 3 coins when he does not actually have the Duke character, will make its current value not as strong, but will provide with the opportunity of capitalizing it in the future, by planting a seed of false confidence in its opponent's mind. An example of this kind of bluff would be to use the Income action, that provides 1 coin, or the Foreign Aid action, that provides 2 coins, instead of using the Tax action, which provides 3 coins, while having the Duke character card. This would plant the false knowledge of our agent not having the Duke card in our opponent's minds, making it really easy for them to challenge our agent when he will make an action that actually requires the Duke character card.

To achieve this feat, some kind of predictive theory of mind component should be added, since to be able to do this, our agent has to be able to predict its opponents mind based on actions that he has not yet made. A way to include this into this project would be to make our decision making algorithm used along with a planner that would plan the actions our agent would make so that it could win. This would provide a way for the theory of mind component to also be included in the planning and with that, be capable of doing this kind of high level deception, that only experienced players are able to do.

Bibliography

[1] J. Bond, CharlesF. and M. Robinson. The evolution of deception. *Journal of Nonverbal Behavior*, 12(4): 295–307, 1988. ISSN 0191-5886. doi: 10.1007/BF00987597. URL `http://dx.doi.org/10.1007/BF00987597`.

[2] B. M. DePaulo, D. A. Kashy, S. E. Kirkendol, M. M. Wyer, and J. A. Epstein. Lying in everyday life. *Journal of personality and social psychology*, 70(5):979, 1996.

[3] D. L. Cheney and R. M. Seyfarth. *Baboon metaphysics: the evolution of a social mind*. University of Chicago Press, 2008.

[4] M. D. Hauser. Costs of Deception: Cheaters are Punished in Rhesus Monkeys (Macaca mulatta). *Proceedings of The National Academy of Sciences*, 89:12137–12139, 1992. doi: 10.1073/pnas.89.24.12137.

[5] P. Ekman and W. Friesen. *Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues*. Malor Books, 2003. ISBN 9781883536367. URL `https://books.google.pt/books?id=TukNoJDgMTUC`.

[6] M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski. Measuring facial expressions by computer image analysis. *Psychophysiology*, 36(02):253–263, 1999.

[7] M. Flanagan. *Critical Play: Radical Game Design*. MIT Press, 2009. ISBN 9780262062688. URL `https://books.google.pt/books?id=-VXH43WH5Z4C`.

[8] R. S. Siegler and G. B. Ramani. Playing linear number board games—but not circular ones—improves low-income preschoolers' numerical understanding. *Journal of Educational Psychology*, 101(3):545, 2009.

[9] J. Wilde. The effects of the let's get rational board game on rational thinking, depression, and self-acceptance in adolescents. *Journal of Rational-Emotive and Cognitive-Behavior Therapy*, 12(3):189–196, 1994. ISSN 0894-9085. doi: 10.1007/BF02354596.

[10] F. De Rosis, V. Carofiglio, G. Grassano, and C. Castelfranchi. Can computers deliberately deceive? a simulation tool and its application to turing's imitation game. *Computational Intelligence*, 19(3):235–263, 2003.

[11] V. Carofiglio, F. de Rosis, and C. Castelfranchi. Ascribing and weighting beliefs in deceptive information exchanges. In *User Modeling 2001*, pages 222–224. Springer, 2001.

[12] C. Castelfranchi, R. Falcone, and F. De Rosis. Deceiving in golem: how to strategically pilfer help. Citeseer.

[13] D. Christian and R. M. Young. Strategic deception in agents. Master's thesis, Graduate Faculty of North Carolina State University, 2004.

[14] A. R. Wagner and R. C. Arkin. Acting deceptively: Providing robots with the capacity for deception. *International Journal of Social Robotics*, 3(1):5–26, 2011.

[15] C. F. Bond Jr and M. Robinson. The evolution of deception. *Journal of Nonverbal Behavior*, 12(4):295–307, 1988.

[16] M. J. Wooldridge. *Reasoning about rational agents*. 2000.

[17] D. Ward and H. Hexmoor. Towards deception in agents. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1154–1155. ACM, 2003.

[18] J. Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[19] K. Terada and A. Ito. Can a robot deceive humans? In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 191–192. IEEE, 2010.

[20] J. Dias, R. Aylett, H. Reis, and A. Paiva. The great deceivers: Virtual agents and believable lies.

[21] S. Baron-Cohen. *Mindblindness: An essay on autism and theory of mind.* MIT press, 1997.

[22] M. Harbers, K. v. d. Bosch, and J.-J. Meyer. Modeling agents with a theory of mind. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 217–224. IEEE Computer Society, 2009.

[23] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2007.

[24] M. J. Osborne and A. Rubinstein. *A course in game theory*. 1994.

[25] J. Dias, S. Mascarenhas, and A. Paiva. Fatima modular: Towards an agent architecture with a generic appraisal framework. In *Emotion Modeling*, pages 44–56. Springer, 2014.

[26] J. Kedzierski, R. Muszyński, C. Zoll, A. Oleksy, and M. Frontkiewicz. Emys—emotive head of a social robot. *International Journal of Social Robotics*, 5(2):237–249, 2013.

[27] T. Ribeiro, A. Pereira, E. Di Tullio, P. Alves-Oliveira, and A. Paiva. From thalamus to skene: High-level behaviour planning and managing for mixed-reality characters. In *Proceedings of the IVA 2014 Workshop on Architectures and Standards for IVAs*, 2014.

[28] M. Lee. The ethics of deception: Why ai must study selfish behaviour. In *Procs. AISB'00 Symposium on AI, Ethics and (Quasi-)Human rights*, 2000.

[29] S. D. Gosling, P. J. Rentfrow, and W. B. Swann. A very brief measure of the big-five personality domains. *Journal of Research in personality*, 37(6):504–528, 2003.

[30] K. Carrington. Toward the development of a new multidimensional trust scale. 2007.

[31] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International journal of social robotics*, 1(1):71–81, 2009.

[32] R. Ohanian. Construction and validation of a scale to measure celebrity endorsers' perceived expertise, trustworthiness, and attractiveness. *Journal of advertising*, pages 39–52, 1990.

[33] K. E. Schaefer. *The perception and measurement of human-robot trust*. PhD thesis, University of Central Florida Orlando, Florida, 2013.