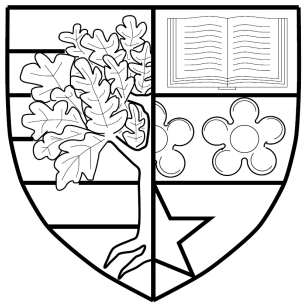
Reinforcement Learning for Trading Dialogue Agents in Non-Cooperative Negotiations.

Doctoral Dissertation by

Ioannis Efstathiou



Submitted for the Degree of Doctor of Philosophy in Computer Science

Interaction Lab

School of Mathematical and Computer Sciences

Heriot-Watt University

2016

"You can discover more about a person in an hour of play than in a year of conversation."

PLATO, 429 - 345 B.C.
LINGARD, 1598 - 1670
ANONYMOUS

Declaration

I, Ioannis Efstathiou

hereby declare that the composition of this thesis submitted for examination has been made by myself and the words are personally expressed. Any exceptions, including works taken from any other authors, are all stated in my text and are also included in my references list. Furthermore, this document has not been submitted for any other qualification or degree.

Abstract

Recent advances in automating Dialogue Management have been mainly made in cooperative environments -where the dialogue system tries to help a human to meet their goals. In non-cooperative environments though, such as competitive trading, there is still much work to be done. The complexity of such an environment rises as there is usually imperfect information about the interlocutors' goals and states. The thesis shows that non-cooperative dialogue agents are capable of learning how to successfully negotiate in a variety of trading-game settings, using Reinforcement Learning, and results are presented from testing the trained dialogue policies with humans. The agents learned when and how to manipulate using dialogue, how to judge the decisions of their rivals, how much information they should expose, as well as how to effectively map the adversarial needs in order to predict and exploit their actions. Initially the environment was a two-player trading game ("Taikun"). The agent learned how to use explicit linguistic manipulation, even with risks of exposure (detection) where severe penalties apply. A more complex opponent model for adversaries was also implemented, where we modelled all trading dialogue moves as implicitly manipulating the adversary's opponent model, and we worked in a more complex game ("Catan"). In that multi-agent environment we show that agents can learn to be legitimately persuasive or deceitful. Agents which learned how to manipulate opponents using dialogue are more successful than ones which do not manipulate. We also demonstrate that trading dialogues are more successful when the learning agent builds an estimate of the adversarial hidden goals and preferences. Furthermore the thesis shows that policies trained in bilateral negotiations can be very effective in multilateral ones (i.e. the 4-player version of Catan). The findings suggest that it is possible to train non-cooperative dialogue agents which successfully trade using linguistic manipulation. Such non-cooperative agents may have important future applications, such as on automated debating, police investigation, games, and education.

Acknowledgements

Initially I would like to thank my first Supervisor, Professor Oliver Lemon, who offered me the chance to work in the intriguing areas of Reinforcement Learning and Pragmatics. He motivated me to study the strategic negotiations that can be learned by intelligent agents in Natural Language Dialogue systems, through the application of various interesting Reinforcement Learning algorithms and techniques. Without his expertise, patience, valuable guidance and experienced suggestions this document would lack content and aim. Each of our meetings was a unique experience which not only enhanced my current knowledge but significantly widened my perspective on academic thinking.

My second Supervisor, Professor David Corne, who as my second mentor now and teacher in the past on several lessons, during my MSc course in Artificial Intelligence, elegantly guided me through various fascinating paths of this area and kept reinforcing my passion towards it through his ingenuity and broad experience. The Heriot-Watt University and its Library department, for providing me all the required knowledge and the flexible means respectively to effectively seek resources and produce this work. The postgraduate students Wenshuo, Aimilios and Nida who showed interest in our research by extending parts of our work now towards their unique goals. Furthermore, everyone involved in the Interaction Lab and especially in the STAC project for their suggestions in our meetings and valuable advice. Last but not least my family, partner and friends for their continuous support and deep understanding of my effort.

Contents

List of Figures

List of Tables

Abbreviations

ASR	Automatic Speech Recognition
BP	Building Plan
$BRLA$	Baseline Reinforcement Learning Agent
CP	Cooperative Principle
$CP-NET$	Conditional Preference Network
DM	Dialogue Manager
HCA	Hand Crafted Agent
HCI	Human Computer Interaction
$HMMDP$	Hidden Mode Markov Decision Process
HRI	Human Robot Interaction
LA	Learning Agent
LC	Locutionary Cooperation
MA	Manipulative Action
MDP	Markov Decision Process
NLG	Natural Language Generation
NLP	Natural Language Processing
$NPRLA$	Negative-Positive Preferences Reinforcement Learning Agent
OM	Opponent Model
PC	Perlocutionary Cooperation
PCP	Perlocutionary Cooperative Principle
$POMDP$	Partially Observable Markov Decision Process
$PPRLA$	Positive Preferences Reinforcement Learning Agent
RL	Reinforcement Learning
RLA	Reinforcement Learning Agent
$SARSA$	State Action Reward State Action
SLU	Spoken Language Understanding
TTS	Text To Speech

Chapter 1

Introduction

Remarkable work has been done in cooperative dialogue systems over the past years. Many interesting results bring human users even closer to computers - especially when it comes to interacting with them through speech. Research in automated conversational systems has almost exclusively focused on the case of cooperative dialogue though, where a dialogue system's main goal is to assist humans in specific tasks, such as buying airline tickets [109] or finding a place to eat [113]. Furthermore, apart from Human-Computer Interaction (HCI), impressive results have been reported in the field of interaction between intelligent agents when the task's goal is common and they learn how to collectively work towards it [103]. In all those cases the agents (artificial or human) try to maximize their combined utility in a cooperative environment [114].

1.1 Motivation

The complexity of dialogue increases when the interaction has a non-cooperative basis and especially when the information of the participants is imperfect. There are still many unexplored paths to be investigated and many models are still insufficient for that type of interaction [101]. Practical and theoretical interest [38] on non-cooperative dialogues has been shown, where an agent may act in order to achieve its own goals rather than those of its interlocutors. Some examples where it is beneficial for an automated agent not to be fully cooperative are: i) in an attempt to gather information from a human or another artificial agent, ii) when the agent is trying to persuade, argue, or debate, iii) when attempting to sell something, iv) in an effort of detecting illegal activity (for example on internet chat sites during police investigation), or v) in the area of believable characters in video games and educational simulations [38, 90]. A very important field in which non-cooperative dialogue behaviour is desirable is in negotiation [101, 77, 35], where hiding informa-

tion (and even outright lying) can be advantageous [26]. Deception is considered to be an essential part of successful military operations. According to Sun Tzu "All warfare is based on deception" and Machiavelli clearly states in *The Discourses* that "Although deceit is detestable in all other things, yet in the conduct of war it is laudable and honourable" [2]. Indeed, according to Dennett [19] deception is a required capability for higher-order intentionality in Artificial Intelligence.

In a language task modelled through the application of Decentralised Partially Observable Markov Decision Processes [103], it has been suggested that multi-agent decision theory triggers Gricean cooperative principles [39]. In another related work [104], agents maximised their joint utility using conversational implicature. Recently, methods drawn from the field of Machine Learning were applied in order to optimise *cooperative* dialogue management, where the decision of the next dialogue move to make in a conversation is on focus in an attempt to maximise an agent's overall long-term expected utility. That usually translates as meeting a user's goals [113, 84]. With the particular research it has been argued that robust and efficient dialogue management strategies can be learned from data, but only the case of cooperative dialogue was taken into consideration. Those Machine Learning methodologies used Reinforcement Learning (RL). As we will discuss in Section 2.1, RL uses a reward function that gives positive feedback to the agent when it meets a goal, which was that of the user in the above cooperative case.

The goal of the current work is to provide answers to questions originating from *non-cooperative* negotiation environments with imperfect information. In this case, distinct personal objectives usually do not promote the agents' mutual support but -in contrast- they build up competition. Furthermore, the players do not have a clear picture of their opponents' states or goals. Thus miscommunication, complication and even failure of the dialogue process is often observed due to a system's limited capabilities of identifying, managing and recovering from an unnecessary long series of non-cooperative dialogue acts [43].

We define as **linguistic manipulation** all those dialogue actions (utterances) that aim for personal gain. Those actions may be either deceptive (i.e. lies) or persuasive (i.e. via repetition of truth), and we will see later that deception and persuasion, which are based on the "nature" of the negotiator, might both serve manipulation. In the current thesis they are both learned and used by our RL agents, during trading negotiations, where the environment is non-cooperative. As we have mentioned earlier, that means each participant may act to satisfy his/her own goals rather than those of other participants because the goals diverge or conflict.

1.1.1 Example dialogues

A real dialogue example of this kind that we model, taken from the "Settlers of Catan" (nowadays known as "Catan") game corpus (STAC project, `http://www.irit.fr/STAC/`), is provided below:

1. - A: Do you have rock?
2. - B: I've got lots of wheat [in fact, B has a rock]
3. - A: I'll give you 2 clay for a rock
4. - B: How about 2 clay for a wheat? [B *insists* on offering wheat]
5. - A: I'll give 1 clay for 3 wheat
6. - B: Ok, it's a deal.

In this work, reinforcement learning algorithms mainly based on SARSA [97] have been developed, in an attempt to learn how to successfully confront adversarial trading strategies in non-cooperative environments. Thus at the beginning of this research, a simple, sequential, non-cooperative, non-zero-sum game with imperfect information was invented between two players called "Taikun", serving as a toy-model of real-world trading negotiations. An example of the type of non-cooperative dialogue behaviour which we are generating in our simple trading game "Taikun" is given by agent B in the following dialogue. It demonstrates the power of the explicit manipulation (implicature) through the "I really need X" utterances, where X is a particular resource:

1. - A: I will give you a sheep if you give me a wheat
2. - B: No
3. - B: I really need rock [B actually needs wheat]
4. - A: OK
5. - A: I'll give you a wheat if you give me a rock
6. - B: OK

Here, A is deceived into providing the wheat that B actually needs, because A believes that B needs rock rather than wheat or sheep.

After "Taikun", we worked in a more complex model (the game "Catan") that led us to a more advanced type of non-cooperative dialogue behaviour, where all of the trading proposals have manipulative effects (implicit manipulation):

1. - A: “I will give you a wheat and I need 2 clay”[A lies - it does not need clay but it needs wheat]
2. - B: “No”
3. - A: “I’ll give you a rock and I need a clay”[A lies again and it actually needs rocks too, but it does not have any rocks to give]
4. - B: “No”
5. - A: “I’ll give you a clay and I need a wheat
6. - B: “Yes”

Here, B is deceived into providing the wheat that A actually needs, because B believes that A needs clay (A asked for it twice) rather than wheat and rock (that it offered). Similar human behaviour can be observed in the Catan game corpus [1]: a set of on-line trading dialogues between humans playing Settlers of Catan. We analysed a set of 32 logged and annotated games, which correspond to 2512 trading negotiation turns. We looked for explicit lies, of the form: *Player offers to give resource X (possibly for Y) but does not hold resource X* - such as in turn 3. in the above example. 11 turns out of 2512 were lies of this type[1]. Since this corpus was not collected with expert players, we expect the number to be larger for more experienced negotiators. Other lies such as asking for a resource that is not really wanted, cannot be detected in the corpus, since the player’s intention would need to be known.

In the experiments that will follow we will also see that a Reinforcement Learning Agent (RLA) improves its performance by maintaining the adversarial preferences in its state representation, using different models for that reason. Considering the history of the trades (during the dialogue), the RLA manages to significantly outperform one which does not do that. We will also evaluate our policies with humans and show that manipulation is effective against them. We will conduct experiments with agents which detect manipulation based on logical contradictions in dialogue and show that learned RL policies are still capable of winning more often than these adversaries, highlighting the importance of such detection for our society and demonstrating that these adversaries are harder to manipulate though. Finally we will transfer and test our learned policies from bilateral to multilateral negotiations of the multi-agent game “Catan”. We will see that trained RL is still very successful there, even if it is trained versus one opponent, as it then treats them all as one.

[1]Unfortunately, information regarding the result was not available.

1.2 Ethical issues in Robotic deception

Nowadays robots are considered to be a closely related subject to our human society. Human Robot Interaction (HRI) rapidly progresses but various parties have already expressed their worries about their potentially harmful behaviour, such as even an impending revolution [55, 89]. It is crucial to analyse the term deception though that artificial intelligent agents are capable of using, not only with its possible dangers but with its benefits to the society too. According to Bond and Robinson [9]: "deception simply is a false communication that tends to benefit the communicator". However, when the benefit of the communicator harmlessly serves a personal goal and even matches that of a greater good then it is considered to be a virtuous act. According to the current outcome of this thesis and the work of other researchers [33] [107], robotic deception is currently amongst us, it can be established between robots as well as between a robot and a human, but this should be achieved for at least a harmless -if not a solely righteous- purpose. It certainly deserves our immediate attention and is already under criticism due to its potential nefarious uses. However, as this criticism is still on early stages, Shim and Arkin [90] recently attempted to classify the robotic deception and to provoke deeper consideration on this field in order to investigate both of its aspects [2].

The research in this particular area is slow-paced and a few but significant steps have been made so far from a behavioural angle. Floreano et al [33] suggest that deception is evolutionary developed in heterogeneous individual robots in their attempt to secure their own "food" source. Wagner and Arkin, in their attempt to examine when [107] a robot should deceive and how [108][2] through the use of games theory and interdependence theory, provide insightful results with the creation of an algorithm that identifies social situations that confirm the need of robotic deception. It focuses on the appropriate actions to be taken, beliefs and fitting communication but from a sociological perspective rather than conversational that our research is based on. Hence partner modelling through the creation of dependent and independent outcome matrices and theory of mind played a significant role in their work.

The trust between a robot and a human [106] is an essential element of cooperation. Research in automated conversational systems, that this work studies, has almost exclusively focused on the case of cooperative dialogue as we have discussed. However, non-cooperative dialogues such as those that are produced through this work, where an agent may act to satisfy its own goals rather than those of other participants, are also important. It would be advantageous for the society if automated agents were capable of manipulating and detecting illegal activity in a chat room for instance, during police investigation, through non-cooperative negotiation.

[2] *When* and *how* to deceive (i.e. manipulate) where questions that "followed" us throughout our whole work as we will see from Chapter 3 onwards.

Deception may be born of pre-existing trust. Humans have an innate need of showing trust, of being cooperative [39]. That suggests though that they are also in danger of being deceived for nefarious purposes, and therefore deception detection is also a significant matter. Related work based on that has been made in Chapter 5 (and especially in Section 5.8) of the thesis, as deception and its detection should be both used for the society's well being. One of the main goals of this research is to demonstrate though that deception in non-cooperative environments can effectively be (and should be) learned by artificial intelligent agents, providing that its intentions are harmless for the humans and -even better- aim for the common good.

1.3 Research Questions

This work has developed Reinforcement Learning (RL) algorithms that learned how to form successful opposing strategies according to those of their adversaries. The goal was to negotiate with the adversaries optimally in non-cooperative trading scenarios. In detail, the RL agents learned how to manipulate their adversaries by persuading, hiding information, lying, bluffing and mapped their needs in a way that allowed exploitation. The development of this idea started by verifying the following simple questions:

1. Reinforcement Learning: Is it possible to create a Reinforcement Learning algorithm based on Temporal-Difference that can learn how to win in a simple non-cooperative game with perfect information (such as "tic-tac-toe")?

2. Reinforcement Learning: Will the above learning procedure require the same amount of training games with that of other successful methodologies? How successful is this algorithm?

After that we proceeded by attempting to provide precise answers to the following -more sophisticated- questions:

1. Reinforcement Learning: Will the above algorithm be able to also model successfully an adversary in a Markov Decision Process (MDP) environment of a more complex, non-cooperative game with imperfect information such as poker? Or do we have to follow other routes (such as POMDPs) in order to efficiently handle this partial information only? How successful can an MDP framework be for a bilateral non-cooperative negotiations environment?

2. Game Theory: Which is the most simplified, strategically functional and promising trading game that we can use in order to achieve non-cooperative trading negotiation? Do we need to perhaps create one by ourselves for the sake of time by avoiding unnecessary complexities of rules and mechanisms that known games have and do not fit exactly to our interests?

3. Game Theory & Pragmatics: By deciding to invent a new trading game, how can we fulfil all of the above set of requirements (simplified, strategically functional and promising for our researching needs) efficiently? What kind of rules does this game need to have in order to promote the trading element between two artificial intelligent agents up to a specific desirable point? Can interesting[3] dialogue policies be formed by playing this game?

4. Pragmatics: With the assumption that we have already implemented a promising learning algorithm and a trading game that has the potential to answer precisely our scientific questions, *what kind* of dialogue policies do the adversaries need to follow in order for us to learn how to produce interesting opposing ones?

5. Pragmatics & Reinforcement Learning: Would our learning algorithm be able to learn how to use linguistic manipulation to beat its opponents -through deception for example? Would manipulation provide an advantage to its strategy? How should the linguistic manipulation be expressed and used by a RL dialogue agent?

6. Reinforcement Learning: By assuming that manipulation is advantageous and a dialogue agent can successfully learn how to use it, how can we detect it? Is it possible to develop agents which are able to detect it?

7. Pragmatics & Philosophy & Psychology: What are the possible ways to manipulate through discourse and when is the best moment to do that? Can a RL dialogue agent learn how and when to do that in a non-cooperative trading negotiation?

8. Reinforcement Learning: How can we optimize the Reinforcement Learning algorithms that we use in order to reduce their running times and increase their performances?

9. Reinforcement Learning: Given that a RL dialogue agent maintains the history of its opponents' moves in its state representation, is it possible to increase its trading performance? What kind of methods can we use in order to capture the adversarial preferences effectively?

10. Reinforcement Learning: Will our RL trading dialogue policies be successful in multilateral non-cooperative negotiations? Can we learn policies in bilateral negotiations which will be also successful in multilateral negotiations of the same domain? The reason would be to avoid learning unnecessary information

[3]Interesting for the areas of Pragmatics mainly, as well as Philosophy, Game Theory and Psychology.

(noise) from the multilateral negotiation that would affect the RL agent's trading performance.

11. Reinforcement Learning: Will our trained RL dialogue policies (especially the manipulative ones) be successful against humans?

12. Reinforcement Learning: How can we generalise our non-cooperative dialogue agents to successfully negotiate with other agents or humans in real-world trading negotiation scenarios? Will our findings be applicable to trading negotiations only or to other non-cooperative negotiation domains too?

Overall, the main research question that the thesis pursued is:

- Can dialogue agents learn how to successfully trade in non-cooperative negotiations?

1.4 Requirements

There is a unique, required target from this work as its title suggests: effective Reinforcement Learning of non-cooperative dialogue management for trading agents. Various paths were investigated in order to achieve that, involving elements drawn from different fields such as Philosophy, Psychology, Game Theory, Dialogue Management, Machine Learning and Pragmatics. Hence effort was put to initially understand in depth different aspects of those fields that would eventually allow us to effectively compose our learning algorithms piece by piece. The first stage was expected to develop algorithms that would be able to confront different adversarial strategies in non-cooperative trading games with imperfect information. The second stage was expected to teach to the intelligent agent how to manipulate (e.g. deceive) its opponents, enriching its language by using Natural Language efficiently (with focus on Pragmatics), according to the rules of those games. The third step would eventually generalise our agent(s) by reaching a higher and more complex level, resulting to a non-cooperative manipulative trading negotiator which would become capable of successfully negotiating in real-world scenarios. The interlocutors would be either intelligent agents which take human decisions or (even better) humans themselves. As we will see in detail through this thesis, most of our requirements were met and new, unexpected challenges arose.

1.5 Contributions

Here we will discuss the contributions that this work has led to. The reader may also want to refer to Section 11.1 where these contributions are listed thematically. The present thesis investigates the capabilities and limits of traditional tabular Reinforcement Learning, as in a Markov Decision Process (MDP) framework, in games where there is hidden information about the adversarial goals, states and preferences. Despite the fact that the imperfect information suggests the use of a partially-observable Markov Decision Process (POMDP), and much work has been made on that area, our results suggest that an MDP is capable of successfully dealing with hidden knowledge (to some extent) in our case. That occurs even in the cases where the environment's dynamics continuously change, therefore making the problem a non-stationary MDP [17, 91, 36], as we will see in most of our experiments due to the dialogue actions and the game rules. We will also show that the normal trading actions have a stochastic effect (that of a possible trade) and a deterministic one (that of linguistic manipulation), as we will examine in detail in Section 8.2, resulting to non-stationary MDP dynamics and corresponding successful policies [28].

Our agents also show that through Reinforcement Learning in an MDP they are able to learn how to efficiently use explicit manipulation [25, 26] in Chapters 4, 5 and implicit manipulation in Chapter 8 (either through deception or persuasion [28] in Section 8.2) as well as Chapter 9 (where we evaluate the deceptive policy in a multi-agent environment), to improve their performance. With the realistic assumption that a trading adversary can be affected by implicature, and it can demonstrate a hindering trading strategy (i.e. boycott) against an agent which insists on particular trades, such agents learn how to successfully manipulate even when there are severe penalties associated with risks of exposure [27] (Sections 5.4.1 and 5.4.2). Adversaries which detect deception based on logical contradictions are harder but still able to be manipulated by our algorithms [105] (Section 5.8).

In the field of discourse studies and Philosophy, our Reinforcement Learning agents bring an important argument of Van Dijk [20] to light, according to which there is an everyday conventional inference of dishonesty from manipulative acts. That negative effect cannot be taken for granted though as manipulation according to Dillard and Pfau [21], as well as O'Keefe [78] also occurs through legitimate persuasion. This is what our Reinforcement Learning work suggests too with the similar performances of our deceptive and persuasive agents [28] in Section 8.2. Hence we emphasise the significance of Attardo's perlocutionary cooperation [6] through the agents' dialogue actions, that we examine in Section 3.1.3.

We also present a novel way of encoding the state space of trading dialogues that reduces its size to 0.5% of the original [28] in Section 7.2.5. Hence it reduces

the training times dramatically and overcomes problems related to high memory demands in the traditional tabular Reinforcement Learning. This method automatically converts all of the numeric states to a significantly smaller number of states (compressed representation). It takes into consideration the distance from goal, the availability of the resource as well as its quality (goal or non-goal resource) and uses a few different characters. The performance of this method is nearly as high as that of the tabular representation.

This thesis also shows that trading dialogues are more successful when the learning agent builds an opponent model – an estimate of the (hidden) goals and preferences of the adversary – and learns how to exploit them in Section 8.3. Conditional preference networks (CP-NETs in Section 8.4) [11] have been implemented and used in our agents' state representation resulting in higher performances than those of agents which did not include them [29]. In Chapter 9, our previous RL trained policies played games against agents in a multi-agent environment where there were 3 opponents and their strategies varied. Results there showed that the policies were capable of winning significantly more games than their adversaries, and the manipulation was successful, even if they were trained in a different, simpler, bilateral negotiations environment.

Manipulation was not effective only against rule-based agents, but against Bayesian agents too which simulated human behaviour (Chapter 9). In the same chapter, the adversarial preference representation of the trained RL policies that were used there resulted to decent performances too. Finally, important information about human play and negotiation reasoning, was collected in games that humans played in our game "Taikun" (Chapter 6), against our trained RL agents. The agents managed to win half of the games and the manipulation affected the human players through confusion. In Chapter 10 humans played our game "Trading in Catan" versus our trained RL dialogue policies and lost most of the games. We show there that RL dialogue agents can be very successful in non-cooperative negotiations against humans. The manipulative agent's policy was effective too, resulting to nearly the same number of wins as those of the goal-oriented (non-manipulative) agent's policy, in the same number of rounds[4].

[4]The number of rounds was the same for both the manipulative and the non-manipulative agents. Despite the fact that the manipulative agent is not trying to reach the goal numbers of resources as soon as possible (in contrast to the non-manipulative goal-oriented agent), because it uses manipulation (e.g. lies) and therefore it "wastes" rounds for this, it still wins nearly the same number of games.

1.6 Publications

This work has resulted in the following publications:

1. Efstathiou, I. and Lemon, O., 2014. Learning non-cooperative behaviour for dialogue agents. Proceedings of the 21st European Conference on Artificial Intelligence (ECAI). Prague, Czech Republic, pp.999-1000. (results from Chapter 4 of this thesis)

2. Efstathiou, I. and Lemon, O., 2014. Learning non-cooperative dialogue behaviours. Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL). Philadelphia, PA, U.S.A, pp.60-68. (results from Chapter 5 of this thesis)

3. Efstathiou, I. and Lemon, O., 2014. Learning to manage risks in non-cooperative dialogues. Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2014 - DialWatt). Edinburgh, Scotland, pp.173-175. (results from Chapter 5 of this thesis)

4. Vourliotakis, A., Efstathiou, I. and Rieser, V., 2014. Detecting Deception in Non-Cooperative Dialogue: A Smarter Adversary Cannot be Fooled That Easily. Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2014 - DialWatt). Edinburgh, Scotland, pp.252-254. (results from Chapter 5 of this thesis)

5. Efstathiou, I. and Lemon, O., 2015. Learning non-cooperative dialogue policies to beat opponent models: "The good, the bad and the ugly". Proceedings of the 19th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2015 - GoDial). Gothenburg, Sweden, pp.33-41. (results from Chapter 8 of this thesis)

6. Efstathiou, I. and Lemon, O., 2016. Learning better trading dialogue policies by inferring opponent preferences. **To be presented**: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016). Singapore. (results from Chapter 8 of this thesis)

1.7 Thesis outline

In this document, along with relevant work of various researchers which is thoroughly studied in the Background Chapter 2, the reader will find information about

the design of our initial model, the game "Taikun" in Chapter 3. Various experiments based on linguistic manipulation, manipulation detection, and the corresponding detailed results are covered in Chapters 4 and 5, while in Chapter 6, we investigate the human factor in the game. In Chapter 7 and 8 the reader will find the design of our main model, the game "Catan", along with corresponding experiments on opponent models (i.e. based on adversarial preferences) and their results which are presented there respectively. In Chapter 9 we discuss experiments and results mainly based on multi-agent manipulation and preference representation in the JSettlers research environment. Chapter 10 discusses human experiments on Catan. The range of topics that was covered will hopefully convince the reader that our expectations have been mostly met in Chapter 11, which is the Conclusion. In that chapter we also provide a personal evaluation of the produced work, we report possible future applications of our findings and we further discuss the vision of this work. After that the references of this thesis follow. Finally the reader will find the Appendix, including some of our preliminary work and more details of the various implemented algorithms.

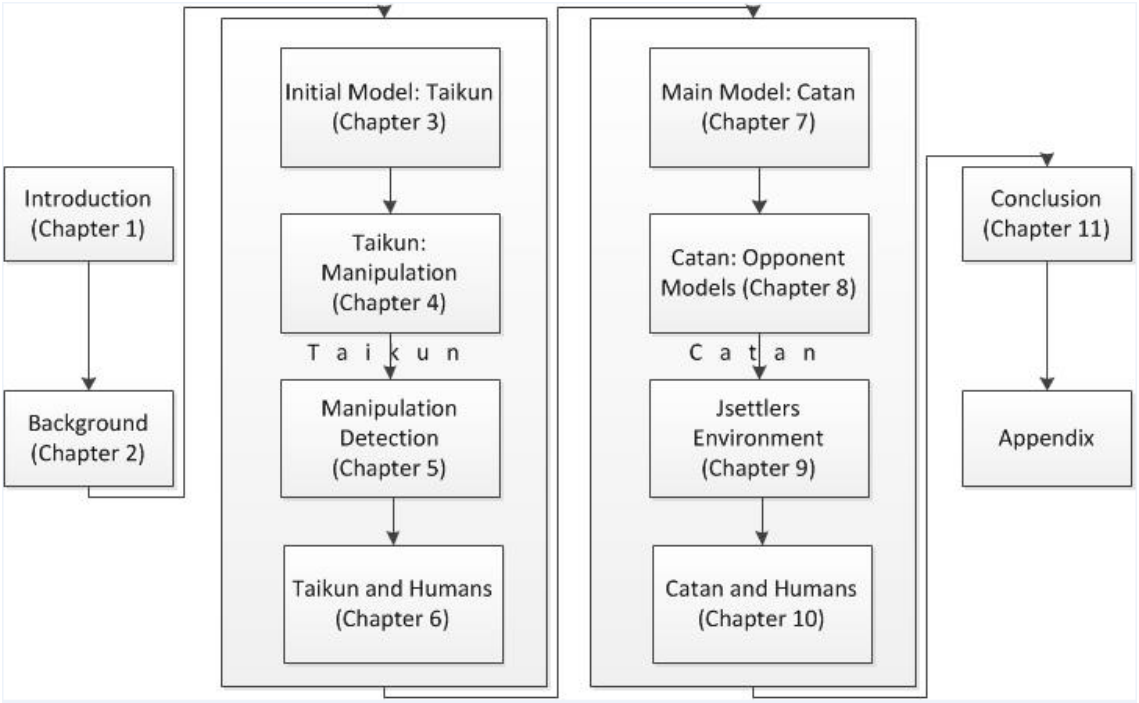


Figure 1.1: *The structure of the thesis.*

Chapter 2

Background

In this chapter we will examine in detail all those different areas that have motivated and inspired the current work. In Section 2.1 we will discuss Reinforcement Learning and analyse the important elements that directly relate to this work, such as exploration and exploitation in Section 2.1.1, Markov Decision Process (MDP) in Section 2.1.2, Temporal-Difference Learning in Section 2.1.5 and SARSA learning algorithms in Section 2.1.6. Section 2.2 describes the non-stationary nature of our MDP environment examining relevant work.

Game Theory in Section 2.3 has also been taken into consideration when we implemented our games ("Taikun" and our version of "Catan"). Hence we created environments that were suitable for our research and reflect real life scenarios. Sections 2.3.2 (Cooperative and Non-cooperative games), 2.3.3 (Nash Equilibrium), 2.3.5 (Dominant Strategy) and 2.3.6 (Perfect and Imperfect Information Games) have mainly inspired our work in Chapter 4, about our game "Taikun". However Game Theory was not the main component of this research. Reinforcement Learning in Dialogue Systems (Section 2.4), Reinforcement Learning in Non-Cooperative games (Section 2.4.1) and especially Reinforcement Learning in Negotiation Dialogue Management (Section 2.4.2) as well as CP-NETs (Section 2.5) and Pragmatics (Section 2.6) proved to be the main components of the final work. Thus the reader will find experiments based on these subjects from Chapter 4 onwards.

2.1 Reinforcement Learning

According to Sutton and Barto [98] the general concept of learning is naturally connected to the interaction with the environment. From its early stages, the human being interacts with the world and develops an understanding of the cause and effects of its own actions, and learns to make plans that provide it with particular results. This information, that is stored in the form of knowledge, is then carried over to

later stages of life and provides miscellaneous plans (policies) of acting in various circumstances (states).

Our behaviour is directly affected by the environment which –as a teacher- provides us with positive or negative rewards depending on our actions. In this way, a very active area of Machine Learning which is Reinforcement Learning, takes an agent through a series of different states and actions in order to eventually provide it with the optimal policy that dictates the most secure and effective (reward-wise) way of achieving its goal. This goal-oriented learning process is solely based on interaction with an uncertain environment through mathematical analysis. In Computer Science an intelligent agent is defined as an autonomous entity (i.e. a software program) that has the capability of learning through a computational approach as is discussed by Russell and Norvig [87]. Such an agent will be our main focus of this Reinforcement Learning study.

As we will discuss in detail in Section 2.1.2, the environment in RL is typically considered to be a Markov Decision Process and therefore it is fully observable. That means there is no hidden information that is taken into consideration by the learning agent. While learning, the agent is in a state s_t at a particular time point t. Through an action a_t that it performs, it traverses onto a new state s_{t+1} in the next time point $t+1$. It then receives a reward r_t from the environment at the particular state s_{t+1}. Simply stated, the "quality" (value) of the reward eventually defines that of the state's action as we will examine later and therefore we typically assign positive rewards to states which are desirable and negative ones to those which are not desirable. In this way, when the learning process stops, the agent by selecting the highest in value actions (i.e. those of the optimal policy) eventually traverses to the most desirable state. This is where the reward is the highest.

2.1.1 Exploration and Exploitation

In a Reinforcement Learning problem, a unique important factor that requires early consideration is the ratio between exploration and exploitation, also known as e-greedy behaviour. The learning agent with the use of exploration uncovers new aspects of the environment, by performing different actions that could potentially become the optimal for the currently studied learning problem in the future. Exploitation, on the other hand, assists the agent on maintaining a knowledge base capable of dictating to it the best (so far) action that should be taken (given a current state) in order to retrieve a reward and eventually reach its goal. In this way the agent avoids repetitive failures over sets of states-actions that have been reached before and learns the optimal policy [98].

The above dilemma between exploration and exploitation affects the time as

well as the quality of the Reinforcement Leaning process and therefore has been thoroughly studied by Tokic [100] who applied a probability in the agent's exploration based on the Temporal-difference error obtained from value-function records on a multi-armed bandit [56] application. Furthermore, Ishii et al. [53] introduced a balancing method between exploration and exploitation that was based on random actions variation and observations on environmental changes and performed well on maze tasks. The probabilities of the state transitions were approximated by Bayesian inference (including a "forgetting" factor) and the Reinforcement Learning process is based on that.

According to the work of Kearns and Singh [60] the trade-off between exploration and exploitation is of significant importance as the restriction of exploration could mean the lack of convergence to the optimal policy. On the other hand, the redundancy of the exploration could mean poor performance results in long periods of time. In order to successfully confront these issues they proposed a method that attempts a multi-step policy of actions when visiting an unseen so far state through a specified probability. This was in contrast to the typical view of e-greedy exploration at that time where with certain probability only a single action could be attempted. In this work we mainly use 20% exploration (and 80% exploitation) at the beginning of training, as we will see from Chapter 4 onwards, which is gradually decreased to 0% at the end of training (where there is 100% exploitation).

2.1.2 Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical framework where an agent performs a set of actions a over a space of states s, traversing through them in discrete time t. It is a fully-observable environment meaning that the agent's sensors provide access to every aspect of each state, on any given time point. It is important to note that in an MDP framework the next state is only dependent on the current state and action selected, and therefore conditionally independent from the history of the states so far. This is known as the Markov Property and therefore a state of a specific time point s_t is considered to be Markov if and only if:

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \ldots, s_t]$$

[81]

In Reinforcement Learning the environment is typically considered to be a Markov Decision Process and it is stochastic. Thus the state transition probability, for a current state s given its following state s', Pss' is defined as: $Pss' = P[s'|s]$. Hence we define the state transition matrix from each current state s to any other following

state s' as:

$$\mathcal{P} = \begin{matrix} \\ from \end{matrix} \begin{matrix} \quad to \\ \end{matrix} \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

Each row of the above matrix sums up to 1. A Markov decision process is a sequence of states that have the Markov property and is defined as a tuple: $< S, P >$ where S is a finite states set and P is a state transition probability matrix, $Pss' = P[s'|s]$ [81]. In general, the MDP can be defined as a 5-tuple: $< S, A, P, R, \gamma >$, where S is the set of states, A is the set of actions, P the probability that an action a will change a state s to s', R is the immediate reward that will be received after the state transition occurs and γ is the discount factor which indicates how important the future rewards are for our problem.

According to Mitchell [71], in a Reinforcement Learning setting the agent at any given time point t observes a state s, performs an action a, receives a reward r_t and the next state s_{t+1} follows. Based on the Markov property that we saw previously, the Markov Reward assumption is also: $P(r_t|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(r_t|s_t, a_t)$. Then the task of the agent is to learn a policy $\pi : S \to A$ where it chooses actions that maximise the expected reward: $E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots]$, where $0 < \gamma < 1$, over $P(r_t|s_t, a_t)$ and $P(s_{t+1}|s_t, a_t)$ for every initial state.

2.1.3 Policy Value Function

Following Mitchell's discussion [71], given a policy $\pi : S \to A$ we define as the value function for this policy π:

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

r_t is the immediate reward on a specific time point t and γ, where $0 < \gamma < 1$, is a discount factor that exponentially decreases the future rewards. However, the goal is to find the optimal policy π^*, where:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

Therefore there is an optimal value V^* of a state s of the optimal policy π^* that will be just abbreviated as $V^*(s)$. Such an optimal policy exists for every Markov Decision Process framework according to Mitchell [71]. So, for example, by considering that the agent is located in a grid-world that consists of 6 cells (as

represented in Figure 2.1) and one of them is the goal state G, given the immediate reward values below the task is to find the shortest route towards the goal state.

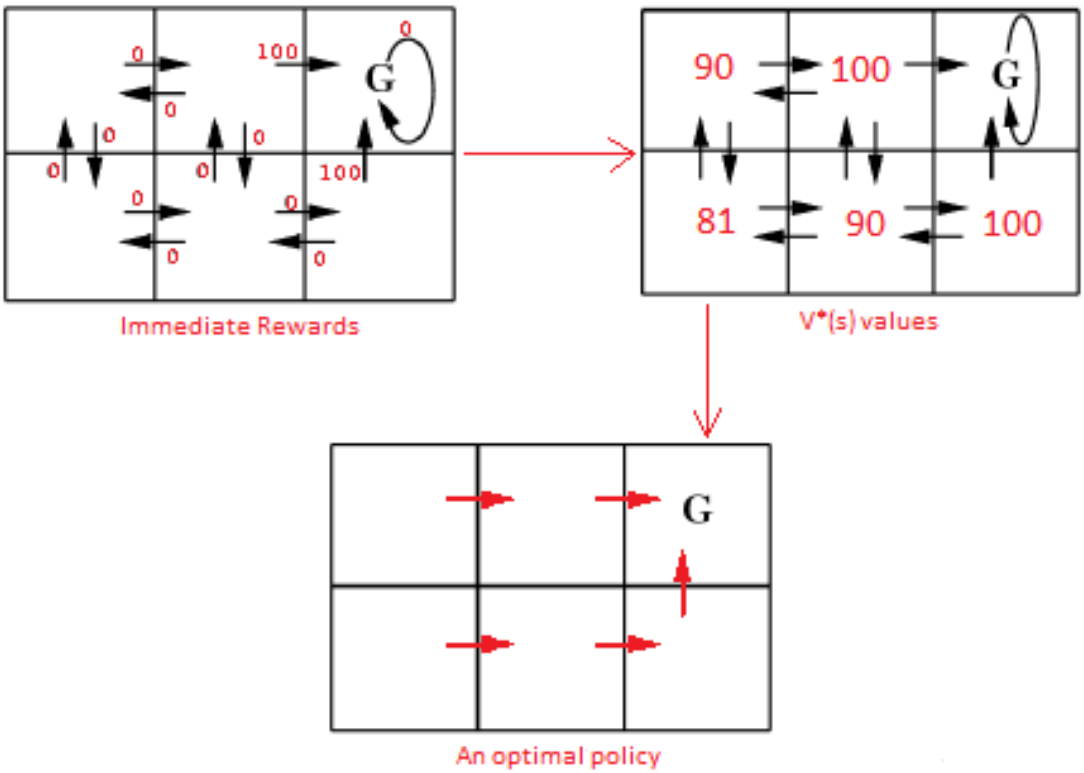


Figure 2.1: *Value Iteration algorithm on the Grid-World. Using the immediate reward values the $V^*(s)$ values are calculated and then the optimal policy's inference eventually occurs. The task, as is being indicated by the optimal policy, is for the agent to reach the goal state G following the shortest route. The image is taken from [71] and is modified according to our example.*

Using a value iteration algorithm (Figure 2.2) on the above example and always assuming that $P(s_{t+1}|s_t, a_t)$ is known, $V(s)$ is being initialised randomly. The agent then traverses through the environment arbitrarily and until there is enough confidence that it has visited every state s from the set of states S while performing every action a from the set of actions A and the policy is solid, this procedure is being repeated by initialising the $V(s)$ again and so on. This is how the optimal policy is eventually being reached. Nevertheless, using the algorithm's execution, many sub-optimal policies are being obtained (one is shown in Figure 2.3.) until convergence has been reached (i.e. the best policy was found, there is no more learning to occur).

Initialize V arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
$\Delta \leftarrow 0$
For each $s \in \mathcal{S}$:
$v \leftarrow V(s)$
$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Figure 2.2: *The Value Iteration algorithm. Taken from [97].*

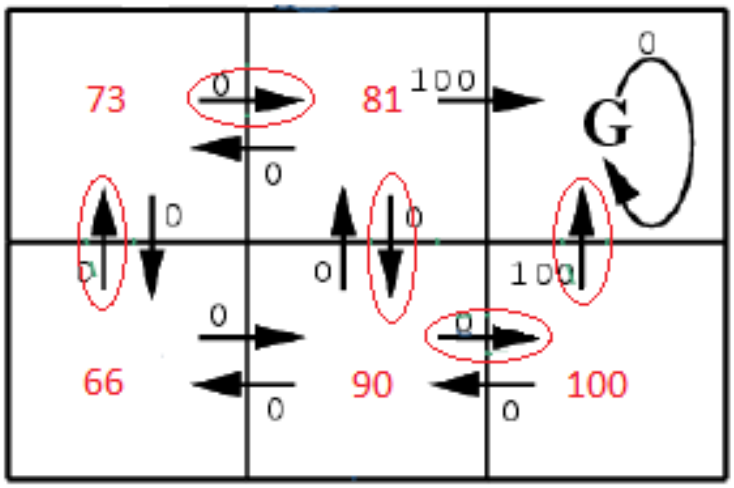


Figure 2.3: *An example of a sub-optimal policy during a value iteration algorithm in a grid-world. Values of states and chosen actions are marked with red. The image is taken from [71] and is modified according to our example.*

2.1.4 Q-Learning

We have seen so far the way to obtain the optimal policy in the case where we know $P(s_t|s_{t-1}, a_{t-1})$. According to Mitchell [71], a further question is what happens when we do not know this probability. The answer comes from Q-Learning [110]. A new function is defined:

$$V^*(s) = E[r(s, \pi^*(s))] + \gamma E_{s'|\pi^*(s)}[V^*(s')]$$

$$Q(s,a) = E[r(s,a)] + \gamma E_{s'|a}[V^*(s')]$$

Please note that the above function is similar to the V^* that we have seen before. The $Q(s,a)$ defines the value of an action a in a particular state s. As the $P(s_t|s_{t-1}, a_{t-1})$ is not needed any more by the agent, it can learn the optimal policy just by knowing the Q(s, a) and -most important- it can learn the Q value without the knowledge of $P(s_t|s_{t-1}, a_{t-1})$ (i.e. "model-free").

$$\pi^*(s) = \arg\max_a Q(s,a)$$

$$V^*(s) = \max_a Q(s,a)$$

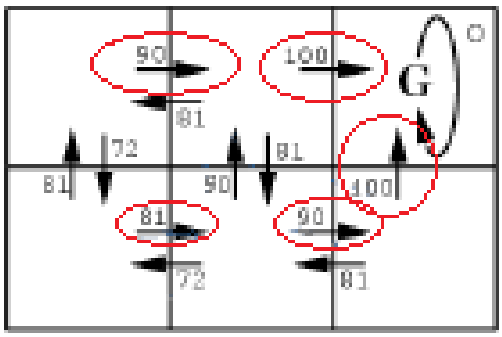


Figure 2.4: *Q-values of actions in a grid-world. The best policy is being obtained by simply selecting the highest Q-values (marked with red ink) of each of the 6 state actions, as shown on the above picture. Image is taken from [71] and is modified according to our example.*

As there is a strong relation between the Q and V^*, Q can be written by recursion as:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t)) = r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a')$$

That leads to the approximation of Q as:

$$Q^-(s,a) \leftarrow r + \gamma \max_{a'} Q^-(s', a')$$

In the above equation, s and a are the current state and action respectively while s' and a' is the following state and action (after applying action a in the state s).

The algorithm for using Q-Learning in a deterministic environment starts by initialising all the Q values of the states actions to 0 (can be random too). Then starting with a random state, an action is selected, executed and a reward is received. This action traverses the agent to a new state, the approximate Q value of the previous state and action is updated as shown in the equation below and the procedure is being repeated by visiting all of the states until convergence has been

reached (that is, the Q-actions obtain the highest value) [71].

$$Q^{-}(s,a) \leftarrow r + \gamma \max_{a'} Q^{-}(s',a')$$

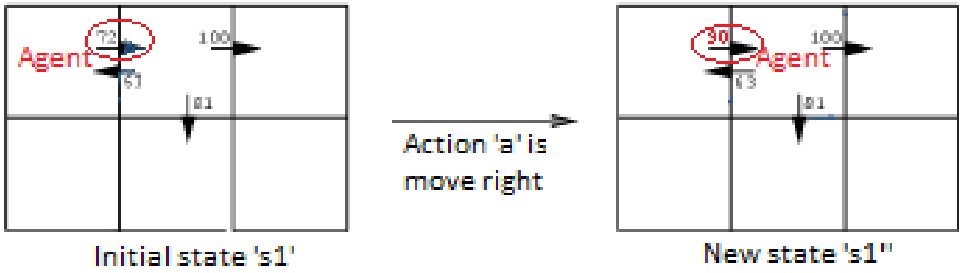


Figure 2.5: *Updating the approximation of Q-value. The agent moves right and the update changes the value from 72 to 90. Image is taken from [71] and is modified according to our example.*

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Figure 2.6: *Q-learning algorithm taken from [94].*

$$Q^{-}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} Q^{-}(s_1', a') \leftarrow 0 + 0.9 \max\{63, 81, 100\} \leftarrow \mathbf{90}$$

Apart from the deterministic case, Q-Learning applies to non-deterministic cases too and the approximation of Q successfully converges to Q as proved by [111]. The training formula is then modified to:

$$Q_n^{-}(s,a) \leftarrow (1 - a_n) Q_{n-1}^{-}(s,a) + a_n [r + \max_{a'} Q_{n-1}^{-}(s',a')]$$

where

$$a_\pi = \frac{1}{1 + visits_n(s,a)}$$

2.1.5 Temporal-Difference Learning

The difference between two successive Q-value estimates is being reduced efficiently due to Q-Learning as we have seen in the previous section. However, according to Mitchell [71], when we need to expand this difference to more than one time step (that is more than two successive estimates) then Temporal Difference Learning is useful. Its learning formula is therefore calculated and defined as:

$$Q^{(1)}(s_t, a_t) \equiv r_t + \gamma \max_a Q^-(s_{t+1}, a)$$

(Difference for one time step)

$$Q^{(2)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a Q^-(s_{t+2}, a)$$

(Difference for two time steps)

$$Q^{(n)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + ... + \gamma^{n-1} r_{t+n-1} + \gamma^n \max_a Q^-(s_{t+n}, a)$$

(Difference for n time steps)

Thus, we have:

$$Q^{\lambda}(s_t, a_t) \equiv (1 - \lambda)[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + ...]$$

(Temporal Difference Learning formula by combining all the above together [93])

Equivalently it can be written as:

$$Q^{\lambda}(s_t, a_t) = r_t + \gamma[(1 - \lambda) \max_a Q^-(s_t, a_t) + \lambda Q^{\lambda}(s_{t+1}, a_{t+1})]$$

The constant $0 \leq \lambda \leq 1$ blends the estimates from the applied look-ahead distances. Temporal Difference Learning calculates current estimates using previous learnt ones.

Based on the book of Sutton and Barto [97] where they study a "tic-tac-toe" example, the applied formula of our preliminary work (Section A.1.2) that is based on Value iteration is:

$V'(s) \leftarrow V(s) + \alpha * (V(s') - V(s))$, where:

α is the learning rate, $0 \leq \alpha \leq 1$,

s' and s the current and previous states respectively,
V' and V are the current and previous values.

2.1.6 SARSA(0) and SARSA(λ)

SARSA [95] is an on-policy control learning method and is based on Temporal Difference. On-policy means that the algorithm's current behaviour (actions) is directly connected to the updated policy, in contrast to Q-Learning (that we examined previously), which is an off-policy learning method. The name originates from the State-Action-Reward-State-Action sequence that is used to update the Q-values of its state-action pairs. This sequence suggests that a learning agent which is currently on state s, takes an action a, receives a reward r, its state changes to s' and then it takes a new action a'. The benefit of using SARSA compared to Q-Learning is that, due to that sequence, it creates optimal policies that are highly affected by the exploring mode of the learning agent. Based on what we examined in the previous sections and according to the authors of the book [97], the algorithm of SARSA(0) is as follows:

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma Q(s', a') - Q(s, a)\big]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Figure 2.7: *SARSA(0) learning algorithm taken from [95].*

SARSA(λ) [96] introduces the use of eligibility traces (denoted by $e(s, a)$) to the SARSA(0) method that provide us with a dynamic degree of strengthening previous state-actions. In detail, eligibility traces are temporary records that indicate how often each state-action pair has been visited and how "valuable" their roles are for our problem. Therefore they are used through the application of the parameter λ, which in combination with γ (discount factor), regulate the above degree of strengthening. For very stochastic problems, low values of λ are suggested (i.e.

0.4) as they have proved to be effective in appropriate examples [96], while in more deterministic cases higher ones (i.e. 0.9) are used. According to Sutton and Barto [97], the algorithm for SARSA(λ) can be seen in Figure 2.8.

Initialize $Q(s,a)$ arbitrarily and $e(s,a)=0$, for all s,a
Repeat (for each episode):
  Initialize s, a
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all s,a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      $e(s,a) \leftarrow \gamma\lambda e(s,a)$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

Figure 2.8: *SARSA(λ) learning algorithm taken from [96].*

In this thesis we use the above algorithms (especially SARSA(λ)) and we investigate various combinations of the λ and γ parameters as we have cases with both stochastic and deterministic actions. Furthermore, the algorithms' update rule applies to the whole history of the game's state-action pairs that the learning agent currently plays.

2.2 Non-Stationary MDPs

A very interesting problem that arises when an agent learns in an MDP environment which suddenly changes its dynamics (parameters of the transition T and reward R functions), is that of "non-stationarity". The case where this sudden change is based on an unknown factor which can't be perceived by the learning agent has been studied by Silva et al. [91]. In non-stationary environments, traditional reinforcement learning algorithms are forced to constantly re-learn the policy in their attempt to exploit current dynamics (or *contexts* according to the authors). Hence, Silva et al. implemented a method called RL-CD (Reinforcement Learning with Context Detection) which solves this problem and is also effective in noisy scenarios. In more detail, they argue that their RL-CD algorithm performs better in this kind of environment than a frequently used model-free one (Q-Learning), which

does not know a priori how the environment works, as well as better than a popular model-based one (Prioritized Sweeping), which already holds information about the transition function T and the reward function R. It is important to mention here though that these algorithms were created to learn in stationary MDP environments, as standard Reinforcement Learning considers that no change in the framework's dynamics ever takes place.

RL-CD assumes that the system consists of distinct partial models and each one of them belongs to different dynamics of the environment. Each of these partial models is capable of predicting when the environment changes (by detecting context changes) and only one of them (the one with the highest *prediction quality* [92]) is activated at any given time until another of a better prediction quality is found in an evolutionary manner. Each of these partial models include transition and reward functions that through the process of standard reinforcement learning eventually lead to a local optimal policy. Update of the transition and reward functions takes place based on previous experience tuples including current state, action, next state and reward. The authors of this work clearly state that the transition of the environments' dynamics does not depend on the learning agent's actions though and therefore we conclude that our case is a relevant, but quite special, subclass (the environment's dynamics change only due to some of the learning agent's actions) as we will examine in depth later.

Choi et al. [17] suggest the use of MDP for non-stationary cases such as ours (multi-agent environment where the learning agent's action affects its dynamics), in controlling the *mode* transition process. The mode, which is considered as an MDP, is a fundamental part of their non-stationary model that they call hidden-mode Markov decision process (HMMDP). This subclass of non-stationary MDP environments assumes that the dynamics of such an environment are restricted to a limited number of hidden modes. As there is no relation between the control system's responses and the modes' transition, which is completely stochastic in their work, they propose that a Markov chain evolves these modes with time. The main point was to introduce a specific variable to hold the current mode (a distinct MDP) that the environment is in. According to the authors, the HMMDP model is a specialised POMDP (Partially Observable Markov Decision Processes) case that in non-stationary environments proves to be faster and less complex than a POMDP. Learning is mainly focused on model-based Reinforcement Learning where an environmental model is initially obtained and then the optimal policy can be found. It occurred through the use of a custom Baum-Welch algorithm and the following environmental properties were considered:

- The non-stationary environment consists of a number of modes which hold specific environmental dynamics. Each one of these modes is considered to be a stationary environment that requires its own policy and only one expresses

the environment in any time point.

- The modes are hidden, they cannot be directly identified but only approximately by observing the sequence of the previous states and actions.
- The control system's response does not affect the modes' transition. The latter are stochastic events which may happen due to external factors.
- Modes do not change frequently.
- The modes' number is usually significantly smaller than that of the states.

The HMMDP model does not require the two last properties whatsoever and therefore it is quite flexible in capturing real world applications. However the number of modes must be known in advance. This interesting, model-based RL work focused only on learning the non-stationary environmental model but not on deriving the optimal policies of each mode. Thus, issues such as exploration-exploitation were not taken into consideration. Due to the fact that -again- the learning agent's actions do not affect the environmental dynamics and our approach is also relying on a model-free basis (where learning occurs while interacting with the environment), we only keep the future use of a special variable to characterize our current hidden mode (MDP) during the learning process, that indicates whose turn it is to propose a trade or respond to one (Chapters 3, 4 and 5).

2.3 Game Theory

Game Theory or "Games of Strategy" as stated by the pioneers of the field Neumann and Morgenstern [74] is the mathematical analysis of cooperative and competitive strategies resulting from the interaction of entities. These entities, common in Computer Science as agents, can be programs, robots, individuals, teams, companies etc. The goal is to infer and predict optimal actions in order to maximize the success of these policies and therefore the accumulated profit. According to Neumann-Morgenstern, an essential question when it comes to studying a game -for the sake of classification- is whether or not the sum of the players' payments at the end results in zero. If the answer is yes, then the game is called zero-sum game (e.g. poker). In the opposite case, the game is known as non-zero-sum game (e.g. "Catan" board game) [74].

In a game, the plan that a player may have in advance is defined as strategy. It includes specific choices that the player will make over a set of potential states, given that the rules of the game are known before-hand and the information remains consistent. This shouldn't be considered though as a constraint. As stated

by Neumann-Morgenstern: "...if we require each player to start the game with a complete plan of this kind, i.e. with a strategy, we by no means restrict his freedom of action. In particular, we do not thereby force him to make decisions on the basis of less information than there would be available for him in each practical instance in an actual play." [75]. The concept of strategy is instead a thoughtful mental weight, reinforcing the player's insight and functions as a goal-driven assisting utility.

2.3.1 Pure and Mixed Strategy

Neumann and Morgenstern [74], studying at length the concept of strategy, divided it into two categories. They defined as strict or pure strategy of a game the set of deterministic actions (or moves) that a player will wholly choose in every distinct situation. On the other hand, a mixed strategy is characterized by probability distributions of its stochastic actions focusing on the frequencies of their selection. The need of choosing a mixed strategy arises from the uncertainty of "picking" between pure strategies. In the cases where the player would like to trigger adversarial prediction, a mixed strategy is highly suggested (such as in a two-person zero-sum game as suggested in [57]).

2.3.2 Cooperative and Non-cooperative games

From the early stages of the history of Game Theory, the concept of the game was separated into two distinct categories. One of them, the so called "cooperative games" (as stated by John Nash [73]), included the games where the players create various forms of affiliations in order to collectively reach often shared goals. The book of Neumann and Morgenstern [74] suggests a theory based on this category as discussed by [73]. The second category, includes non-cooperative games, where affiliations between players do not exist as they carry on independently towards personal targets. The lack of communication[1] is common in this kind of games that played a significant role in the study of the founder of these two terms, John Nash [73]. Examples of popular non-cooperative games are backgammon, chess, tic-tac-toe, rock-paper-scissors, poker. Our game Taikun as well as the Catan board game (they will be both examined in depth later in the thesis) are also non-cooperative

[1]The lack of communication was a major issue that we experienced in our experiments with humans (Chapters 6 and 10). The human players are very cautious in non-cooperative games and avoid trading quite often in our cases. The reasons will be analysed in the corresponding Chapters 6 and 10.

games where the players trade resources in order to win, but they also involve communication.

2.3.3 Nash Equilibrium

John Nash was the first to introduce the idea of the equilibrium point in his Ph.D. thesis [72] in 1950, as the main element of his theory about non-cooperative games. According to his work [73], this point -that is nowadays known as Nash equilibrium- expresses opposing "good" strategies during a game. The key concept is that if a player decides to change his/her strategy then, given the fact that the strategy of the adversary remains the same, there is no way of increasing his/her performance. Hence, each one of these strategies is considered to be optimal versus those of the other players. There is at least one equilibrium point in any finite non-cooperative game according to John Nash [73].

2.3.4 Pareto Optimality

The idea behind Pareto Optimality originates from the Italian economist Vilfredo Pareto [76], known for his income distribution study as well as his analysis based on individuals' preferences. Influenced by the view of Economics [7], a game's solution is considered to be Pareto Optimal if it provides the best possible outcome to all of the players. Thus, it is not possible to improve a player's state without worsening the state of any of the others and therefore it is a solution that is preferred by all of them. Usually Pareto Optimality is treated as a Nash Equilibrium (Section 2.3.3). However, this equality ceases to exist in the case where the players' pay-offs can all be increased. Then there is still Nash Equilibrium but not Pareto Optimality any more as there would be still space for improvement.

2.3.5 Dominant Strategy

In his same work [73] as we saw above, John Nash suggested that a strategy is dominant if it provides the largest pay-off to its player when compared to those of her adversaries. No matter what patterns of actions the other players will perform and what any other strategies they may adopt, there can be absolutely no better strategy than a dominant one. Hence all of the other strategies are defined as

dominated and their meaning is the opposite of that of the equilibrium point, where opposing "good" strategies exist. In a deeper analysis, a strictly dominant strategy presupposes that it provides a higher pay-off than any other strategy while on the other hand, a weakly dominant strategy provides a pay-off at least as high as any of those of the other strategies. Following the same reasoning, a strictly dominated strategy is defined as a strategy which is dominated by any other one while a weakly dominated is the strategy which is as dominated by at least one of the others [34].

2.3.6 Perfect and Imperfect Information Games

The prisoner's dilemma game [23] is a game of imperfect information and the reason is because the criminals cannot communicate and therefore each one of them does not have knowledge about the other's actions. Based on that example, the imperfect information game as discussed by Neumann and Morgenstern [74] is the sequential game where the players do not have complete information about their opponents' actions up to any given point. An example of an imperfect game is poker. On the other hand, a perfect information game is the one where, at any given time point, each player knows exactly the moves that his opponents have made and we further consider that in this game only one player action happens at a time. Examples of perfect information games are: chess, backgammon, tic-tac-toe. Our "Taikun" and the "Catan" board games that we will examine from Chapter 3 onwards are both imperfect information games as the players cannot see the available resources that their opponents have and their goals are unknown.

2.4 RL in Dialogue Systems

Many researchers such as Rieser and Lemon [82], Georgila and Traum [38] or Williams and Young [112], argue that holding a conversation through the use of machines is a challenging task. Humans naturally obtain and develop the required skills in order to communicate with each other through time. For a machine though and specifically a dialogue system, there is often a dialogue designer to expertly dictate specific actions that the system should perform under specific conditions. The machines' skills then are being obtained and developed under this dictation of actions that is also known as "dialogue strategy". This strategy is an essential component of the Dialogue Manager (DM) that automatically determines the whole system's behaviour.

The dialogue system consists of three general modules: input, output and control module respectively. Automatic Speech Recognition (ASR) and Spoken Language Understanding (SLU) are components that comprise the input module. The output module includes the Natural Language Generation (NLG) component as well as a Text-to-Speech (TTS) system. The control model is in fact the Dialogue Manager which follows a specific dialogue strategy. ASR performs the conversion of the user's speech to text. The SLU is a parser that analyses input text from ASR and generates a string of various features that have specific meanings for the system, such as Speech Acts. This string is then being processed by the DM (which also considers the dialogue history) and it finally selects the optimal action to perform based on its dialogue strategy. The output module through the use of the NLG system initially interprets these actions to text and then, via the TTS engine, it converts this text to audio for the human user.

The authors above [83] further discuss that the concept of a Natural Language dialogue is similar to that of a board game, requiring constant strategic planning under uncertainty. Thus, they characterised dialogue as temporal, meaning that the value of an action depends on the quality of the dialogue's further progression. That means, action selection in a dialogue strategy requires careful, long term planning and it is heavily affected by the current state of the agent in the environment as well as by the future deeds. Being in a specific state, it is hard to tell whether or not a current specific action is the ideal due to the fact that the future is unknown.

However, it is feasible to determine whether or not the performance of a whole dialogue has been adequate (by user ratings or task success) and then with the use of Reinforcement Learning consider and apply appropriate values to those actions that led to the current result. The sequential decision nature of RL with the use of delayed rewards (can be positive or negative as mentioned in Section 2.1) at the end of the dialogue is able to provide a precise characterisation to each of the dialogue actions overcoming local minima. An action that rationally might be considered wrong for the current state of the game (or dialogue) might be eventually proven to be the optimal for success in the long run, following a specific potential strategy.

Rieser and Lemon [83] further characterise dialogue as dynamic. That means the interaction takes place in a stochastic environment where unpredicted events may happen. For instance in a dialogue, the human participant might make statements that would change the rest of the dialogue flow in an entirely unforeseen way. Thus the dialogue strategy must be robust, providing successful actions in such unpredicted circumstances. The exploration phase of Reinforcement Learning (Section 2.1.1) plays a significant role in adaptive dialogue systems and is known as evaluative feedback, where the environment provides feedback to actions whose value cannot be predefined. This feedback can be either good or bad, in a form of a positive or negative reward. The learning agent (dialogue system in this case) learns

how to adapt to the environment according to that, by maximising the incoming positive rewards and minimizing the negative ones (penalties).

The dialogue can be considered as a Markov Decision Process (details in Section 2.1.2). A dialogue strategy consists of mappings from dialogue states to actions in a RL framework, where the learner traverses through these dialogue states as through the nodes of a supposed network. The dialogue is dynamic, as has been previously mentioned (non-deterministic environment) and therefore the transitions are stochastic. The learning agent performs a dialogue action and changes the future dialogue states along with their actions (alters the environment) of the rest of the dialogue. Hence, the states of a dialogue in an MDP framework are mainly represented as dialogue history information, including specific features of the past, current user input action features (user's speech act and confidence value given by the speech recogniser), and task-oriented features (such as user's goal). The dialogue actions in an MDP are typically represented as requests, confirmations, or reports of relevant information.

2.4.1 Reinforcement Learning in Non-Cooperative Games

Several studies of various researchers have taken place in the past about Reinforcement Learning in non-cooperative games, as is being studied in this work, although they do not consider dialogue. In this section though we will discuss approaches which are mainly focused on Game Theory. According to Ishibuchi et al. [52], in a multi-agent non-cooperative repeated game based on a custom rule-set, a Fuzzy Q-Learning algorithm performed better compared to other strategies. In more detail, the game involved competition over a number of markets and the goal was to optimally select the market where a product would sell higher due to its increased demand -because of short supply- at that specific market. Thus, the price of a product in a specific market was driven from all of the agents' actions. Random strategies, optimal strategies as well as simple Q-Learning strategies were applied and suggested that Reinforcement Learning and in particular Fuzzy Q-Learning is a successful method to work with in a similar framework.

Focused on market games again, Erev and Rapoport [30] reported that even minimal given information in a simple Reinforcement Learning model is able to provide with an equilibrium point (as discussed in Section 2.3.3) rapidly. Qualitative alterations between features of various different learning processes may not play the most important role in this kind of games as they are proven to not affect significantly the approximation of optimal behaviours in such environments. These features cannot be ignored though. Amongst other conclusions that are purely directed to Game Theory and market games, we generally maintain from this work

the fact that simple RL models are able to successfully capture collective behaviours in stochastic games[2]. The work of Vishnu and Tapas [102] is also based on markets. With the use of a stochastic approximation based RL methodology they provide solutions to market concepts (represented in their work as matrix games of many players) where there can be large sets of actions. Their approach is able to obtain Nash equilibrium from this kind of market game models through the use of a Value learning function in an MDP framework.

Littman [68] argues that in a two-player non-cooperative game, where the adversary is considered to be a part of the MDP environment (as in our case again) from the perspective of the learning agent, Reinforcement Learning and in particular an algorithm influenced by Q-Learning can successfully form the optimal policy. The "max" operator during the update step of the Q-Learning was replaced by a "minimax" operator that assists on the smooth convergence of the agent's policy to a fixed and "safe" strategy. However, such an agent would be vulnerable to an adversary which would attempt to deceive it. Hence this specific part requires further work as it is of high significance. Two-player games based on the MDP framework are a restricted class according to Littman and therefore are considered to be an interesting case, as in this thesis. Especially the part where manipulation arises will be studied later thoroughly.

Extending Littman's work, Hu and Wellman [48] developed a multi-agent algorithm based on Q-Learning and provided proof of its convergence to a Nash equilibrium. According to their findings, in the case where one only Nash equilibrium exists, the algorithm is able to retrieve the optimal strategy under specific restrictions. If more than one equilibrium exists then they suggested that their algorithm should be combined with other learning methods in order to obtain optimal policies. The above convergence does not depend on the sequence of actions that the learning agent has chosen during its learning process but it is a matter of whether or not each of the state-actions has been attempted. Furthermore, exploration and exploitation need further work according to their statements as the Reinforcement learning method in their multi-agent environment requires an infinite amount of trials to converge. The trade-off between the exploration and exploitation is also being investigated cautiously in our research.

Leslie and Collins [66] introduced a model free algorithm based on a stochastic approximation of two time-scales (according to Borkar's theory [10]). Along with the application of a normal reinforcement learning algorithm they proved that both of them were able to obtain a Nash equilibrium for the cases of two-persons zero-sum games as well as multi-player cooperative games. However, in the cases of two non-

[2]This was an interesting point. We will see in Chapter 9 that our RLA, trained versus a single opponent, can still be successful versus many, showing that RL is capable of capturing adversarial "collective" behaviour even in a bilateral negotiations environment by treating them all as one. However in the work of [30] the learning process occurred while playing against many opponents.

cooperative games (the Shapley's rock-scissor-paper game and the N-player matching pennies game) they were incapable of doing so. By extending their stochastic approximation to more than two time-scales, and therefore achieving learning processes for each player at different rates, they managed to solve the above issue by reaching Nash equilibrium in all of their tested games (including the non-cooperative ones).

Camerer and Ho [15] through what they called experienced-weighted attraction learning (EWA) combined features from belief-based models with those of reinforcement learning and developed a model that assigns "attraction" values to a number of strategies according to their initial probabilistic tendencies towards selection. Their update then depends on their performance in a stochastic game (they experimented on 3 different ones) and these strategies affect other hypothetical ones that haven't been used so far and are based on their potential pay-offs in a game. Thus, the reinforcement weight parameter δ expresses the strength of these hypothetical strategies which is also taken into consideration (along with attraction discount factors and a strategy experience weight) when it comes to select the optimal policy. The EWA model performed better than the pure reinforcement models in all of their tested cases and in most of them, it performed better than the belief models too. RL along with belief-based models (and in particular opponent models) influenced us in Section 8.3, as the experiments there address cases where the RLA maintains adversarial preferences in its state representation and successfully learns from them. All of the work of this section shows that RL is a promising method to handle non-cooperative games.

2.4.2 Reinforcement Learning in Negotiation Dialogue Management

This thesis takes into account non-cooperative negotiation but a significant amount of work that has been made by other researchers is based on cooperative negotiation as we will see. Heeman [45] was one of the first researchers to use RL for cooperative negotiation dialogue. He investigated various types of information that should be included in the state representation in order to produce optimal policies, while maintaining the state space reasonable in size. He suggested that by tracking the decisions of the system and by utilising them in a way that restricts its future dialogue behaviour, we can represent the strategy that it uses piece by piece. This method results to good cooperative negotiation policies according to the author [45]. Other important findings that are discussed below mainly originate from RL and dialogue management in various negotiation scenarios (not necessarily

trading ones as in our case), and they were the main inspiration for the current thesis. Georgila and Traum [37, 38] used three different types of cultures (individualist, collectivist and altruist) to tweak accordingly hand-crafted agents, which they called Simulated Users as they simulate the behaviour of real ones. They were based on a dialogue corpus which was not specific to any cultural dimension, and with the use of Reinforcement Learning, they learned culture-oriented negotiation policies in a "one-issue" negotiation scenario. The corpus included dialogues of a florist and a grocer who have different goals and kinds of arguments, based on four issues: the rent, the temperature, the design of the space and the advertising policy.

The produced RL policies accounted for argumentation and persuasion, in contrast to most of the related work that was made up to that point which was based on slot-filling applications (i.e. finding a restaurant). The authors [37, 38] argue that their approach managed to learn and evaluate RL policies for specific cultural dimensions based on information which was not focused on any particular culture. Overall they suggest that RL is a promising method for learning argumentation policies in negotiation domains. In more detail, they used a SARSA(λ) algorithm with an initial exploration rate of 30% in [37], 20% in [38] (similar to our case that we will examine from Chapter 5 onwards) and 20,000 training iterations. The state representation ended up consisting of 8 features, that led from 864 to 4374 states, and 12 actions.

Based on the above, Georgila [35] used a variety of hand-crafted agents (Simulated Negotiators) in order to generate corpora for two RL agents to be trained on. The Simulated Negotiators (SNs) used different goals and arguments (strong and weak ones) in the process of persuading each other, performing irrational actions too for the sake of negotiation variety in the corpus. The negotiation scenarios that were tested now were "two-issues" ones. The goal was for the RL agent to learn policies that persuade the other agent to agree on its preferences. The author [35] shows that the RL agents learned to perform equally with the hand-crafted SNs (which made only reasonable dialogue moves) and sometimes outperformed them, by successfully persuading their interlocutors on their preferences (food type to be served and day of the week that a party should take place). The state representation included 15 features (8 were binary), leading to 786,432 states, and 13 actions were used. The Least Squares Policy Iteration (LSPI) [65] was applied, as it can learn directly from a dialogue corpus, with linear function approximation [70] of the Q-function with 1,680 manually selected features.

Georgila et al. [36] also used single-agent and multi-agent RL techniques to train two agents which simultaneously learn dialogue policies. In the trading negotiation scenario that was used, the two resources change ownership according to the agents' preferences. The non-stationary environment due to the agents' change of preferences as well as due to the fact that both of the agents learn concurrently causes

problems to the single-agent RL algorithm (Q-Learning), which does not converge (it does only in the case where the state space is small). On the other hand, the two applied multi-agent RL techniques both converge and suggest that the traditional single-agent RL versus a simulated user, or a corpus to learn from is not needed. These interesting results bring to light the difficulty of single-agent RL to deal with non-stationary MDPs (that we examined in Section 2.2) once again and therefore provide a strong indication of how difficult it is in our case to train a RL agent which learns how to successfully trade using linguistic manipulation. Its effect constantly changes the adversarial behaviour and therefore makes the environment non-stationary. However our learning algorithms still manage to successfully learn how to use linguistic manipulation in this kind of environments, as we will see later from Chapter 3 until the end of the thesis.

A multi-issue negotiation policy in a two-agent environment was learned by Papangelis and Georgila [79]. A Q-learning method with function approximation was used to train against a hand-crafted negotiation dialogue agent. The trained policy was aimed to be used in negotiations with agents whose behaviour has not been observed before, and in particular humans. The authors [79] report that the learned policy is also designed to work against behaviours that change during the same interaction, resembling our *learning* cases of non-stationary MDP environments, discussed in the Sections 3.3.6 and 8.2.14 of the thesis. The hand-crafted dialogue negotiation agent was based on the agenda-based paradigm [88] which was used for restaurant recommendation and has been used for dialogue management [86] too. The goals and the preferences of the agents are not known by their interlocutor. The state space consisted of 10 features (some are binary) and their work considered a part of the full state space, which they call summary state space and holds a more abstract representation. Five batches of 20,000 training iterations each were used to train the learning algorithm.

The reward function took into consideration the case of non-agreement (a penalty was applied there) and, in the case of an agreement, the distance[3] between the agent's preferences and the agreed ones. It also included the best (possible) achievable score that the policy could result in, assuming that the opponent is absolutely cooperative. For the evaluation phase, the RL agent interacted versus the hand-crafted one (agenda) for 20,000 episodes and humans also rated the negotiations between the RL agent and the hand-crafted one. Both of the agents used 9 actions (those of the agenda rule-based agent). On the evaluation part, the performance of the RL was better than that of the hand-crafted one and humans rated the RL one higher. This work showed that RL is capable of learning successful negotiation policies with the use of argumentation (e.g. persuasion) in multi-issue negotiation scenarios, as it was designed to work against interlocutors whose behaviours have

[3]The notion of distance in the reward function was taken into consideration in our work too, that is discussed in the Section 7.2.3 of the thesis.

not been observed before (i.e. the RL agent was not trained on them).

Based on cooperative negotiation again, Hiraoka et al. [47] used Reinforcement Learning (Neural fitted Q iteration) in a Partially Observable Markov Desicion Process framework, in order to learn persuasion policies to an agent from a negotiation corpus. It consisted of persuasion dialogues between a salesperson who was the persuader and a customer, the persuadee. The persuasion was based on framing [51], that is the use of emotionally charged statements to affect the persuasiveness of a dialogue. According to the authors [47], an example of positive framing would be: "(Camera A is) able to achieve performance of comparable single-lens cameras and can fit in your pocket, this is a point". The goal of the salesperson was to persuade the customer to buy a particular camera, out of five available. The user simulator in this case is the customer (persuadee), who is based on an order one Markov Chain and uses a Bayesian network[4] and the salesperson is the Reinforcement Learning agent which learns how to successfully persuade the customer with the use of framing. The reward function took into consideration the user satisfaction, the persuasion success (i.e. whether the customer decides to buy the particular camera or not) and the naturalness indicator from real users [69]. There are 13 actions, which consist of pairs of framing statements and general-purpose functions (GPF) [54].

The belief state of the RL agent includes the features of the reward function and the reward that was calculated at the previous turn. The evaluation took place with the simulated user and real ones. In detail, the RL policy was based on a random one (there was obviously no learning there), a non-framing and a framing one. After 10,000 training dialogues and testing on 1,000, the authors state that learning greatly improves the performance and framing is somewhat effective to the user simulator. Corresponding comparisons occurred between the random policy and the framing or the non-framing one. Furthermore, the first author (wizard) evaluated the system by selecting the actions (and introducing a new policy called "human") according to the Wizard of Oz framework [62] along with 13 more participants (evaluators). The results from all the 4 policies that were used by the evaluators and the wizard suggested that framing is a significant factor for persuasion and it is effective with real users in persuasion dialogues.

Hiraoka et al. [46] recently showed that in non-cooperative multi-party trading scenarios, three RL algorithms were capable of successfully trading against random and rule-based policies. The rewards that were given to the LAs incrementally, proved to assist the applied learning algorithms significantly more than those that were given at the end of the dialogue (as in our cases). Furthermore, the use of a multi-layered perceptron to approximate the Q-function resulted in the best performance, compared to a classic Q-learning algorithm with function approximation and a least squares policy iteration [65]. There the Q-function is again calculated

[4]That resembles our case in Section 9.1.4, where our trained RL agent plays the "Catan" game against a Bayesian agent which acts based on a corresponding trading dialogue corpus.

through a linear function approximation [70]. The authors emphasise the fact that even in simple scenarios of multi-agent dialogues, successfully learning to negotiate is a very hard problem.

Recently, Keizer et al. [61] used RL in a multi-agent (4-player) non-cooperative trading game "Catan" and successfully learned how to outperform three expert rule-based opponents (called Bots in Section 9.1.1) as well as three Supervised Bayesian agents (called Bayes in Section 9.1.4)[5]. Training, testing and evaluation took place in the JSettlers research environment [99], an open source client-server system which allows playing the game with human players through a graphical interface but also includes artificial ones. The artificial players which are expert hand-crafted agents (Bots) use advanced building strategies and complex rules for their trading negotiations with their opponents. Focus, such as in our work, was given on the trading part and therefore the RL agent had to learn how to successfully trade by following the same building plan with that of the Bots. According to the same game play reasoning, the Bayesian agents' trading logic was based on a corpus collected by [1]. It consisted of annotated human player data, with focus on the trading part, and the Bayesian agent simulated that behaviour. The authors reported that the RL agent, which was learning by playing training games against three of those two different kinds of agents, resulted to better performances (more than the expected 25% level) than those of the Bots or the Bayes agents. 40,000 training games were played and then they were tested on 10,000 ones. Overall, the Monte Carlo Control (MCC) method that was used for optimising the MDP learning policies proved to be successful after training on human trading dialogue data. Thus hand-crafted trading policies are suggested to be a less successful approach. This work though did not investigate manipulation in Catan, which will be discussed in Sections 8.2 and 9.2 of this thesis for the cases of bilateral and multilateral negotiations respectively.

In the area of persuasion dialogues and with focus on opponent modelling, Hadjinikolis et al. [42] proposed a technique for augmenting the opponent model (OM) of an agent with information *likely* related to that which is already contained in it. The inferred information is based on the knowledge about other agents with similar beliefs and on our current assumption about the opponent's belief. The agent's history of dialogues[6] is taken into account and in particular the times that certain arguments follow other specific ones. The authors consider those arguments related if they belong in the same dispute line of the dialogue (i.e. they support each other semantically). Thus it is suggested that an agent's policy is more effective when the agent relies on how often particular arguments follow specific others. The con-

[5]Our RL policies have been also tested against those opponents in the same environment, as we describe in detail in Chapter 9.

[6]Similar cases will be discussed in Sections 8.3 and 8.4, where our RL agent takes into consideration the history of the adversarial preferences in its state representation, and successfully learns based on those. We take into account not the frequency, as the authors do, but the quality (i.e. the type of the preferred resource) of the preferences and the conditional relations between them.

tribution of their work includes a method for creating a corresponding relationship graph that represents those arguments within a specific semantic context. It also consists of a technique for updating and augmenting the agent's beliefs about its opponent's ones, according to the arguments' relations in the same dispute line, and the definition and analysis of a Monte Carlo [44] simulation methodology where the augmentation takes place and converges. The augmentation technique calculates the likelihood of the consecutive arguments according to observations on the agents behaviour in dialogue, through Monte Carlo simulations. Through sampling, they infer the probability values that indicate the occurrence of an argument following a previous one in the same dispute line.

This section described work that has inspired the current thesis the most. Similarly to Georgila and Traum [37, 38], we implement LA's based on RL, and in particular SARSA(λ), to learn how to successfully negotiate. As in [38], our initial exploration rate from Chapter 5 onwards is 20%, which is gradually decreased to 0% at the end of the training. Our LAs learn how to negotiate versus handcrafted agents which demonstrate various behaviours, not based on corpora as in [37, 38] but on other empirical studies (based on Pragmatics, Philosophy, Psychology) as we will examine in detail from Chapter 3 onwards. The same authors used argumentation and persuasion (as we saw in [47] too), which motivated us to use linguistic manipulation (which can be persuasion too as we will se in Chapter 8) in our non-cooperative environments.

Preferences played a significant role in Georgila's work [35, 36] as we have examined above. The same occured in our case where we used opponent modelling (as in [79]). Similarly with [42] our RLAs take into consideration the history of the adversarial preferences, but based on the quality (i.e. the type of the preferred resource) of the preferences and the conditional relations between them (CP-NETs that we will discuss in the next section). Our reward function from Chapter 7 onwards resembles that of [79], where the distance is addressed. Keizer et al. [61] work was important for us too as we experimented (Chapter 9) in the same multi-agent environment using our trained policies from our bilateral negotiations (Chapter 8). Unlike [61], we applied linguistic manipulation there too.

2.5 CP-NETs

Boutilier et al. [11] introduced the Conditional Preference Networks (CP-NETs) as a means of representing graphically preference relations that are being formed naturally in people's everyday life. Based on the Ceteris Paribus interpretation that means "all else being equal", CP-NETs are capable of specifying various types of

relations between preferences in a structured, compact and direct way. Avoiding the concept of uncertainty in actions or states and focusing exclusively on a world's setting similar to that of the MDP (Section 2.1.2, where everything is observable, the information is complete and there is a finite number of distinct states and actions), they apply a typical preference ranking over a set of action outcomes. For example, $o1 > o2$ means that the outcome 1 is more desirable than the outcome 2 and $x1x2 > y1y2$ (or $x1 \wedge x2 > y1 \wedge y2$) means that the outcomes of x1 and x2 are preferred than those of y1 and y2 etc.

According to Boutilier et al. [11] definition of CP-NETs: "A CP-net over variables $V = X_1, \ldots, X_n$ is a directed graph G over $X_1, \ldots, X_n$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. Each conditional preference table $CPT(X_i)$ associates a total order with each instantiation U of X_i's parents $Pa(X_i) = U$."

Despite the fact that the CP-NET's semantics allow reference to variables that belong to arbitrary finite domains, the following example that is inspired by those of Boutilier et al. [11] and will be studied thoroughly uses binary variables: *Let's assume that an artist's preferences for drawing a picture includes three variables, P, C, and Q that are pen, pencil and quill respectively. She strictly prefers using black than green colour for both the pen's ink and the pencil, regardless of the quill's ink colour. However, the choice between the red and the green quill ink is affected by the combination of pen and pencil colours. In the case where the pen's ink colour and the pencil's colour are the same, then the green quill ink is not of her taste and therefore she prefers the red quill ink. In the case where the pen's ink colour and the pencil's colour are different, then the red quill ink is not of her taste and therefore she prefers to use the green quill ink.* The Figures 2.9, 2.10 and 2.11 represent the details of the preference relations between the variables of the example above.

Pb $\succ$ Pg	Cb $\succ$ Cg

Pb $\wedge$ Cb	Qr $\succ$ Qg
Pg $\wedge$ Cb	Qg $\succ$ Qr
Pb $\wedge$ Cg	Qg $\succ$ Qr
Pg $\wedge$ Cg	Qr $\succ$ Qg

Figure 2.9: *Preference relations for the "picture drawing" example. The choice of the pen's ink colour and that of the pencil affects the colour choice of the quill's ink.*

Boutilier et al. [11] classical view of CP-NETs as well as of the induced preference graphs considers them as a network where each of the nodes has at least one connection to another one and belongs to a distinct level of preference, as can be seen in the Figures 2.10 and 2.11. Figure 2.9 represents analytically the preference relations of the variables that are discussed in the above "picture drawing" example.

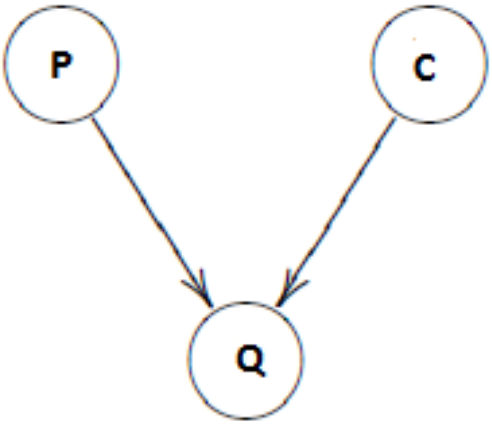


Figure 2.10: *CP-NET of the "picture drawing" example. The quill's ink colour is conditionally dependant (equally) by both the colours of the pen's ink and that of the pencil.*

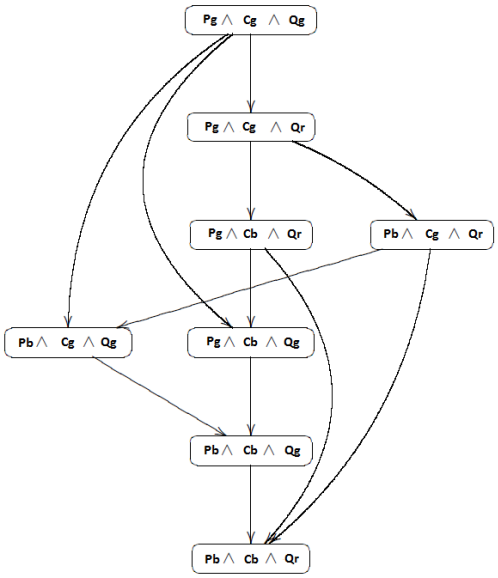


Figure 2.11: *Induced preference graph of the "picture drawing" example. The highest in preference node Pb ∧ Cb ∧ Qr is at the bottom of the tree and all of the others follow in a hierarchical manner.*

In Figure 2.10, the CP-NET of the example considers P and C to have the same (highest) level of preference while the Q is conditionally dependant (equally) by both the P and C. Thus, in the inferred graph of Figure 2.11 there are six distinct levels of preference and the highest in preference node (that is at the bottom of the tree) includes the expression Pb ∧ Cb ∧ Qr that stands for: black pen's ink and black pencil and red quill's ink. This is the highest in preference node. All of the other preference nodes follow in a hierarchical manner based on the example's description and the representations of the Figures 2.9 and 2.10.

2.5.1 CP-NETs in Dialogue Acts

Asher, Bonzon and Lascarides [3] based on Boutilier's classical view of CP-NETs, proposed a methodology for modelling the preferences that affect the discourse during a dialogue act. It resulted to the concept of a partial CP-NET, where relations between preferences in a discussion are represented and computed, allowing also discourse expressions that sometimes do not belong in a specific degree of preference and therefore are left unranked. That happens because the expressed preferences in Natural Language may be vague. Hence, partial CP-NETs may lead to more than one optimal outcome as they can be acyclic.

In Natural Language Processing (NLP), preference retrieval and representation from texts is a complex task that according to the authors [3] can be accomplished through partial CP-NETs. They initially analyse the discourse into rhetorical related units that consist of a set of labels (such as plan-elaboration, explanation, question answer pairs etc.) based on their theory of discourse interpretation (SDRT). After that, they gradually compose them then back to form a partial CP-NET by following a recursive manner, inspired by Game Theory techniques that are able to deal with non-aligned preferences.

The ERC project STAC (`http://www.irit.fr/STAC/`) aims to develop models based on strategic non-cooperative negotiation. Due to the fact that its relevance to the current thesis' subject is high, our findings contribute to STAC project. Furthermore our work is inspired by the challenges and motivations of that project and is considered to be a part of it. STAC makes use of the multi-player board game "Catan" and in particular a relevant research environment called "JSettlers" [99]. In Chapters 8, 9 and 10 we focus our attention on that game as well as the "JSettlers" environment and conduct experiments. "Catan" data that have been accumulated so far from games between human players have been annotated by Asher, Bonzon and Lascarides with the use of partial-CP-NETs [3], leading to very interesting results as we saw above. Hence, CP-NETs play a significant role in our work and experiments which utilise them in Reinforcement Learning have been conducted as we will see in Section 8.4.

2.6 Pragmatics

Pragmatics is the area of linguistics that studies the use of language in context, along with its various mechanisms that we use to express potential actions or events. Grice's cooperative principle marked the birth of Pragmatics in linguistics and has been the subject of many scholars [41]. It was stated as follows: "Make your con-

versational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged" [39].

2.6.1 Gricean Maxims and Implicature

Four categories (known as Gricean Maxims) emerge from the above statement: *Quantity, Quality, Relation and Manner*. Quantity characterises the amount of information that needs to be contributed to the conversation. The sub maxims that result from that are two: First, the contribution needs to hold the required amount of information and second, this information must not be more than is required as it would result in a waste of time and misdirection. Quality refers to the truthfulness of the information. There are two related maxims again here. The communicator must not say that she believes not to be true and that she lacks adequate evidence for. Relation refers to the need of information to be relative and Manner to the way that the information is said, as the talk should be regulated, brief and clear.

Some of the above maxims may be more essential than others in maintaining the cooperative behaviour of a dialogue, and all require that of Quality. Grice also argues that there are other maxims, such as social, moral, or aesthetic ones that belong to a secondary level. However the four that have been mentioned above along with the conversational implicatures which are generated from them are the most important in discourse. People naturally learn and behave based on these principles since childhood when engaged in dialogues with their interlocutors. That means it would require effort to invent for instance a lie than to speak the truth. Hence it is reasonable and easy for us to follow these rules in order to ensure co-operative dialogue. According to Grice, this kind of dialogue is characterised by a common, immediate aim. It is also based on mutually dependent talk and mutual understanding regarding the communication's finalisation as well as its flow [39].

A sentence's suggestion which resides between the points of literal expression and that of the clear implication is called implicature according to Grice. It is related to the uncertainty of the maxims' violation and its notion can be categorised in three distinct kinds:

- Conversational implicature
- Scalar implicature
- Conventional implicature

The conversational implicature is divided into three cases. The first case covers the situations where the communicator disregards a maxim on purpose, for the

sake of a non-literally expressed context. An example might be: "He took up the gauntlet" as a response to "How did Brian respond to Jeremy's challenge?". The second case studies situations where the speaker through a maxim (i.e. Relation) "directs" her interlocutor to the correct interpretation. For example, "There is a screwdriver next to you" could be the response for "Do you know how can I dismantle this console?". The third case includes the situations where the communicator makes use of conflicting maxims (i.e. Quantity and Quality) in order to imply that she does not have the evidence of the correct answer. For instance, "It is next to you or next to the lamp." might be the response for "Do you know where the screwdriver is?" that explains this.

Scalar implicature covers the cases where used words have conventional meanings (e.g. "some", "all", "none"). Its functionality suggests that all of the other similar but more informative utterances would be false as the communicator does not use a more powerful term (Maxim of Quantity) on the same scale intentionally. An example would be "I have repaired some of your bicycle" implying that the bicycle is not fully repaired yet. The Gricean Maxims do not affect the third category of implicatures though, the Conventional implicature, which resides in every statement's agreed meaning and is invoked through the use of the word "but". For instance, the utterance "I just had lunch but I am hungry" is a suitable example which implies that being hungry right after lunch is contradictory. Gricean cooperative principles have been shown to emerge from multi-agent decision theory, in a language task modelled using Decentralised Partially Observable Markov Decision Processes [103], and in related work conversational implicature was argued to be a by-product of agents which maximise joint utility [104].

During dialogue, people are very likely to accept implicature subconsciously as a natural attempt of cooperation, despite the fact that the discourse may have a non-cooperative character. According to Grice: "it is just a well-recognized empirical fact that people DO behave in these ways; they have learned to do so in childhood and not lost the habit of doing so" [39]. Furthermore, Ladegaard based on that states: "The Gricean approach to cooperation does acknowledge, however, that the CP may be violated. But what may appear to be an example of a non-cooperative activity is in fact, when analysed at a deeper level, to be seen as an example of cooperation, because the point of the violation is to create an implicature. This means that a seemingly irrelevant response will still be interpreted as cooperative at a deeper level, because we will look for a meaning beyond what is expressed. In other words, because the CP applies, the interlocutor knows that the speaker's seemingly irrelevant remark was made with the intention of generating an implicature" [64].

2.6.2 Gricean Maxims and Non-cooperative Dialogues

Apart from the cases of cooperative dialogue though, Grice's view of the Maxims seems to be non-applicable to those which lack cooperation. Non-Gricean behaviour is being analysed from a game-theoretical perspective [4]. According to various researchers [6, 5], Grice had little to say for non-cooperative dialogue. Hence Attardo [6] argues that the Cooperative Principle (CP) of Grice [39] should be decomposed into two distinct cooperation levels in order to locate the exact meaning of some implicatures too, which otherwise would be impossible for us to decode. The first one, locutionary cooperation (LC), expresses the amount of cooperation that the dialogue's participants must put into their utterance so that its intended meaning will be successfully deciphered. The second level, perlocutionary cooperation (PC), expresses the amount of cooperation that the dialogue's participants must put into their utterance so that its intended effect will be successfully achieved for the benefit of everyone. In other words, the PC focuses on whether the communicator's goal was achieved through her utterance or not, by serving a common good cause (by being cooperative). This is where the Perlocutionary Cooperative Principle (PCP) is based on according to Attardo (the example that he gives here is: "be a good Samaritan!"). On the other hand, deception is considered to be cooperation on the LC level and non-cooperation on the PC level [6], such as in our case that we will analyse in later chapters.

Ladegaard [64], based on dialogue examples between students and teachers, argues that sometimes the goal of the dialogue is to miscommunicate. In fact, miscommunication in terms of non-cooperation may be preferred than Gricean cooperation as it can be the core of a successful discourse strategy. According to Grice, people will naturally attempt to cooperate and produce a successful dialogue that is ultimately based on negotiable solutions. However the social context is missing from his theory. Hence, Ladegaard points out that in cases where an individual is forced into a social circumstance which requires appropriate behaviour, the conversation's failure may be the preferred option.

2.7 Conclusion

Motivated by all of the above work and findings, in the rest of the thesis we are going to implement and use Reinforcement Learning (RL) in an MDP framework to train trading dialogue agents in non-cooperative negotiations. We expect RL to outperform complex hand-crafted agents. We will create our own non-cooperative game and experiment on that, as well as on other more complex environments. We

will verify whether RL in MDPs is capable of effectively handling hidden information regarding the adversarial goals and states. We are also aware that we are going to deal with non-stationary MDPs as we will use linguistic manipulation (e.g. through implicature), inspired by concepts in Pragmatics, which will affect the adversarial behaviour. We will verify whether it will increase the performance of traditional RL, as well as focus on its detection for ethical reasons. We will include in RL information regarding the adversarial preferences and evaluate whether it can learn more successful policies based on that, given that the environment is noisy due to its non-stationary character and hidden information. In our attempt to generalise our findings and consider them applicable to negotiation domains outside trading, we will focus on using empirical analysis based on Pragmatics mainly, as well as Philosophy and Psychology. The analysis of the main research will commence in the next chapter, where we will discuss the non-cooperative trading game "Taikun" that we created in order to start experimenting with the above ideas.

Chapter 3

Initial model: Taikun

To initially investigate non-cooperative dialogues in a controlled setting we created a 2-player, sequential, non-zero-sum game with imperfect information called "Taikun". Motivated by the principle of Occam's razor and being inspired by the game-theoretical subjects that we discussed in Chapter 2, we shaped this game as simply as possible, while including key features of a non-cooperative resource trading game. Some of these features include:

- the goals' divergence
- a resource is wanted by both sides to promote competition
- there are resources that are only needed by one side to force trading as lack of communication is a common problem (we have seen this in Section 2.3.2 too)
- possibility of creation of Nash Equilibrium (discussed in Section 2.3.3) through the game's strategies
- imperfect information

The goal was also to implement mechanisms that are not restrictive for the future of this research and therefore can be flexibly extended to capture different aspects of trading and negotiation. We call the 2 players the "adversary" and the "learning agent" (LA). The two players can trade three kinds of resources (wheat, rocks, sheep) with each other sequentially, in a 1-for-1 manner, in order to reach a specific number of resources that is their individual goal. The player who attains their goal resources wins. Both players start the game with one resource of each type (wheat, sheep, and rock). At the beginning of each round the game updates the number of resources of both players by either removing one of them or adding two of them, thereby making the opponent's state (i.e. the cards that they hold) unobservable. Thus, in the long run, someone will eventually win even if none will ever trade. However, trading is highly promoted by the mechanisms of the game as its efficient use can provide faster victory.

3.1 A simple game

Taikun is a game that has been invented with the purpose of resembling any simple non-cooperative negotiation between two individuals who seek personal gain through equal unforeseen opportunities (i.e. the game's random update turn). As in a real-world scenario, some trading resources may be currently needed by either both sides (i.e. the wheat in our case) or only one. The LA does not know though which of the resources are common as the opponent's goal is hidden. However the common resource (wheat) is a secret link between their needs that emphasises the importance of the non-cooperative negotiation and enhances its competitiveness. Each Taikun game attempts to capture fragments of a generic, real-world trading session where one party only manages to achieve its goal first and then proceeds to the next session having the experience of the previous one. We consider the game as a toy model for real-world negotiations.

Initially we will examine the game's characteristics (Section 3.1.1), the actions (trading proposals and responses) that the players can use (Section 3.1.2) and those which manipulate, and the learning agent's architecture as well as those of its adversaries. We will discuss the details about the non-cooperative dialogue that we generate and the implicature that is used. Finally we will examine the history logs of played games, the learning algorithms that were used and the background of the experiments and the adversaries, that follow in the next two Chapters.

3.1.1 Game's characteristics

Important details and rules of the game are listed below:

- Sequential, non-cooperative/competitive, non-zero-sum[1] game, with imperfect information
- 2 players
- 3 different resources to trade (wheat, rocks, sheep)
- Each player starts with either 0 or 1 of each resource (for Chapter 4 only, this was changed to only 1 of each resource for both of the players in the experiments of Chapters 5 and 6)

[1]By adding up all of the players' gains and then subtracting from them all of their losses, what remains is not equal to zero. This is because of the update turn at the beginning of the round which may add or remove resources, as we will examine later in this Section.

- The goal is to reach 4 and 5 of two specific resources respectively (4 wheat and 5 rocks in the case of the learning agent and 4 wheat and 5 sheep in the case of its adversary). There is a resource (wheat) that both of the players need while the other one is needed only by one player.
- The players might start with the same kind and number of resources because the resource initialization mechanism is random[2].
- Each round consists of an update on the resources turn, the LA's trading proposal turn (and adversary's acceptance or rejection) and finally adversary's trading proposal turn (and LA's acceptance or rejection). Wins or draws are calculated at the end of the round.
- The update changes one only of the resources at random by +2 or -1. If the resource is 0 then the -1 cannot be applied and therefore the resource will remain 0. If the resource is 4 then only the +2 can be applied on it. However, due to the update rule that occurs at the beginning of each round, one of the resources may be also "capped" (please see below).
- When a resource is "capped", that is its number is 5 or more, then no change of the update rule can be applied to it. Trade can still change its quantity though.
- The update turn promotes the trades and "pushes" both of the players towards their goals too. During the update turn the actions that may occur (their probabilities are uniformly distributed) from the game are: +2 wheat, -1 wheat, +2 rocks, -1 rock, +2 sheep, -1 sheep. Only one of these 6 actions will be performed (if applicable) by the environment (game) to the players. This action does not have to be the same for both.

3.1.2 Actions (Trading Proposals)

Trade occurs through specific trading proposals that may lead to acceptance from the other player. These proposals are:

In update turn: No trading proposals or responses can be taken during this turn from any of the two players.

[2]The resource initialization mechanism is random only for the experiments of Chapter 4 though, in Chapters 5 and 6 both of the players start with 1 of each resource.

In learning agent's turn: only one "1 for 1" trading proposal may occur from the learning agent or nothing (7 actions in total), that is:

1. I will do nothing
2. I will give you a wheat if you give me a rock
3. I will give you a wheat if you give me a sheep
4. I will give you a rock if you give me a wheat
5. I will give you a rock if you give me a sheep
6. I will give you a sheep if you give me a wheat
7. I will give you a sheep if you give me a rock

The adversary responds by either saying "OK" or "No" in order to accept or reject the learning agent's trading proposal.

In adversary's turn: The same seven trading actions may occur from the adversary and the learning agent now responds in the same way by either accepting or rejecting the adversary's trading proposal.

3.1.3 Additional actions (Deception - Scalar Implicatures)

In our second (Section 4.2) and third experiment (Section 4.3) of Chapter 4 (as well as in Chapter 5 as we will see), three manipulative actions[3] have been added only to the learning agent's set of actions:

1. "I really need wheat"
2. "I really need rock"
3. "I really need sheep"

The adversary believes these statements[4], resulting in modifying the probabilities of making certain trades (discussed in Sections 4.2 and 5.2) as it will start hindering the LA's strategy by restricting resources which the LA has stated that it needs and offers the others more. Hence the manipulating actions always affect the adversary, in contrast to the trading ones which depend on the adversary's response. Note that

[3]Through the thesis these explicit manipulative actions are either expressed as "I really need X" or "I need X", where X is a resource. There is no difference between them in our work.

[4]The reasons are discussed in Section 3.3.3.

in the current model we assume that only these 3 manipulative actions potentially have an effect (explicit manipulation) on the adversary's reasoning about the game. An alternative would be to allow all the normal trading utterances to have some manipulative power (implicit manipulation), as we do from Chapter 7 onwards. For example the LA's utterance "I will give you a wheat if you give me a rock" could lead the adversary to believe that the LA currently needs rock. For the time being, we prefer to separate out the manipulative actions explicitly, so as to first study their effects in the presence of non-manipulative dialogue actions. From Chapter 7 onwards, we will examine the case where all trading proposals can cause adversaries to change their game strategy. Each one of the above three manipulative utterances imply that "I don't really need any of the other two resources", as both of the players are fully aware that three different resources exist in total and more than one is needed to win the game, and therefore they serve as scalar implicatures [104].

Our trading dialogues are linguistically cooperative (according to the Gricean Cooperative Principle [39]) and are based on Attardo's Locutionary Cooperation (LC) [6], since their linguistic meaning is clear from both sides and successful information exchange occurs. Non-linguistically though, where the communicator's intention is on focus, they are non-cooperative, since they they aim for personal goals. Hence they violate Attardo's Perlocutionary Cooperative Principle (PCP), according to which someone acts by being a good Samaritan, as is suggested by the author. Deception according to Attardo is cooperation on the LC level and non-cooperation on the Perlocutionary Cooperation (PC) level such as in our case. We will show that the LA learns how to include scalar implicatures in its dialogue to successfully deceive its adversary by being cooperative on the locutionary level and non-cooperative on the perlocutionary level. It chooses to act in that way instead of using only normal trading proposals which -despite the fact that they are non-cooperative on the perlocutionary level- lack the effect of manipulation (i.e. deception) and result on a much lower performance.

3.1.4 The Learning Agent

The game state can be represented by the learning agent's set of resources, its adversary's set of resources, and a trading proposal (if any) currently under consideration. In Chapters 4 and 5 the learning agent (LA) plays the game and learns while perceiving only its own set of resources and the turn of the game (e.g. adversary's trading proposal). Hence we track up to 19 of each type of resources, along with the turn of the game which is indicated by a binary value, and therefore we have 20 x 20 x 20 x 2 (=16,000) states. This initial state space is later extended with elements of history (previous dialogue moves in the same episode as will be

discussed in Section 3.2.3, for the experiments of Chapter 4 only). Estimates of the other agent's state (e.g. beliefs about what the adversary needs) will be included in the LA's state later as we will see[5]. The LA is aware of its winning condition (to obtain 4 wheat and 5 rocks) in as much as it experiences a large final reward when reaching this state. It learns how to achieve the goal state through trial-and-error exploration while playing repeated games. It learns how to successfully propose trades and respond to the adversary's trades.

The LA is modelled as a Markov Decision Process [97]: it observes states, selects actions according to a policy, transitions to a new state (due to the adversary's move and/or a update of resources), and receives rewards at the end of each game. This reward is then used to update the policy followed by the agent. The rewards that were used in the experiments of Chapters 4 and 5 were 1,000 for the winning case, 500 for a draw and -100 when losing a game. The winning and draw cases have the same goal states and that would initially suggest the same reward but they can be achieved through different strategies. Experiments that we have conducted using either the above rewards or the same rewards for win and draw have verified this. The learning agent's performance is slightly better when the reward for a win is 1000 and 500 for a draw.

The LA was trained using a custom SARSA(0) for the experiments of Chapter 4 and SARSA(λ) learning method [97] for the experiments of Chapter 5, with an initial exploration rate of 0.2 that gradually decays to 0 at the end of the training games. After intense experimentation with the learning parameters of SARSA(λ) we found that with λ equal to 0.4 and γ equal to 0.9 we obtain the best results for our problem and therefore these values have been used in all of the experiments of Chapter 5.

3.1.5 The Adversaries

We investigated performance with several different adversaries. As a baseline, we first need to know how well a LA which does not have manipulative moves at its disposal can perform against a rational rule-based adversary. Our hypothesis is then that a LA with additional manipulative moves can outperform this baseline case when the adversary becomes somewhat gullible, even in the cases (see experiments of the Sections 5.4.1 and 5.4.2) where it can detect them and severe penalties will be applied. A "gullible" adversary is one which believes statements such as "I really need rock" and then acts so as to restrict the relevant resource(s) from the LA (hindering behaviour). Our experiments with the restrictive adversaries (see

[5]In Chapters 8 and 9 we use learning agents which maintain information in their state representations about the adversary's preferences based on previous trades.

experiments of Section 5.3) show that this gullible behaviour may originate from sound reasoning (the results of Section 6.1.3 suggest that too). The adversary confronts in this case a very important dilemma. It suddenly does not know if it should stay with its goal-oriented strategy (baseline) or instead it should boycott the LA's stated needed resources. A priori, both of these strategies sound equally successful, and we will show that their performances are indeed very close to each other.

3.1.6 History log of the played games

A detailed history log of the played games was implemented. That allowed us to extract important information of the strategies that are followed by the learning agent and its adversary as well as check their "soundness" during implementation. Below is an example of a testing game selected at random, where the LA tests its learned policy and no learning occurs any more (agent is being 100% exploitative, using RL terms). The LA is the non-manipulative one and the adversary is the rule-based "strict" one that we will see in the next chapter (Section 4.1). The actions are not in the range of 1-7, as we saw in Section 3.1.2, but in the range of 0-6. For example, "I will do nothing" is action 0 in this case and not action 1:

New game: 3500001
Agent's action: 6, Round: 1, State: 1 1 1
Adversary's response: 1, Round: 1, State: 1 0 2
Adversary's action: 0, Round: 1, State: 1 0 2
Agent's response: 0, Round: 1, State: 1 2 0
.
.
.
Agent's action: 1, Round: 8, State: 5 4 2
Adversary's response: 1, Round: 8, State: 2 5 4
Adversary's action: 3, Round: 8, State: 2 5 4
Agent's response: 0, Round: 8, State: 4 5 2
Agent won, Round: 8, State: 4 5 2

The actions have been discussed in Section 3.1.2. The responses are "1" in case of "OK" and "0" in case of "No". The state is represented by the number of available resources (wheat, rocks, sheep).

3.2 Algorithms

The learning algorithm (custom SARSA(0)) that is used in the experiments of Chapter 4 is based on that of the tic-tac-toe game (discussed in Section A.1.2) as well as on SARSA(0). It was created for the following reasons:

- Memory efficiency.
- Learn both the trading proposal part and the response part of Taikun.
- Learn to respond to the adversarial trading proposals without taking them into consideration (e.g. include them in the state representation) during the learning process.

In Chapter 5 the algorithm is exclusively based on SARSA(λ) though. The reason is that we needed a "guaranteed" algorithm[6] which leads to more accurate policies and faster training times, as the representation is tabular, even if the memory demands are higher as we will see. We use SARSA algorithms due to the "State-Action-Reward-State-Action" sequence that they use to update the Q-values of their state-action pairs. The updates through this sequence (rather than considering only the current state-action-reward) result in very accurate policies and therefore seemed to be ideal for the stochastic character of Taikun (as well as that of Catan that we will examine from Chapter 7 onwards).

3.2.1 Similarities between the LA's first (custom SARSA(0)) and second algorithm (SARSA(λ))

Fundamental similarities between the two algorithms that were used are listed below:

- Theoretical or forward algorithms: in each observed state the learning agent looks forward in time when it comes to optimally combine future rewards.
- Tabular: all of the estimates of the states' values are saved in the computer's main memory (i.e. in an array).
- On-line updating: the updates of the state values occur on each time step, as soon as the increments of the Q-values are calculated.

[6] in contrast to our custom SARSA(0) algorithm, which was developed in the beginning of this research and therefore it is very "experimental" as we will see.

3.2.2 Differences between the learning agent's first (custom SARSA(0)) and second algorithm (SARSA(λ))

Custom SARSA(0):

- The algorithm's implementation considers the history of a round, which is the whole sequence of states-actions since the system's update turn (which occurs first in a round, Section 3.2.3), when it follows its policy while it is exploiting (in order to respond to adversarial trading proposals).
- It is based on Temporal Difference(1-step) or TD(0), with bootstrapping[7]. Update occurs by back-propagating the value to the previous state-action *only*.
- It is inspired by SARSA(0) but it is still quite different. The exact way that this algorithm works is discussed in Section 3.2.3.

SARSA(λ):

- The algorithm's implementation considers only the current observed state-action when it follows its policy while it is exploiting.
- It is based on Temporal Difference(λ), with bootstrapping as above, but the update occurs by back-propagating values to *all* of the previous state-actions that led to the LA's current state.
- It works in exactly the same way as that in Section 2.1.6. In other words, it is a SARSA(λ) algorithm as is defined by [97].

3.2.3 More details on the first algorithm's implementation (custom SARSA(0))

This algorithm (used in Chapter 4, the pseudocode is in Section A.2.3 of the Appendix) is based on SARSA(0) and the value iteration algorithm that we used in the "tic-tac-toe" game (discussed in Section A.1.2 of the Appendix). When it is exploiting, it chooses the action in a game state (or simply called here as *state*[8], discussed below) that has currently the highest value, in contrast to the case of "tic-tac-toe" where it simply locates the following state with the highest value and then

[7] All TDs have bootstrapping in contrast to Monte Carlo. That means that they update previous values based on existing estimates and they do not wait until the end of the episode.

[8] In this Section *only*, by *state* we mean the game's state (e.g. LA's turn) because we were influenced by the "tic-tac-toe" example. Hence we refer to the LA's state (which consists of its current resources and the game state id) as *LAstateId*, for example LAstate0.

it figures out what the action to perform is by comparing them. This algorithm has been further modified according to the ideas that follow.

Each one of the game's rounds includes the update turn (state 0)[9] which consists of the LA's state (LAstate0) and trading proposal (action 0). The round also includes the LA's turn (state 1) which consists of the LA's state (LAstate1) and response (action 1). Finally the game's round includes the adversary's turn (state 2), where the game's random update of the resources will occur (the LA's algorithm does not take this action into consideration though) and consists of the LA's state (LAstate2). The LA plays the game and learns while perceiving these states from its own perspective. This is how it considers and stores them in its array list. According to this perspective, the update state (state 0 or U) already contains information about the actions of the system (i.e. random changes to the LA's and to the adversary's resources) and this is where the LA proposes a trade (takes action 0) as we mentioned. This state U is the root of a "tree" which will be recorded in the array list.

The LA's turn (state 1 or G) is next. It includes information about the previous action (LA's trading proposal, action 0) that has been taken. In this state, the LA is now called to select a response action (action 1) to the trading proposal of its adversary. The adversarial trading proposal is not taken into consideration by the learning algorithm when it responds. Hence, it considers both the previous state (U) *and* the current state (G) in order to respond (while it is exploiting). In this way, which captures the history of the current round of the game, the LA gains *some* information regarding the adversary's proposal based on the resources that the LA had in the previous turn, its trading proposal there and the resources that it currently has available[10]. The LA's response action (action 1) along with the LA state (LAstate1), which both consist the state 1 or G, are inserted in the list next to the previous state-action (state 0 or U) that occurred. They are considered to be one of the four alternative states (due to the two different adversary's responses and then to the two different LA's responses) which follow after the previous state 0 (or U) that we examined. After the G state, next comes the adversary's turn (state 2 or D). It includes information about the previous action (response) of the LA. In this state the game is now called to take an action, by updating the resources of both the agent and its adversary, which eventually leads to another update state (state 0 or U). This will follow again the same reasoning that we discussed above,

[9]As we mentioned above, we do not mean in this section the LA's state in "MDP" terms. We mean the game's state or turn from the LA's perspective, which consists of the LA's state and the LA's action.

[10]In other examples (where the following idea might apply), it would be also useful to assume in this point that the previous trading proposal of the LA has affected to *some* degree the adversary's proposal (this was one of the points which influenced us to think of linguistic manipulation, discussed in Section 3.1.3). In our current case we only assume though that the LA's previous trading proposal and resources, along with the LA's current resources provides some information about how to respond to the adversary's trading proposal, without knowing what it will be.

continuously, until learning stops:

$$LAstate0 + action0 \rightarrow LAstate1 + action1 \rightarrow LAstate2$$

which is equivalent to:

$$state0 \rightarrow state1 \rightarrow state2$$

or

$$U \rightarrow G \rightarrow D$$

The learning agent stores in its array list "trees" that have always update states (state 0 or U) as their roots. After a U state, four G states follow (they are not immediately sequential). Similarly, after each G state, several immediately sequential D states (their actions are not considered by our algorithm as they are performed by the game) follow in the array list, as they result from the LA's responses to many different adversarial proposals. An example is given below:

$$|U|G|D|D|G|D|D|D|U|G|D|U|\ldots \rightarrow ArrayList$$

The formula that we used is based on value iteration, which was used in the "tic-tac-toe" example (discussed in Section A.1.2):

$V'(s) \leftarrow V(s) + \alpha * (V(s') - V(s))$ where,

α is the learning rate ($0 \leq \alpha \leq 1$), it reduces gradually to 0 at the end of the training games for the sake of convergence,

s' and s are the current and previous[11] states respectively,

V' and V are the current and previous values. Due to the fact that this algorithm actually considers the LA's state-actions though as we saw (despite the fact that we refer to them as game states, or simply states), the above formula is further inspired by that of SARSA(0) (discussed in this section). Hence the formula that the algorithm uses to update is:

$Q'(s, a) \leftarrow Q(s, a) + \alpha * (r + Q(s', a') - Q(s, a))$ where,

α is the learning rate ($0 \leq \alpha \leq 1$), it reduces gradually to 0 at the end of the training games for the sake of convergence,

r is the reward (it is set to 5 for unobserved states where there is no win, draw or loss),

s' and s are the current and previous LA's states respectively,

[11]For the algorithm's easier comprehension (as discussed in Section 3.2.3) we refer to these states as *current* and *previous* rather than the typical *new* and *current*.

Q' and Q are the current and previous Q-values of the LA's state-actions (game states) respectively.

Thus this algorithm uses a simplified formula of SARSA(0), hence the name custom SARSA(0) as it uses the sequence state-action-reward-state-action to update too. As we have also seen in Section 2.1.3, the learning agent according to what happens in the game assigns values (which in our case are Q-values of the LA's state-action pairs) to the above states and then it back propagates them to their previous states. In detail, when the LA updates on its trading proposal turn, then the algorithm locates the same U state[12] in the list and back-propagates the values to the previous state (D), which is somewhere in the list (it normally performs an exhaustive search to find it). In the case where the LA updates on its response turn (adversary's trading proposal), then it locates the same G state in the list, along with the same U state which preceded the G state, and back-propagates the values to the previous state (U). In state D, which is the next one, according to the result of the game (i.e. win, draw, nothing so far) the corresponding reward is back-propagated to both of the previous states G and U (this is why it resembles SARSA(0), in some cases it considers the state-action-reward-state-action chain when it updates, as we saw in Section 2.1.6).

When it proposes a trade while exploiting, it selects the action whose state has the highest value in the list (policy), by checking for identical beginnings ("roots") of "trees" (states 0 or U). When it responds to an adversarial trading proposal while exploiting, it selects the action whose state (G) has the highest value in the list (policy) by checking for two first identical parts of the "trees" (state 0 or U and state 1 or G), in other words the history of the round. Thus, every state (which is in fact a typical RL state-action pair as we have mentioned before) has a value (Q-value) indicating its importance in the game. The agent learns to pick only the highest in value game states, in order to find the best action to take, and that's how it learns various successful strategies. The e-greedy behaviour of the learning agent is based on a gradual reduction in the exploration's percentage -game after game- that resulted in doubling its learning performance as we will see in the next chapter. The rewards (converted to game state values or even better, Q-values of LA's state-actions) that were used were 1,000 for the case of the learning agent's win, 500 for the draw and -100 for the case of a loss. Checks for possible win are performed at the end of state 2 (D). As the number of the training games reached very high amounts (up to 3.5 million games) the graphs that were produced at the end of testing (discussed in the next chapter) are based on cycles of games as we will see. Each one of these cycles refers to a specific predefined (in the code) amount of training games.

Some more details that help to understand the algorithm which can be found in

[12]To clarify again, by same *state* we mean the same game's state, which consists of the same LA's state and same action.

the Appendix (Section A.2.3) are:

- In the greedy cases for state 1 (G), we still investigate whether or not the state 1 exists in the list after the action 0 (LA's trading proposal), because the adversary might or might not have accepted the LA's trading proposal. If it does not exist then the algorithm instead explores, and then adds the state (LA's state along with the action) to the list.
- The actions that can be either performed at random (because of exploration) or greedily (because of exploitation) require availability of a resource to be checked. That means a random action that cannot be performed because the (giveable) resource is not available will be generated at random again and again until the resource is available. In this way, when the agent greedily will select an action from an identical state in the list, that action will always refer to a resource which is available.

3.2.4 Details and parameters of the second algorithm's implementation (SARSA(λ))

The second algorithm is exclusively based on the typical implementation of SARSA(λ) that we saw in Section 2.1.6, shaped in a way that allows the sequential flow of Taikun's game-play[13]. As we have discussed there, in SARSA(λ), λ is the eligibility trace, α is the learning rate and in combination with λ they alter together the learning process according to the problem's nature. In our algorithm a is gradually being reduced to 0 during the training process to provide flexibility to the learning process and eventually force convergence. The LA's state consists of its resources and the game's turn (e.g. LA's turn). The actions are discussed in Sections 3.1.2 and 3.1.3.

By studying Sutton and Barto book [97] (in particular Chapter 7) we concluded that we should set λ to 0.4, mainly due to our game's stochastic level that resembles that of relevant examples of the book, such as the "19-state random walk task" (in Sutton and Barto book[14] [97] in Chapter 7). The examples there suggest that low λ values should be used with problems which include stochastic actions (e.g.

[13]The adversary's trading proposal turn follows that of the LA. That means the LA learns how to propose trades and how to respond to trades. These actions are discussed in Section 3.1.2.

[14]The book's graphical representations (Figures 7.6, 7.9, 7.17) of various Temporal-Difference algorithms in the "19-state random walk task" example suggested that we should probably set λ to 0.4. These graphical representations showed the root mean-squared errors (RMSE) obtained from different λ and a values. The errors represented the difference between the states' true values and those that the learning methodologies have found. λ equal to 0.4 seemed to be a good overall choice for our case.

"19-state random walk task"). On the other hand, high λ values should be used in cases where the actions are deterministic (e.g. 0.9 in the grid-world example, Figure 7.12 of [97]). Our experiments in Section 5.6 with different tested values of λ suggested that λ equal to 0.4 was indeed successful. As Taikun's stochasticity is high, due to the actions and the system's random update, λ equal to 0.4 was considered to be a wise initial choice, and especially in our baseline case, where the learning agent's six trading actions (proposals) are stochastic (their effect is based on the adversary's response). From Chapter 7 onwards we will use a higher value of λ though, which will be 0.9 as we will see. We will do that because all of the normal trading proposals will also manipulate the opponent, and the manipulation effect is deterministic (always affects the adversary as we saw in Section 3.1.3).

3.2.5 Advantages and disadvantages of the two algorithms / Results

We have seen in Section 2.1.2 that a state has the Markov Property (in an MDP) by containing all the information needed to choose the next action. Our first (custom SARSA(0)) algorithm "extends" the above statement by considering sometimes a part of the whole round[15], or in other words the history of a round, during learning (as we examined in Section 3.2.3). The main reason that we did that though was to confront the problem of learning to respond to adversarial trading proposals without considering them (i.e. representing them in the LA's state). If we would represent them in the LA's state the problem might become hard and unsolvable for RL, and therefore we wanted to deal with it as simply as possible at the beginning of this research. Another reason that the custom SARSA(0) algorithm sometimes considers the history of a round, was to make the learning process as accurate as possible, as the round's history provides more information to the learning process than a single game's state (turn). The round's history is taken into consideration by the LA while it is exploiting or updating whenever it responds, as we discussed in Section 3.2.3. The updates after a win, loss, or draw are also back-propagated through this history. Furthermore, as Taikun consists of the LA's trading proposal turn, where the LA chooses one of the seven actions to take, and the adversary's trading proposal turn, where the LA chooses one of the two responses to take, an effort was made with both of our algorithms to make one policy only (instead of two, one for the trading proposals and one for the responses) which would successfully propose trades and respond to adversarial trading proposals too.

The SARSA (λ) implementation (our second algorithm) is fully consistent with

[15]In Taikun each round consists of the update turn, the LA's turn and the adversary's turn.

the Markov Property though. Each state contains all of the information needed to chose the next action. Hence the hypothesis now was that this is enough for our problem and it may outperform our first algorithm. SARSA (λ) required all states s and actions a initially to be defined (included in the list) while our first algorithm adds them in the list while playing the game. That means in SARSA (λ) we give an approximation that might be wrong, it might require more (or less) memory than it is really needed. Our first algorithm seems to be better on that aspect as it works better in problems where we cannot pre-define the initial state action space. It dynamically adds states actions in the list while observing them for the first time. In contrast to SARSA (λ), it does not *always* wait for the next action of the next state to occur when it comes to update but once the next state is reached then the value of the state (game's state where we consider the LA's state along with an action) is back-propagated. From Chapter 7 onwards though, where the Catan experiments are discussed, the SARSA(λ) algorithm has been further modified to dynamically add state-action pairs in the list while observing them for the first time. Hence it uses the exact amount of memory that is really required.

Our custom SARSA(0) requires less memory to run the training games, despite the fact that the memory management that it performs is not effective (as those "trees" of alternative game states with actions contain states and actions which are repeated in other "trees"). Even an old desktop PC though with 4GB RAM was able to provide results in all of our cases that will be examined in Chapter 4, where the number of the training games reached 3.5 million. The results were slightly worse than those that SARSA(λ) produced though and required much longer times. The long running times was the main reason that our second algorithm was implemented. It is faster in every case but it requires higher amounts of RAM. This is mainly due to its back-propagating mechanism which updates all of the state-actions that led to the current state, constantly requiring a long temporary list for that to hold them. Another reason was that it created at the beginning of the training a list including a large number of (estimated) states and actions[16]. It learns faster (fewer training games and time) and the results are slightly better than those of our first algorithm.

The first algorithm is a "lite" version that could solve similar in complexity problems (up to this point) requiring only a desktop PC with low RAM (6GB RAM is recommended). Our second algorithm, given that a high RAM machine is available (at least 20GB of RAM), performs slightly better and it is much faster. To conclude by introducing numbers, our first learning agent (custom SARSA(0)) wins more often than the strict adversary (the one which was used only in Chapter 5[17])

[16]As we have mentioned, this from Chapter 7 onwards changed to a dynamic memory allocation method, which adds state-action pairs in the list while observing them for the first time.

[17]The strict adversaries of Chapter 4 and 5 are not exactly the same as we will see. The adversary of Chapter 5 is more difficult to beat because it has additional rules. Hence we compare here both of our learning agents (custom SARSA(0) and SARSA(λ)) against the strict adversary of Chapter 5.

after approximately 350,000 training games. Our second learning agent (SARSA(λ)) requires approximately 100,000 training games. The manipulative cases require approximately 35,000 training games from the custom SARSA(0) agent, to start winning more often, and approximately 10,000 training games from the SARSA(λ) one.

3.2.6 Q-Learning and Value iteration not suitable for Taikun

In Q-Learning we need to initially apply rewards to the actions of all those states that would immediately either reach a goal state (wins, which could be also draws) or other non-desired states (for example defeats). In Taikun, this is not possible as it is not clear to the LA when the defeats occur by considering only its own state. Moreover, we can only approximately determine when the wins or the draws will occur, as a win could be in fact a draw. Winning states can be states of draws too, as both the LA and the adversary can be in winning states on the same time point (turn) and this is considered by the game's rules to be a draw. The same problems apply to the Value iteration algorithms too. Hence, both of these Reinforcement Learning methods were considered to be not suitable for a learning agent in Taikun and therefore we concluded on a SARSA(λ) algorithm. With this algorithm (as with our custom SARSA(0) too) we could still back-propagate the reward from each of the game's final events (e.g. loss) rather than from a single state. Other reasons that convinced us to use SARSA have been mentioned in Section 3.2.

3.3 Experiments background

In this part we will discuss the background of the Taikun experiments that will follow in Chapter 4 and 5. Experiment 1 and Experiment 2 refer to the first and second experiments of the Chapter 4 and 5 as they are similar in logic. First the RLA plays against an adversary which follows a strict rule-based strategy (baseline). Then we include manipulating moves in its action set and the RLA plays against the adversary which starts by following the strict strategy but it is also susceptible to the RLA's manipulating actions. All of the improvements in performance that will be reported in the thesis are absolute unless stated otherwise.

3.3.1 Adversary's strategy in Experiment 1 / Baseline strategy

This strategy was designed to form a challenging (strict) rational adversary for measuring baseline performance and the details of the experiment will be discussed in Chapters 4 and 5. It cannot be manipulated at all, and non-cooperative dialogue moves will have no effect on it – it simply ignores statements like "I really need wheat". The strict rule-based strategy of the adversary will never ask for a resource that it does not need (in this case rocks). Furthermore, if it has an available non-goal resource to give then it will offer it. It only asks for resources that it needs (goal resources: wheat and sheep). If it does not have a non-goal resource (rocks) to offer then it offers a goal resource only if its quantity is more than it needs, and it asks for another goal resource if it is needed (this applies only to the strict adversary of Chapter 5, therefore making the strict adversary more difficult there).

Following the same reasoning, when replying to the LA's trading proposals, the adversary will never agree to receive a non-goal resource (rock). It only gives a non-goal resource (rock) for another one that it needs (wheat or sheep). It also agrees to make a trade in the special case where it will give a goal resource whose quantity is more than it needs for another one that it still needs (as above, this applies only to the strict adversary of Chapter 5). This is a strong strategy that wins a significant number of games. In fact, it takes about 100,000 training games before the LA (based on our SARSA(λ), Section 5.1) is able to start winning more games than this adversary, and a random LA policy loses 66% of games against this adversary. It is a strategy that has the potential to win. Our learning agent is not fully aware of its own goal state and by interacting with the environment (the adversary is a part of it) tries to form an optimal policy (winning strategy). On the other hand, the adversary is fully aware of its own goal and rational trading actions and that makes its strategy initially dominating.

3.3.2 Adversary's strategy in Experiment 2 / Manipulated strategy

The adversary in Experiment 2 of Chapter 4 (and Chapter 5 as we will discuss) retains the above strict baseline policy but it is also susceptible to the non-cooperative moves of the LA. For example, if the LA utters "I really need rock", weights of actions which transfer rock from the adversary will decrease, and the adversary will then be less likely to give rock to the LA. Conversely, the adversary is then more likely to give the other two resources to the LA. In this way the LA has the potential

to mislead the adversary into trading resources that it really needs. Details of these experiments will be discussed in Chapter 4 and 5, where the same logic applies.

In detail, the adversary's action set includes only actions (0-6)/responses (0-1) and their manipulation weights. Its goal is to perform actions (or responses) that have the highest weights. These weights are being modified by the three manipulative actions of the agent. The adversary fills in the actions' list with all of the possible actions/responses along with their weights and as long as the learning agent uses the three manipulative actions, these weights change accordingly (details in Sections 4.2 and 5.2)[25] [26].

3.3.3 Why is the adversary's manipulated behaviour based on sound reasoning?

In this section we will examine the reasons that might explain the adversarial "gullibility" and hindering behaviour as in a real world trading scenario. The adversary believes the learning agent's manipulating utterances (i.e. "I really need sheep") and therefore trades only those resources that aren't currently stated as needed (i.e. wheat and rocks) in order to make things harder for it (hinders the LA) and eventually win the game. The reasons behind this behaviour are:

1. Our initial assumption: The adversary is gullible by nature. For example, previous experience (familiarity) or even a social bond might explain that, such as friendship where there is trust.

2. Based on our results from the restrictive adversaries (experiments in Section 5.3): The adversary confronts a very important dilemma ("should I keep following my goal or should I now take advantage of my opponent's need?"). 50% of the people who participated in a related experiment have this dilemma (as we will see in Section 6.1.3). The adversary then makes the -rational- decision (which is actually a very successful strategy as we will show in our restrictive adversaries experiments in Section 5.3 too) of starting to restrict (boycott) the LA's needed resources to make things even harder for it.

3. Based on Pragmatics: The LA uses scalar implicature to its adversary (i.e. "I really need wheat"). The adversary is very likely to accept that subconsciously as a natural attempt of cooperation from the LA despite the non-cooperative character of the game. As we have seen in Section 2.6.1 too, according to Grice: "it is just a well-recognized empirical fact that people DO behave in these ways; they have learned to do so in childhood and not lost the habit of doing so" [39]. Furthermore, Ladegaard based on that states: "The Gricean approach

to cooperation does acknowledge, however, that the CP may be violated. But what may appear to be an example of a non-cooperative activity is in fact, when analysed at a deeper level, to be seen as an example of cooperation, because the point of the violation is to create an implicature. This means that a seemingly irrelevant response will still be interpreted as cooperative at a deeper level, because we will look for a meaning beyond what is expressed. In other words, because the CP applies, the interlocutor knows that the speaker's seemingly irrelevant remark was made with the intention of generating an implicature."[64]

4. Based on other non-cooperative games: Opponent models (OM) with hindering abilities (i.e. the adversary's boycott) have previously been shown to be important in non-cooperative negotiation games such as the "Machiavelli" card game [8].

Hence the adversary's manipulated behaviour due to the interlocutor's implicature is observed in humans too and the findings of the successful RL trading policies that the agents have learned can be used in real-world trading negotiations. As scalar implicature can be used in other contexts too and not only during trading, it is suggested that those findings might be effective in a variety of non-cooperative negotiation environments (e.g. a political debate) where such a "gullible" behaviour can be expected for the above reasons and may lead to a hindering behaviour. Those findings in regard to *when* and *how* should linguistic manipulation (i.e. implicature) be used in order to be effective, will be examined in the experiments that will follow later in the thesis (Chapters 4, 5 and 8).

3.3.4 Restrictive adversaries

In Chapter 5 we investigate performance against adversaries which cannot be manipulated, but their strategy is to always restrict the LA from gaining a specific type of resource. We need to explore how well a manipulated adversary (for example one which will no longer give rocks that only its opponent needs) performs. This will show us the potential advantage to be gained by manipulation and most important, it will generalise our problem by showing that the restriction (boycott) of a resource that only the opponent needs, or of a resource that both of the players need, are actually reasonably good strategies compared to the baseline case. Hence, the manipulated adversary has indeed a reason for choosing to restrict resources (hinder the LA) rather than staying with its rule-based strategy (as we mentioned in the previous section). In other words it has a rational reason to become gullible and fall in the learning agent's trap.[25]

3.3.5 Exposing (detective) adversaries

In Chapter 5 we also extend the problem to include possible negative consequences of manipulative LA actions. The adversary begins each game with a probability of detecting manipulation, which increases after every one of the LA's manipulative actions. In more detail, every time the LA performs a manipulation, there is an additional chance that the adversary notices this (starts at 1-in-10 or 1-in-20 and increases after every manipulative move, up to 100% in the case of the 10th or 20th respectively manipulative attempt). The consequences of being detected (exposed) are either that the adversary will refuse to trade with the LA any further in that game, or that the adversary automatically wins the game. In the former case the LA confronts a medium penalty of exposure as the game still goes on while in the latter case the penalty is the highest possible (immediate loss). In these two cases there is always a high risk associated with attempting to manipulate, and the LA has to learn how to balance the potential rewards with this risk [25].

3.3.6 Hidden Mode MDP triggered by manipulative actions

In the Background Chapter 2 we discussed Non-Stationary Markov Decision Processes [17, 91] and analysed relevant work. Our case of learning against a gullible adversary which starts by following the baseline strategy and then switches to that of the restriction, whenever a manipulative action occurs, is a similar case. This adversary demonstrates different behaviours. Thus the same state in our game is studied from different perspectives because we have different MDPs (with different dynamics). A reasonable question is: "What is the best action for the LA to choose when it does not 'know' in which MDP it is?" In other words, the LA does not take into consideration when the manipulation occurred[18]. It just averages both of these MDPs when it is about to select the most optimal action and it still successfully learns in this non-stationary environment. Hence one important difference in our case is that manipulative actions change the mode, while in HMMDPs [17] modes change stochastically and are independent of the control system's response. However in our case, the end of turns (LA's proposal - adversary's proposal) change the mode too, and this is now independent of the control system's response, as in HMMDPs. Taikun in fact consists of two "games": the LA's turn where the LA proposes trades and the adversary responds and the adversary's turn, where the adversary proposes trades and the LA responds. The LA successfully learned how to propose trades

[18]We preferred to see whether or not tabular RL would be capable of learning in this case, rather than considering the manipulation in the state representation.

and how to respond to trades as we will see, in this non-stationary environment.

3.3.7 Hybrid strategy

The adversary's algorithm here considers both of the outcomes of the baseline and the manipulated (i.e. restriction of resources) strategies, that were previously discussed, when it comes to decide which response to give and which action to perform (as we will see in detail in the corresponding experiments of Chapter 4). If the two actions of the two different strategies agree on the offered and/or the wanted resource (i.e. the resources are the same) then the manipulated action will occur. If not, then "Do nothing" will be chosen. The actions are:

1. Do nothing
2. Give 1 wheat for 1 rock
3. Give 1 wheat for 1 sheep
4. Give 1 rock for 1 wheat
5. Give 1 rock for 1 sheep
6. Give 1 sheep for 1 wheat
7. Give 1 sheep for 1 rock

If the action of the strict, goal-oriented strategy (Experiment 1 of Chapter 4) is, for example, 2 and the action from the manipulated strategy (Experiment 2 of Chapter 4) is also 2 then the adversary will decide to perform action 2 because both the offered and the wanted resources are the same (therefore these 2 actions agree). If instead the actions were 2 and 7 respectively then the adversary would perform the second action (i.e. action 7) as they both ask for the same material (rock) from the agent and *the logic of the manipulated strategy has always a higher impact and will be chosen.* That means it would perform action 7. On the other hand, if the actions were 2 and 3 respectively, then the agent would again perform action 3 as they both have in common the offered material (wheat) and the second action (from the manipulated strategy) has always a higher impact on the adversary's decision as we have already mentioned. From these examples we understand that there are specific groups of actions where the agent would definitely perform an action (i.e. that of the manipulated strategy). However, if the actions of the first (goal-oriented) and second (manipulated) strategies do not belong in the same groups 1, 2 or 3 which are based on the offered resource, or 1, 2 or 3 which are based on the wanted resource (as can be seen below) then the adversary will do nothing.

Groups of actions:

-based on the offered resource-

1. (a) Give 1 *wheat* for 1 rock
   (b) Give 1 *wheat* for 1 sheep
2. (a) Give 1 *rock* for 1 wheat
   (b) Give 1 *rock* for 1 sheep
3. (a) Give 1 *sheep* for 1 wheat
   (b) Give 1 *sheep* for 1 rock

-and based on the wanted resource-

1. (a) Give 1 wheat for 1 *rock*
   (b) Give 1 sheep for 1 *rock*
2. (a) Give 1 wheat for 1 *sheep*
   (b) Give 1 rock for 1 *sheep*
3. (a) Give 1 rock for 1 *wheat*
   (b) Give 1 sheep for 1 *wheat*

In other words, we have 6 groups of "agreement" between the actions 2-7 (shown below). Action 1 is not included because it is "Do nothing". 3 of the groups are based on the offered resource and 3 are based on the wanted resource according to the syntax: (strict strategy action, manipulated strategy action). In detail we have:

1. (2,3) where 3 would be chosen, or (3,2) where 2 would be chosen
2. (4,5) where 5 would be chosen, or (5,4) where 4 would be chosen
3. (6,7) where 7 would be chosen, or (7,6) where 6 would be chosen

   and

4. (2,7) where 7 would be chosen, or (7,2) where 2 would be chosen
5. (3,5) where 5 would be chosen, or (5,3) where 3 would be chosen
6. (4,6) where 6 would be chosen, or (6,4) where 4 would be chosen

that is 12 sets.

The adversary's responses have again the same logic. If both of them are "OK" for a common offered and/or wanted resource of the LA's trading proposal then the adversary will perform accordingly, for the sake of the manipulated strategy. If not, then the adversary will just say "No" to the agent's trading proposal (action).

3.4 Conclusion

In this chapter we discussed in detail our game "Taikun", presenting its characteristics and actions. We also examined the mechanisms of our LA's and adversaries in the experiments that will follow. The LA learns how to exploit its adversary's trading actions in order to win most of the games in all three of our different kinds of experiments in Chapter 4. The adversary there follows a rule-based strict strategy in the first experiment. Then its strategy can be affected by three additional manipulating actions of the LA in the second experiment. In the third one, the LA manages to win most of the games versus the adversary which uses the hybrid strategy that we saw previously. Following a similar reasoning in regard to manipulation, the experiments of Chapter 5 show again that the LA successfully learns how to win most of the games. We conduct experiments there where the adversaries can also restrict particular resources. Others can even expose (detect) the LA's manipulation by applying severe penalties as we will examine.

Chapter 4

Experiments in Taikun: Manipulation

All of the experiments of this chapter are based on our custom SARSA(0) learning algorithm as we have mentioned earlier. The first experiment of Section 4.1 was conducted (as we have discussed in Chapter 3) in order to examine whether or not our learning agent would manage to learn how to defeat the strict adversarial rule-based strategy in most of the games. In general, the adversary in this experiment does not accept a resource that it does not need (in this case rocks or another that its number has reached the goal number) and it offers only the unwanted resource (rocks) in order to receive another that it needs. It is a strategy that has the potential to win. Our learning agent is not aware of its goal state and by interacting with the environment (the adversary is a part of it in our MDP setting) tries to form an optimal policy (winning strategy). On the other hand, the adversary has a goal-directed strategy and that makes its position easy.

In the second experiment of Section 4.2 the adversary believes the learning agent's additional trading manipulative proposals and tries to restrict the resources that the learning agent has declared as "needed" from it. It was very important here to investigate whether or not the learning agent would learn how to use these manipulative actions in order to deceive its adversary and "direct" it where it wants. The third experiment of Section 4.3 examines the case where the adversary combines the logic of the two previous strategies and, still being manipulated, reaches a decision only if the two strategies agree up to a point. In this case it follows the manipulated trading proposal or response. Otherwise, it does nothing as we have seen previously.

Statistical significance results are discussed and presented at the end of this chapter. As this is the first chapter where experiments are presented, much detail will be given on our methodology to give the reader a clear view of our reasoning. For instance we will see that several, small in number training games always start

first to check the soundness of our algorithms, or that there is always a number (usually 20,000) of testing games that follows the training ones in order to check our policy. As the experiments' general methodology remains the same in the next chapters, some details will not be mentioned again there for the sake of space and avoiding repetition.

4.1 Strict adversary

The first experiment (also called Experiment 1) started by running specific (initially low, such as 1k) numbers of training games in order to learn a policy for the learning agent and then we applied that on an even smaller number of testing games. The adversary's algorithm was based on the hand-crafted strict strategy (studied previously in Section 3.3.1) in all of the parts of this experiment. As a first step, starting with small numbers of training games (that gradually reached 160k, Figure 4.1 and 4.2) we quickly realised that the adversary's strict strategy was successful enough to provide it with a big difference on its winning performance over that of the learning agent's.

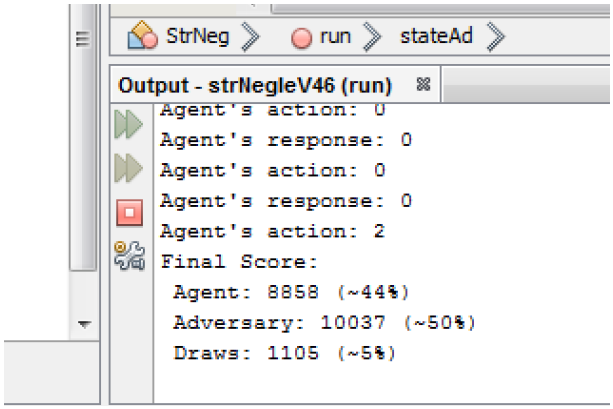


Figure 4.1: *Learning Agent's and Adversary's performance in 160,000 training games of Experiment 1. Trained on 160,000 games and then tested on 20,000 more our learning agent's performance is* 6% *worse than that of its adversary. Screenshots will be provided in similar cases as they refer to the actual research environment where the experiments were conducted and offer a part of the original experience.*

However, the adversary's performance was becoming worse while the number of the training games were increasing. The second step led us to 1.6 million training games. At that point we noticed that the learning agent had finally managed to increase its winning performance to 48%, resulting in a 2% positive difference now for the first time from its adversary which scored 46%. The graph had a similar but smoother view as we can see in Figure 4.3.

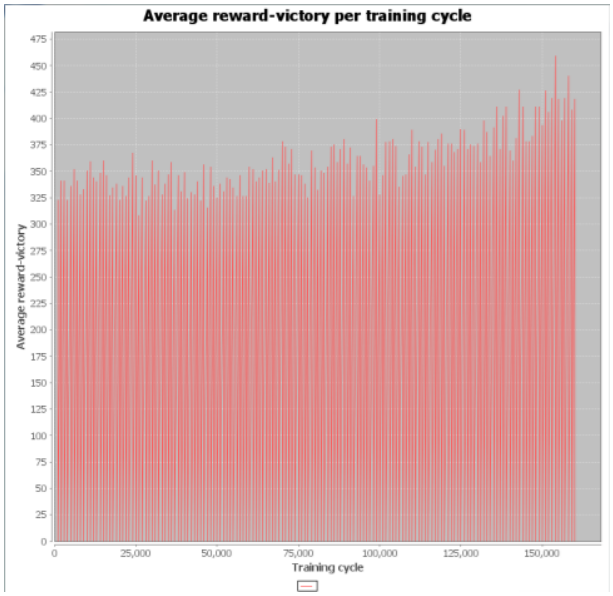


Figure 4.2: *Learning Agent's reward-victory graph in 160k training games of Experiment 1. The learning agent wins more and more training games over time by receiving better rewards on each training cycle.*

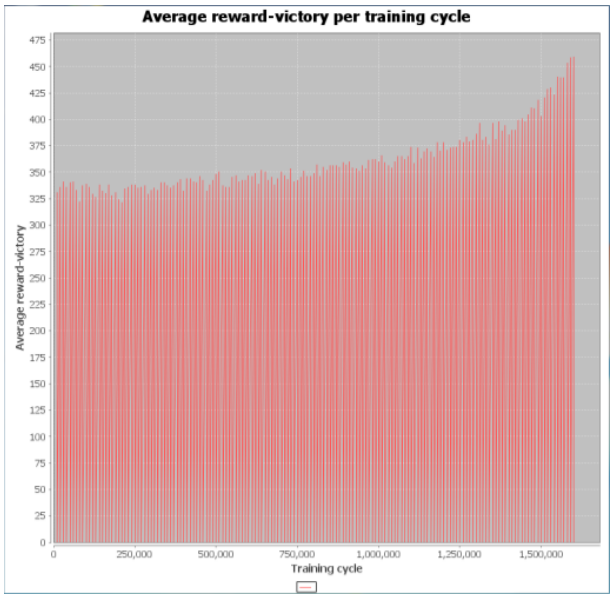


Figure 4.3: *Learning Agent's reward-victory graph in 1.6 million training games of Experiment 1. Similar but smoother trend compared to that of 160k games above.*

During the third step, the difference between the winning performance of the learning agent and that of the adversary was gradually increasing for the sake of the LA, as the number of training games was increasing and we stopped at 3.5 million training games (Figures 4.4 and 4.5). The learning agent scored 49%, having a positive difference of 5% compared to that of the adversary, but the compiling times and the memory requirements reached very high levels and therefore set an initial obstacle.

4.1.1 Changing the exploration rate

As a fourth step, considering the above results thoroughly we concluded that the e-greedy ratio (discussed in Section 2.1.1) of the agent could be altered in a way that would make the learning process faster. Thus we decided to gradually decrease the exploration percentage, game after game, from 20% (which was constant) to 0. This method, with a hope of locating a plateau in the average reward-victory graph, indeed provided us with better results throughout all of our tests in the first experiment.

In detail, at only 350,000 training games and 20,000 more testing games the learning agent managed to win by 3% more often than its adversary. It is important here to mention again that a similar winning difference in their performances has been previously observed at 1.5 million games (that was in fact even less, 2%), before the exploration's gradual reduction took place. A detailed screen shot is provided in Figure 4.6 with the results and the graph.

Similarly, the tests gradually reached again 3.5 million training games where the difference between the learning agent's winning performance and that of its adversary's increased to 9%, in contrast to 5% that used to be before the exploration's gradual reduction took place. Results and screen shots are given in Figure 4.7 and 4.8 respectively. The exploration's gradual reduction almost doubled the winning performance of the learning agent (reached approximately 10%) but it has also doubled the compiling times in all of the various tests that have been conducted as part of the first experiment. Hence, the running time of the 3.5 million training games it requires now almost a day while it used to be approximately 12 hours before the exploration's reduction, in a machine with 4GB of RAM.

Throughout all of the stages of the first experiment we were also collecting data regarding the learning agent's actions and responses in order to retrieve detailed

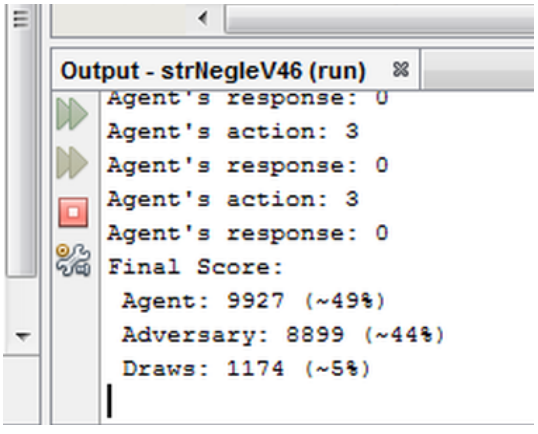


Figure 4.4: *Learning Agent's and Adversary's performance after 3.5 million training games of Experiment 1. Trained on 3.5 million games and then tested on 20,000 more our learning agent's performance is* 5% *better than that of its adversary.*

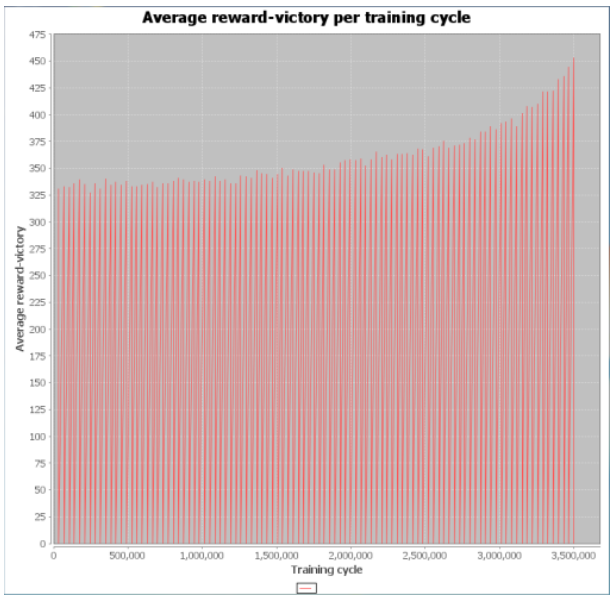


Figure 4.5: *Learning Agent's reward-victory graph in 3.5 million training games of Experiment 1.*

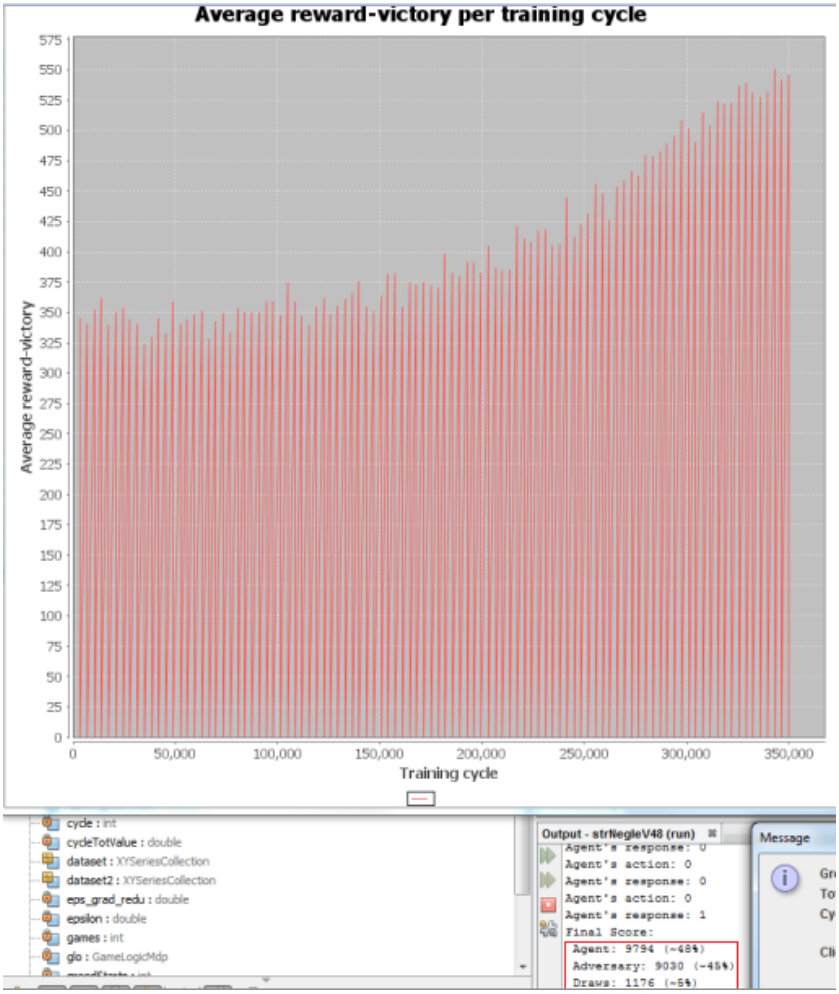


Figure 4.6: *Learning Agent's reward-victory graph in 350 thousand training games of Experiment 1. The LA won 3% more games than the Adversary in 20k testing games that followed after training.*

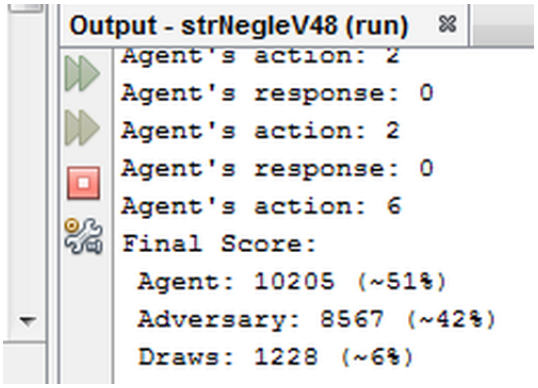


Figure 4.7: *Learning Agent's and Adversary's performance in 3.5 million training games of Experiment 1 with gradual decrease of the exploration rate. Trained on 3.5 million games and then tested on 20,000 more our learning agent's performance is* 9% *better than that of its adversary.*

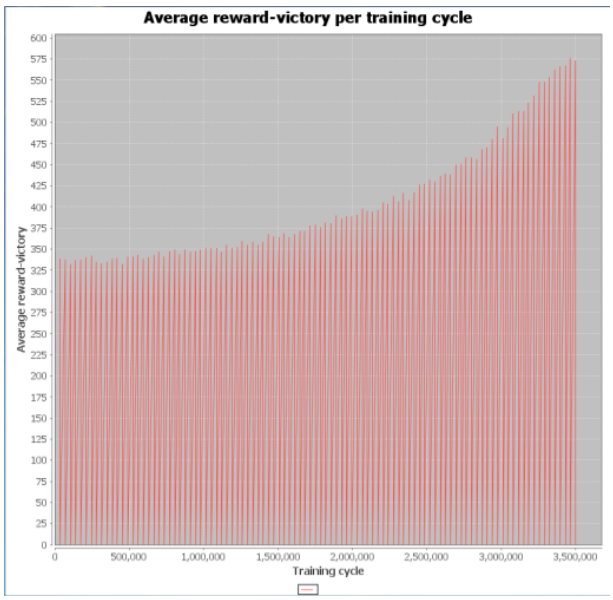


Figure 4.8: *Learning Agent's reward-victory graph in 3.5 million training games of Experiment 1 with gradual decrease of the exploration rate.*

information about the exact strategies that it uses. Therefore we have gathered action datasets from each testing phase. Surprisingly, we found out that in all of our tests so far the learning agent insists on using one specific strategy that becomes clearer as the number of training games increases. In detail, the learning agent uses as little as possible the actions 6. and 7., restricting the sheep resource from its adversary as much as possible and on the other hand it proposes a suitable trade that requires it (action 3.). Here it is important to remember that only the adversary needs 5 sheep to win. This successful, unique strategy might be part of a Nash Equilibrium (discussed in Section 2.3.3) and provide a unique solution to the adversary's constant, rule-based strict strategy. Evidence for that is provided in the figures below after 3.5 million training games.

The learning agent's strategy focuses on restricting the sheep from the adversary

Learning Agent Actions	Action's frequency
1. Do nothing	113617 (34.4%)
2. Give wheat for rock	19555 (5.92%)
3. Give wheat for sheep	**64347** (19.48%)
4. Give rock for wheat	62017 (18.78%)
5. Give rock for sheep	54089 (16.38%)
6. Give sheep for wheat	**10826** (3.28%)
7. Give sheep for rock	**5787** (1.75%)

Table 4.1: *Frequencies of actions of the learning agent (Exp.1), in 20k testing games, after 3.5m training ones.*

as we can see on the Table 4.1 (for example $action3 > action6 > action7$). It is also interesting to notice the lower frequencies (than that of action 3) of the actions 4. and 5. which result to rejection (adversary never accepts rock), therefore having the same effect as 1 ("Do nothing"). These results are illustrated in the pie chart of Figure 4.9.

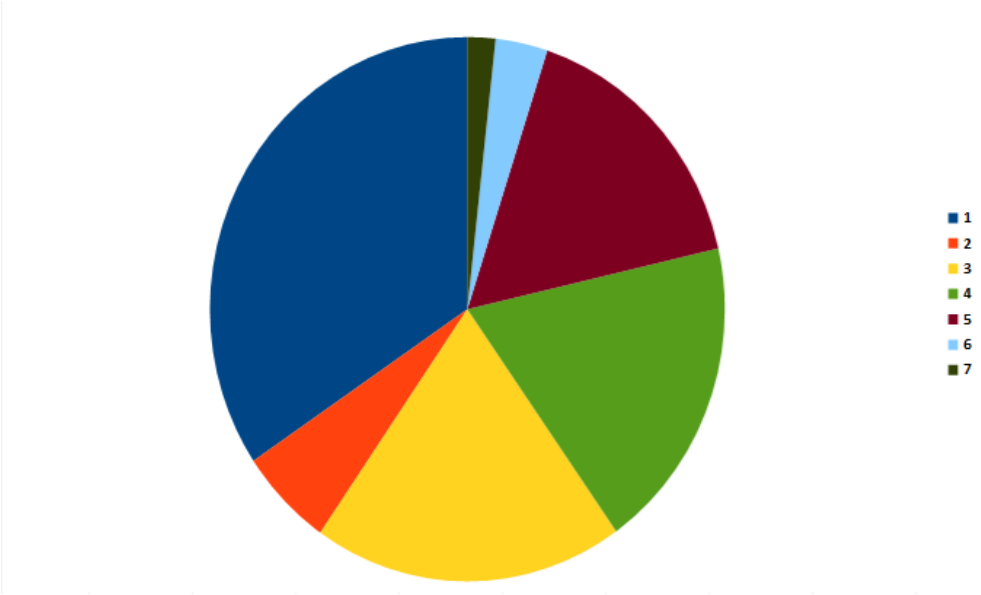


Figure 4.9: *Learning Agent Actions Pie Chart in Experiment 1*

Learning Agent Actions Priority

1-3-4-5-2-7-6

Learning Agent Responses

1. No

2. Yes

Learning Agent Response Frequencies

1. 252368 (83.77%)
2. 48894 (16.23%)

Learning Agent Response Priority

1-2

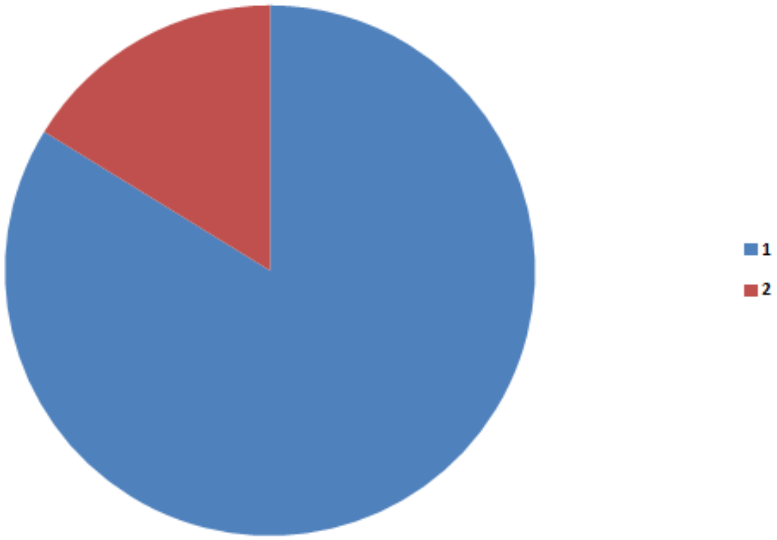


Figure 4.10: *Learning Agent Responses Pie Chart in Experiment 1*

The rejections of the adversary's trades dominate the acceptances as our learning agent wants (learns) to mainly deny the trading actions of its adversary and prohibit its strategy.

4.2 Manipulation

We started this experiment (Experiment 2) by including three manipulative actions in the set of the learning agent's trading actions (as we have discussed in Sections 3.1.2 and 3.1.3). The adversary, being affected and deceived by those manipulative actions, changes the probabilities of its own actions and responses accordingly in order to provide the learning agent with resources that it believes the LA does not need. In other words, the adversary tries to boycott (restrict) resources that the LA stated that it needs by not offering them. In more detail, the probabilities (weights) that represent the adversary's willingness to give (i.e. through trading proposals or responses) each of the three resource types start equally (each one is 33.3%) at the beginning of the training games. When the LA uses one of the three manipulative actions (e.g. "I need sheep") then the adversary's probability of giving the LA's *needed* resource (e.g. sheep) decreases by 5%[1] and the probabilities of offering the other two resources increase accordingly (i.e. equally, each one increases by 2.5%). Therefore the adversary is slightly less likely to offer sheep (or respond "yes" to a trade that asks for sheep) to the LA. The more the LA keeps asking for a particular resource the less likely it is for the adversary to offer it -but instead- it will probably offer one of the two other resources. It does that through corresponding trading proposals or responses to the LA's trading proposals.

The first step of this experiment started by running 350,000 training games and then 20,000 more testing games. It was important for us to investigate whether or not the agent's learning algorithm would be able to learn successful strategies in order to win more games than its adversary. A significant question here for us was: would the learning agent be able to learn how to use its manipulative actions as part of a strategy in order to "direct" its opponent to a desirable way of thinking? The results of the first step were promising. Below (Figure 4.11 and 4.12) are the details of the performances and the graph, as were similarly used for the Experiment 1 too.

As we had no solid indication that the algorithm was learning yet (Figure 4.12), we increased the number of games to 3.5 million (already knowing that at that point the computational memory limit would still allow us to run normally from the first experiment) and tested the learning agent's policy in 20,000 games as previously. The results now were very satisfactory (Figure 4.13). According to the graph of the Figure 4.14 we notice that the learning agent suddenly started to insist on using a specific pattern of trading actions (probably one in particular) that increased its winning performance dramatically after 1.5 million games.

The data that we had collected so far were not enough though to give us precise

[1]The percentages accumulate over multiple turns. These small % changes reflect that the adversary is not *dramatically* influenced by manipulation. If the percentages were much larger (e.g. 30%) then the results presented below would be more extreme.

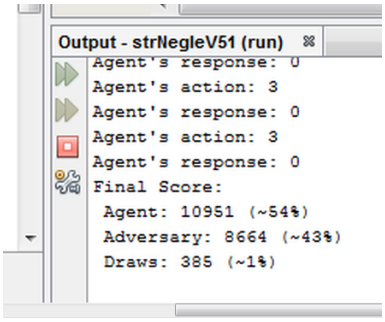


Figure 4.11: *Learning Agent's and Adversary's performance in 350,000 training games. Trained on 350,000 games and then tested on 20,000 more our learning agent's performance is* 11% *better than that of its adversary.*

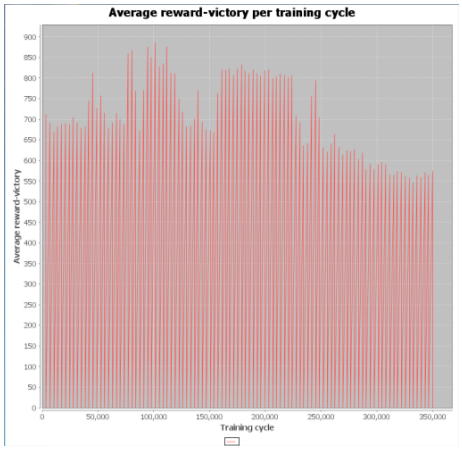


Figure 4.12: *Learning Agent's reward-victory graph in 350k training games. The learning agent's average reward-victory graph (left graph) has not shown any signs of learning yet.*

details about the learning agent's winning strategy. Therefore, as a second step, we worked again (as in the first experiment) with the dataset of the learning agent's actions and responses that were generated from these tests in order to analyse their frequencies. The produced data is the following:

Agent's strategy focuses on action 9 (Table 4.2). That makes the adversary more likely to give wheat and sheep to our learning agent, which hides information from it (i.e. through "I need rocks", as rock is one of the two goal resources).

Learning Agent Actions Priority

1-9-7-10-6-8-3-5-4-2

Learning Agent Responses

1. No
2. Yes

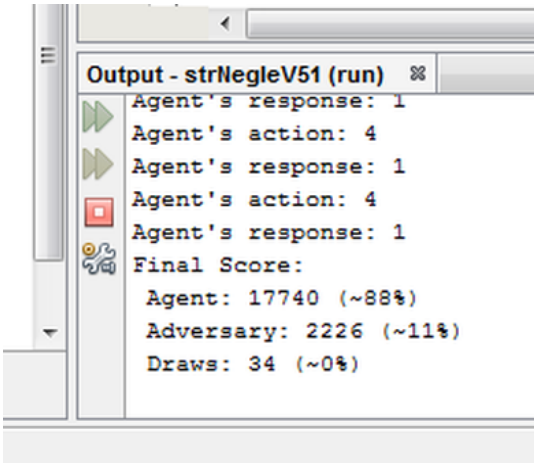


Figure 4.13: *Learning Agent's and Adversary's performance after 3.5 million training games of Experiment 2. Trained on 3.5 million games and then tested on 20,000 more our learning agent's performance is 77% better than that of its adversary.*

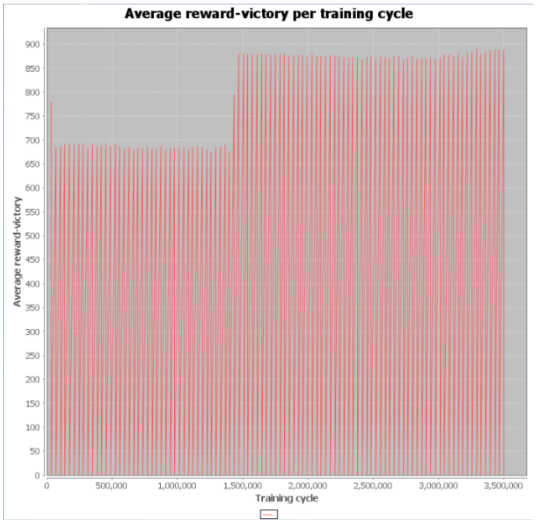


Figure 4.14: *Learning Agent's reward-victory graph in 3.5 million training games of Experiment 2.The learning agent suddenly increased its winning performance dramatically at the point of 1.5 million games and then follows a slightly upward trend.*

Learning Agent Response Frequencies

1. 149291 (32.74%)
2. 306697 (67.26%)

Learning Agent Response Priority

2-1

The acceptance of the adversary's trades dominates in this experiment the rejections. That was good evidence for us to underline the fact that our learning agent

Learning Agent Actions	Action's frequency
1. Do nothing	62003 (13.53%)
2. Give wheat for rock	27505 (6%)
3. Give wheat for sheep	41050 (8.96%)
4. Give rock for wheat	30738 (6.71%)
5. Give rock for sheep	34302 (7.49%)
6. Give sheep for wheat	45876 (10%)
7. Give sheep for rock	59575 (13%)
8. I need wheat	44321 (9.67%)
9. I need rock	**60827** (13.27%)
10. I need sheep	52017 (11.35%)

Table 4.2: *Frequencies of actions of the manipulating learning agent (Exp.1), in 20k testing games, after 3.5m training ones.*

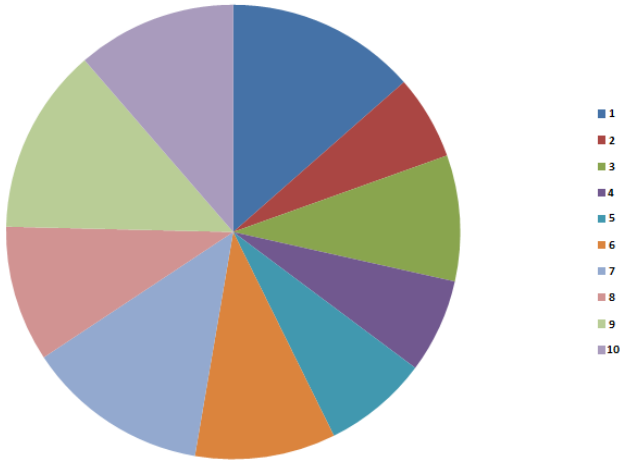


Figure 4.15: *Learning Agent Actions Pie Chart in Experiment 2*

is of course eager to accept these trades because it has triggered them by itself, by initially deceiving its adversary. It was a reasonable piece of information that reinforced the validity of our results.

The outcome so far was positive and offered to us strong evidence that the learning agent has indeed learnt how to manipulate its adversary. Running another test with 3.5 million training games again provided us with similar positive results. However, in this case the agent would use more frequently the manipulative action 10. ("I need sheep") than the 9. Obviously in this case the agent learnt how to lie (as sheep is not a goal resource) to its opponent and showed to us that in the second experiment there can be more than one successful winning strategies (as opposed to the Experiment 1 where only one exists).

As a third step in the second experiment, by showing that the frequency of specific manipulative actions is gradually being increased, it would provide evidence that the agent is indeed learning how to hide information or lie. Therefore in our previous example, where in 3.5 million training games our learning agent learnt how

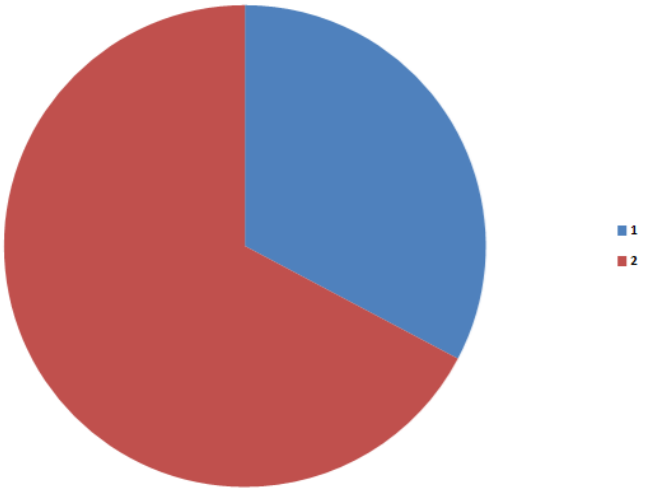


Figure 4.16: *Learning Agent Responses Pie Chart in Experiment 2*

to hide information (by mainly asking for rocks via "I need rocks"), we analysed its manipulative actions in order to verify whether or not the trend of the graph for the action 9 (and perhaps for the other two) would move upwards. We plotted the frequencies of those actions from our action dataset, game after game (Figure 4.17)[2]. Despite the fact that the trends of the actions 8 ("I need wheat") and 10 ("I need sheep") showed no upward trend, that of the action 9 ("I need rock") did (as there are more higher blue dots at the end of the curve than the beginning), supporting our conclusion regarding hiding information.

4.3 Hybrid strategy

In this third experiment (Experiment 3) the adversary's logic is affected by both of the strategies of the two previous experiments, as we discussed in detail in the previous chapter (Section 3.3.7). Hence, the learning agent is called now to learn how to confront a very efficient and stricter strategy compared to those of the previous tests (experiments). We were very curious to see if the algorithm of our learning agent would be able again to learn how to win more games than the hybrid adversary. This time we preferred to run three identical tests of 3.5 million training games each outright (having the knowledge from the previous experiments). The results (in 20 thousand testing games that followed) along with the graphs can be seen in Figures 4.18, 4.19, 4.20, 4.21, 4.22 and 4.23.

The learning agent managed to learn how to win most of the 20,000 testing games (trained on 3.5 million) in all of our 3 different tests. However the learning process proved to be harder in this experiment compared to those of the previous experiments. Still, its algorithm was able to confront the adversarial strategy once

[2]Plotted by Voyant (`http://voyant-tools.org/`)

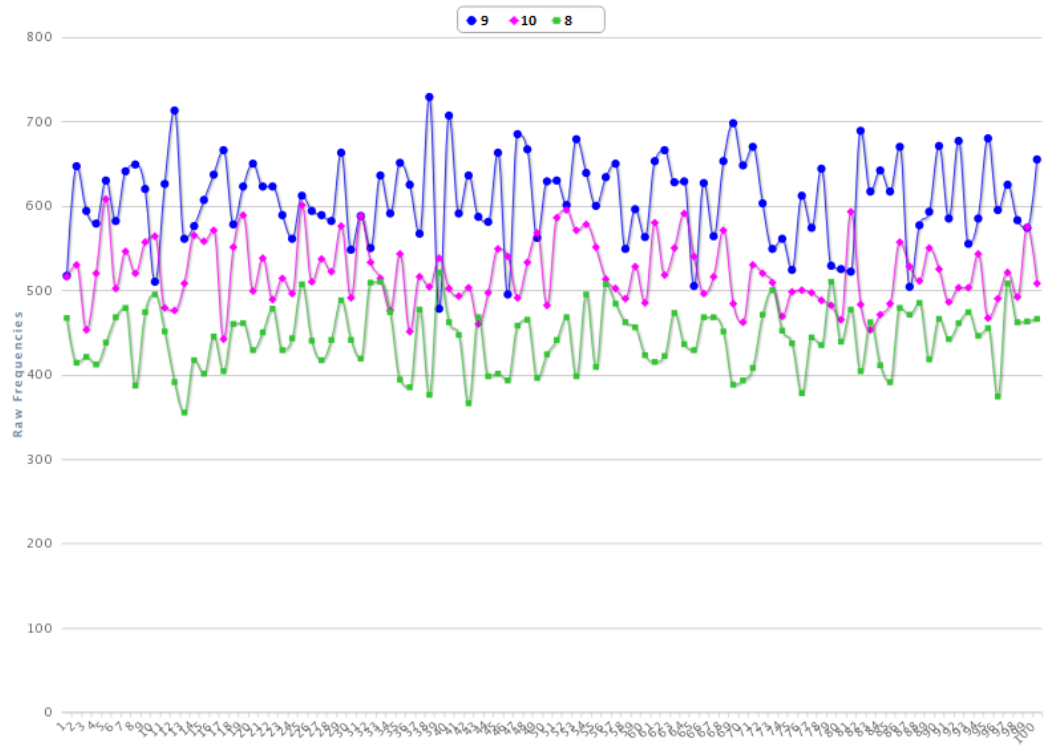


Figure 4.17: *Learning Agent's manipulative actions frequency graphs in 3.5 million training games of Experiment 2. The learning agent uses more and more the action 9. ("I need rock") in order to hide information from its adversary and direct it to a specific state of belief. The trend of its graph presents higher points at the end of the training games than the beginning as the agent uses it more and more due to learning. On the other hand, the actions 8. and 10. seem to be unaffected, as expected.*

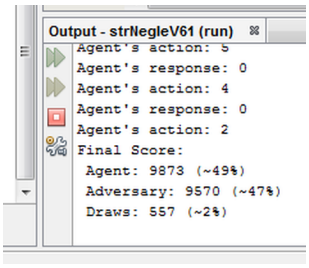


Figure 4.18: *Test 1 Hybrid Results in 20k testing games*

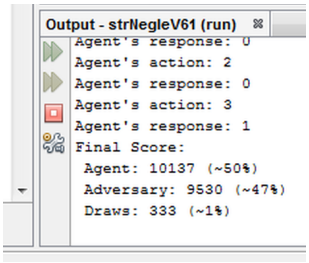


Figure 4.19: *Test 2 Hybrid Results in 20k testing games*

again, with much difficulty though as the differences between the learning agent's

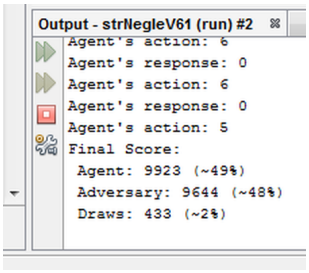


Figure 4.20: *Test 3 Hybrid Results in 20k testing games*

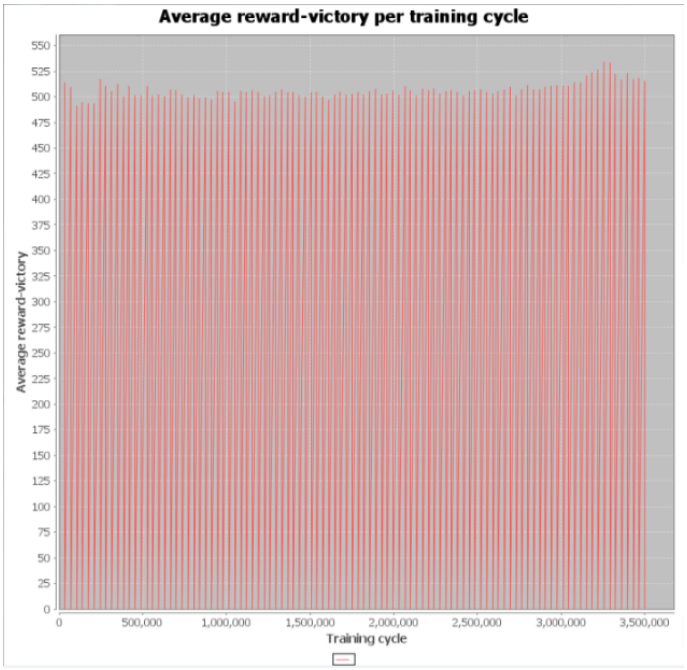


Figure 4.21: *Test 1 Hybrid Graph in 3.5m training games*

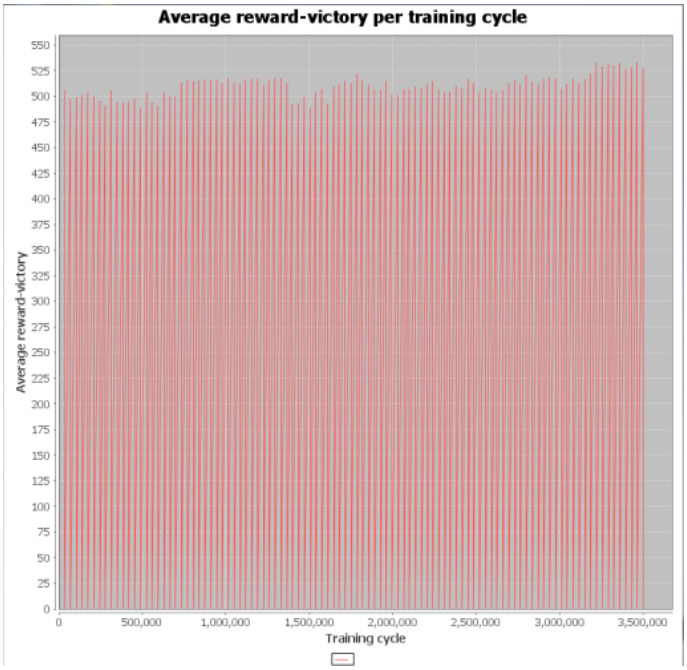


Figure 4.22: *Test 2 Hybrid Graph in 3.5m training games*

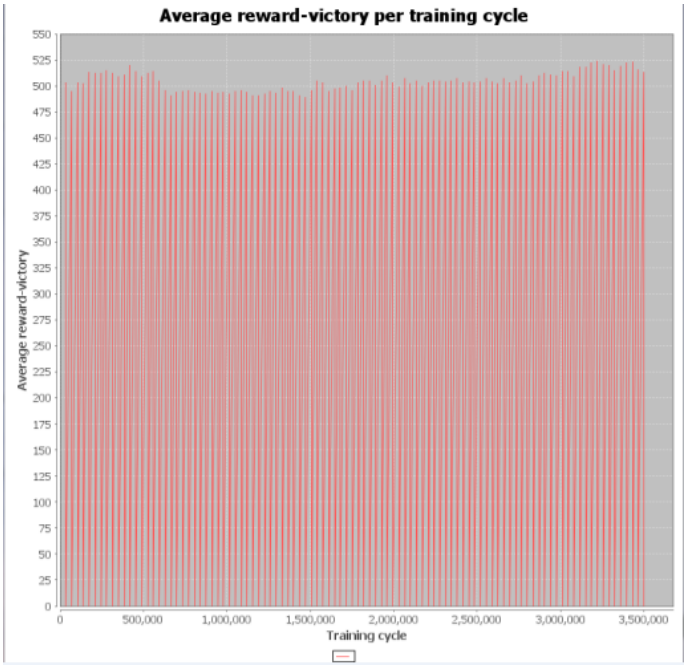


Figure 4.23: *Test 3 Hybrid Graph in 3.5m training games*

performance and that of the adversary are in the range of 1-3%. Overall results of all the conducted experiments so far can be found in Figure 4.24.

4.4 Statistical significance

We performed Z-tests[3] between the winning learning agent sample of our baseline case (wins of the LA in Experiment 1) and that of each one of our other cases (Figure 4.24). By considering only the wins of the LA from the baseline case (Experiment 1) and those of the other two cases we performed Z-tests between their distributions to prove that they are statistical significant. The result of Experiment 2 was significant at $p < 0.05$, but that of the Experiment 3 was not. Hence we can confidently say that the results of the Experiment 2 reject the null hypothesis[4] (Figure 4.24). In more detail, the total number of (testing) games that were taken into consideration in every case were 20,000. The total number of wins from the Experiment 1 were 10,205, and from the Experiment 2 were 17,740. The LA's wins in Experiment 2 have higher score than those of the LA in the Experiment 1 ($z = 82.1065$, $p = 0$). The total number of wins from the Experiment 3 were 10,137. The LA's wins in Experiment 3 do not have higher score than those of the LA in the Experiment 1 ($z = -0.6801$, $p = 0.4965$).

[3]using an online tool (http://www.socscistatistics.com/tests/ztest/Default2.aspx)

[4]The null hypothesis states that the LA's wins in Experiment 2 are not statistically significant compared to those of the LA in Experiment 1.

4.5 Summary

We developed an algorithm inspired by SARSA(0) to train a RLA in our game Taikun. The RLA managed to learned how to win a strict adversarial policy, resulting in a possible Nash-equilibrium, proving that the game is solvable by RL in an MDP framework despite the hidden information of the adversarial resources and goal. Furthermore, by using explicit linguistic manipulation with the form of scalar implicature, the RLA managed to win more often against adversaries which combined the strict strategy with the effect of the manipulation in various ways (e.g. our hybrid approach) that made its task harder. As we will see in the next chapter too, the effect is based on realistic reasons that might apply to human reasoning. In more detail, in this chapter we presented learned policies that successfully deceive or hide information, often falsely asking for resources, dramatically increasing the performance compared to that which does not manipulate. The next chapter will offer more evidence of the effectiveness and importance of linguistic manipulation in RL, presenting deeper connections with Pragmatics and Psychology, through various corresponding experiments.

Best Performances	**Experiment 1** (rule-based strict adversarial strategy)	**Experiment 2** (manipulative actions)	**Experiment 3** (Hybrid adversarial strategy)
3.5 million training games / 20k testing games	***Baseline*** *Agent: 51.025 %* *Adversary: 42.835 %* *Draws:* 6.14 % *Optimal Strategy: Unique (Restriction of the adversary's sheep resource)*	*Agent: 88.7 %* * *Adversary: 11.13 %* *Draws: 0.17 %* *Optimal Strategy: At least two (Hiding information about the rocks resource and lying about the sheep resource)*	*Agent: 50.687 %* *Adversary: 47.65 %* *Draws: 1.665 %* *Optimal Strategy: Currently unknown*

Figure 4.24: *Best performances observed in 3.5 million training games and 20k testing games. In Experiment 1 the agent's unique dominating strategy of restricting the sheep resource from its adversary provides it with a difference in its winning performance of almost* 10%. *In Experiment 2 the adversary, being deceived by the learning agent's various manners (2 different strategies have been observed here), loses by* 77% *more often than the learning agent. The adversary's strategy in the Experiment 3, still loses from that of the learning agent by* 3%. *The LA's wins in Experiment 2 have significant improvement (*) over baseline (wins of the LA in Experiment 1),* $p < 0.05$. *The LA's wins in Experiment 3 do not have significant improvement over baseline (Experiment 1).*

Chapter 5

Experiments in Taikun: Manipulation detection

A new learning algorithm was implemented for all the experiments of this chapter, based on SARSA(λ) that we have examined in Section 2.1.6. Hence, most[1] of the experiments of the previous chapter were re-conducted using the new SARSA(λ) learning agent, along with others. In particular, the first experiment (Section 5.1) uses again the rule-based adversary (which is more difficult to beat in this case than that of Chapter 4) and the second experiment (Section 5.2) uses the gullible adversary which is able to be manipulated (and then it hinders the LA by restricting resources). This adversary always starts playing the game by following the "strict" strategy of the previous rule-based one. The third experiment (Section 5.3) consists of 3 different cases: the LA plays against an adversary which always restricts the wheat resource, or against an adversary which always restricts the rock resource, or against an adversary which always restricts the sheep resource. The fourth experiment (Section 5.4) studies the cases where the adversary of the second experiment has the additional capability of exposing the LA every time it uses one of the manipulative actions. This property is based on a probability which varies as we will see below. Furthermore, the penalty for being "caught" while deceiving also varies. The reader will also find information about our corresponding statistical significance tests (Section 5.5), the parameters that we used in our learning algorithm (Section 5.6), as well as experiments with one manipulation only (Section 5.7) to check *when* is the best time to manipulate. Finally, experiments with adversaries which detect manipulation based on logical contradictions (Section 5.8) will be presented. For reasons of available space, not the whole sequence of experiments will be included below (as they were literally hundreds) but only those which provide a clear picture of what has been done and the corresponding motivation.

[1] We did not use the third experiment with the hybrid adversarial strategy (Section 4.3) because our findings there were sufficient.

5.1 Strict adversary

This strategy was designed to form a challenging rational adversary for measuring baseline performance. It is similar to that of Section 4.1, but now it is more "strict" as we will see in an attempt to make the RL task harder than before. It cannot be manipulated at all, and non-cooperative dialogue moves will have no effect on it – it simply ignores statements like "I really need wheat". The strict rule-based strategy of the adversary will never ask for a resource that it does not need (in this case rocks). Furthermore, if it has an available non-goal resource to give then it will offer it. It only asks for resources that it needs (goal resources: wheat and sheep). In the case where it does not have a non-goal resource (rocks) to offer then it offers a goal resource only if its quantity is more than it needs, and it asks for another goal resource if it is needed[2]. Following the same reasoning, when replying to the LA's trading proposals, the adversary will never agree to receive a non-goal resource (rock). It only gives a non-goal resource (rock) for another one that it needs (wheat or sheep). It also agrees to make a trade in the special case where it will give a goal resource of which it has more than it needs for another one that it still needs (this rule is also used only by the adversary of this chapter, and not from the strict one of Chapter 4). This is a strong strategy that wins a large number of games. In fact, it takes about 100,000 training games before the LA is able to start winning more games than this adversary, and a random policy loses 66% of games against this adversary (See Table 5.3, LA policy "Random").

The LA scored a winning performance of 49.5% against 45.6% for the adversary, with 4.9% draws (Table 5.3), in the 20 thousand test games that followed after 1.5 million training games (Figure 5.1). This represents the baseline performance that the LA is able to achieve against an adversary which cannot be manipulated at all. This shows that the game is "solvable" as an MDP problem, and that a reinforcement learning agent can outperform a strict hand-coded adversary. Here, the learning agent's strategy mainly focuses on offering the sheep resource that it does not need for the rocks that does need (for example $action7 > action2 > action6 > action3$, Table 5.1). It is also interesting to notice that the LA learnt not to use action 3 at all (gives 1 wheat that they both need for 1 sheep that only the adversary needs). Hence its frequency is 0. The actions 4 and 5 are never accepted by the adversary so their role in both of the experiments is similar to that of the action 1 (do nothing). The rejections of the adversary's trades dominate the acceptances with a ratio of 94 to 1 as our learning agent learns to become negative towards the adversarial trading proposals and therefore to prohibit its strategy.

[2]This rule is used only by the adversary of this chapter, and not from the one in Chapter 4. Thus the strict adversary of this chapter is more difficult to beat than that of Chapter 4.

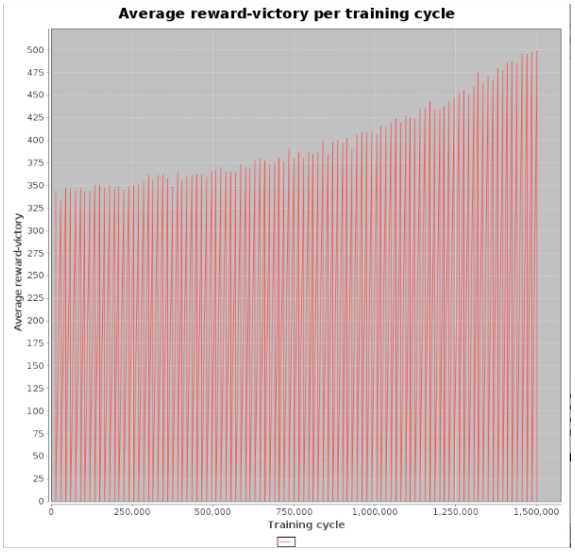


Figure 5.1: *Learning Agent's reward-victory graph over 1.5 million training games of Experiment 1.*

5.2 Manipulation

Here the adversary retains the above strict base-line policy but it is also susceptible to the non-cooperative moves of the LA, as explained above. For example, if the LA utters "I really need rock", weights of actions which transfer rock from the adversary will decrease, and the adversary will then be less likely to give rock to the LA in an attempt to hinder it. Conversely, the adversary is then more likely to give the other two resources to the LA. In this way the LA has the potential to mislead the adversary into trading resources that it really needs. The weights of actions change in exactly the same way as that which was discussed in Section 4.2. The only difference here is that the weights reset (i.e. all become equal) at the beginning of each new game, therefore making the learning task harder.

In this experiment the learning agent scored a winning performance of 59.2% against only 39.75% of its adversary, having 1.1% draws (Table 5.3), in the 20 thousand test games that followed after 1.5 million training games (Figure 5.2). Similarly to the previous experiment, the LA's strategy focuses again mainly on action 7, by offering the sheep resource that it does not need for rocks that it needs (Table 5.1). However in this case we also notice that the LA has learnt to use action 2 very often, exploiting cases where it will win by giving the wheat resource that they both need for a rock that only it needs. This is a result of its current manipulation capabilities. The high frequency manipulative actions 8 ("I really need wheat") and especially 9 ("I really need rock") assist in deceiving its adversary by hiding information (as they are used on their own), therefore significantly reinforcing its strategy as they both indirectly result in gaining sheep that only the adversary needs.

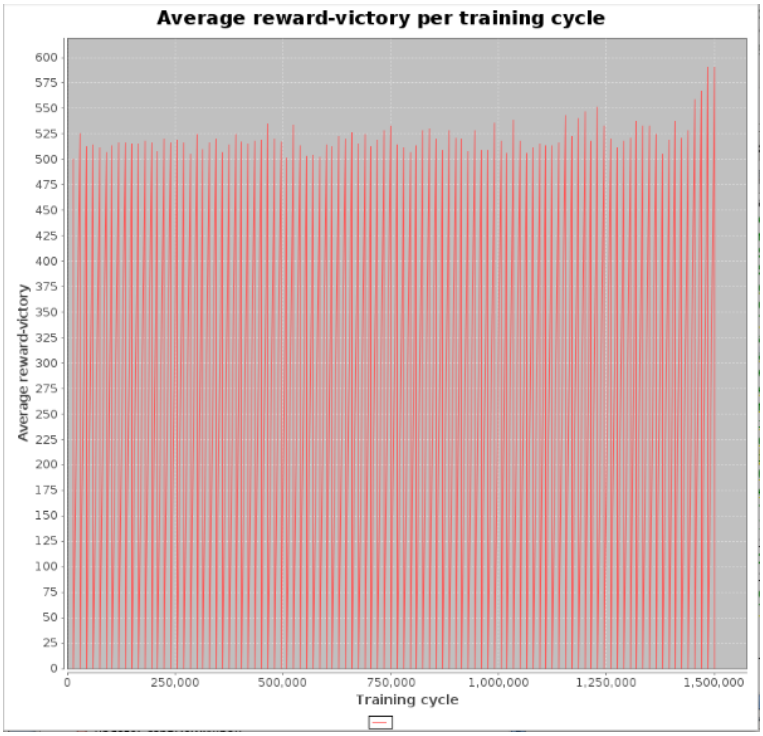


Figure 5.2: *Learning Agent's reward-victory graph in 1.5 million training games of Experiment 2.*

Rejections to adversarial trading offers over the acceptances were again the majority in this experiment. However in this case they are much fewer than before, with a ratio of only 2.5 to 1, as our learning agent is now more eager to accept some trades because it has triggered them itself by appropriately manipulating its adversary.

5.3 Restriction

Here we investigate performance against adversaries which cannot be manipulated, but their strategy is to always restrict the LA from gaining a specific type of resource by never offering it. We need to explore how well a manipulated adversary (for example one which will no longer give rocks that only its opponent needs) performs. This shows us the potential advantage to be gained by manipulation and most important, it will generalise our problem by showing that the restriction (boycott) of a resource that only the opponent needs, or of a resource that both of the players need, are actually reasonably good strategies compared to the baseline case (first experiment). Hence, the manipulated adversary has indeed a reason for choosing to restrict resources (second experiment) rather than staying with its rule-based

Action number	**Exp.1** frequency	**Exp.2** frequency
1. Do nothing	81969 (29.33%)	144727 (27.23%)
2. Give wheat for rock	8077 (2.89%)	46028 (8.66%)
3. Give wheat for sheep	**0 (0%)**	10358 (1.95%)
4. Give rock for wheat	80578 (28.83%)	62874 (11.83%)
5. Give rock for sheep	78542 (28.1%)	55627 (10.46%)
6. Give sheep for wheat	6429 (2.3%)	24687 (4.64%)
7. Give sheep for rock	**23888 (8.58%)**	31132 (5.86%)
8. I really need wheat	-	**68974 (12.98)**
9. I really need rock	-	**87123 (16.39%)**
10. I really need sheep	-	18 (0.003%)

Table 5.1: *Frequencies of actions of the non-manipulative learning agent (Exp.1) and of the manipulative one (Exp.2), in 20k testing games, after training. Bold numbers indicate the trading proposals that were mostly used by the LA and were accepted by the adversary.*

strategy. In other words it has a rational reason to hinder the LA (being gullible) and fall in its trap.

In this (third) experiment the LA uses no manipulative actions. It is the same LA as that of the first experiment. It is trained and then tested against 3 different types of restrictive adversaries. The first one (Experiment 3.1) never gives wheat, the second one (Experiment 3.2) never gives rocks, and the third one never gives sheep (Experiment 3.3). They all act randomly regarding the other 2 resources which are not restricted. In the first case (adversary restricts wheat that they both need), the LA scored a winning performance of 50.015% against 47.9% of its adversary, having 2.085% draws (see Table 5.3) in the 20 thousand test games. In the second case (adversary restricts rocks that the LA only needs), the LA scored a winning performance of 53.375% against 44.525% of its adversary, having 2.1% draws in the 20 thousand test games. In the third case (adversary restricts sheep that only itself needs), the LA scored a winning performance of 62.21% against 35.13% of its adversary, having 2.66% draws in the 20 thousand test games. These results show that restricting the resource that only the opponent needs (i.e. LA only needs rocks) or the resource that they both need (i.e. wheat) can be as effective as the strategy followed by the rule-based adversary (see Table 5.3). The difference in the performances for the former case (rock) is +8.85% and for the latter (wheat) only +2.115%[3]. That means the adversary has indeed a reason to believe that boycotting its opponent's resources could be a winning opposing strategy, motivating its gullibility and hindering behaviour in experiment 2 of Chapters 4 and 5 (Sections 4.2 and 5.2 respectively).

[3]Further experiments showed that having the same number of goal resources (i.e. both need 4 of their own goal resources, rather than 5) still produces similar results.

5.4 Exposure

In this part we extend the problem to include possible negative consequences of manipulative LA actions [27]. The adversary begins each game with a probability of detecting manipulation, which increases after every one of the LA's manipulative actions. In more detail, every time the LA performs a manipulation, there is an additional chance that the adversary notices this (starts at 1-in-10 or 1-in-20 and increases after every manipulative move, up to 100% in the case of the 10th or 20th respectively manipulative attempt[4]). The *consequences* of being detected are either that the adversary will refuse to trade with the LA any further in that game (Experiment 4.1, which consists of 4.1.1 and 4.1.2, of Section 5.4.1), or that the adversary automatically wins the game (Experiment 4.2 of Section 5.4.2). In the former case the LA confronts a medium penalty of exposure as the game still goes on, while in the latter case the penalty is the highest possible (immediate loss). In these two cases there is always a high risk associated with attempting to manipulate, and the LA has to learn how to balance the potential rewards with this risk.

5.4.1 Refusal of trading

In this case when the LA is exposed by the adversary then the latter does not trade for the rest of the game. We have explored two different cases here, one with a 10% chance of exposure (Experiment 4.1.1) which gradually increases to 100% at the 10th attempt and another one (Experiment 4.1.2) with a chance of 5% which gradually increases to 100% at the 20th attempt. The LA scored a winning performance of 50.86% against 46.33% for this adversary, having 2.81% draws in the 20 thousand test games of Experiment 4.1.1. In the second case (Experiment 4.1.2), the LA scored a winning performance of 51.785% against 45.595% for this adversary, having 2.62% draws in the 20 thousand test games. In both cases the 20k test games followed 150k training ones (Figure 5.3). Lower risk provided a slightly better performance to our LA in the second case, which was more successful than that of the first case by +1.66%.

The results showed (Table 5.2) that the LA managed to locate a successful strategy that balances the use of the manipulative actions and the normal trading actions with the risk of exposure. In more detail, the strategy that the LA uses here in both of the cases makes frequent use of the manipulative actions 9 ("I really need rock") and 10 ("I really need sheep") which mainly result in the collection

[4]Nothing particularly hinges upon these numbers. Detection would be much more "rapid" if the percentages were higher.

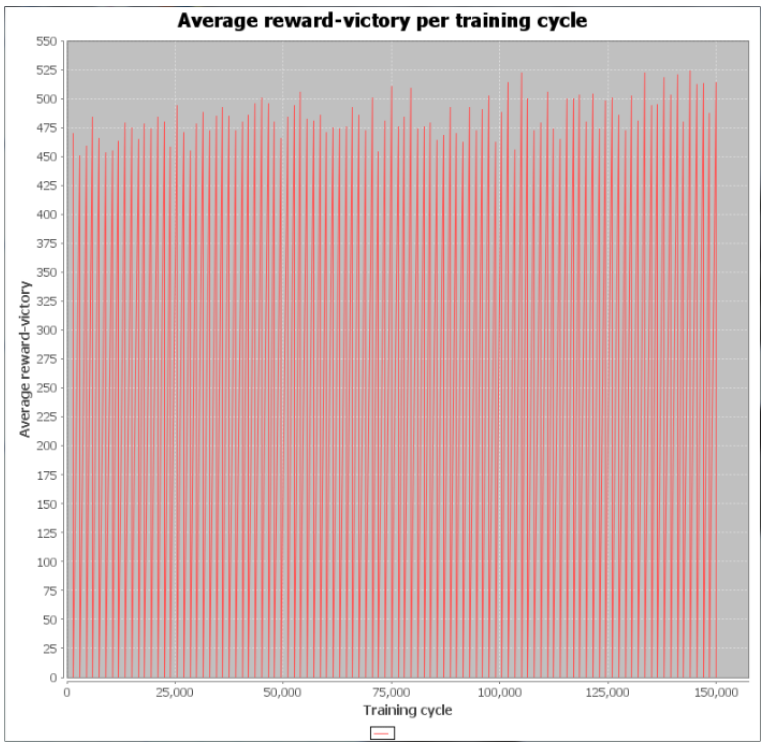


Figure 5.3: *Learning Agent's reward-victory graph in 150 thousand training games of Experiment 4.1.1. The graph of the experiment 4.1.2 looks similar due to the strong connection between the two experiments and therefore has been omitted.*

of wheat that both need to win. These two manipulative actions hide information and lie respectively. Restriction of a resource that both of the players need is a very successful strategy (as our third experiment, Section 5.3 suggests) and the LA managed to locate that and exploit it. The next highest frequency action (excluding actions 4 and 5 that lead to rejection from the adversary as it also follows its rule-based strategy) is 7 ("I will give you a sheep if you give me a rock") that is exclusively based on the LA's goal and along with 6 they "selectively" give back the sheep for goal resources. Rejections to adversary's proposals over the acceptances were in a ratio of approximately 17 to 1 in the first case and 13 to 1 in the second one. The LA is quite eager (in contrast to the baseline case of experiment 1) to accept the adversary's proposals as it has already triggered them by itself through deception.

5.4.2 Instant win

In this case if the LA becomes exposed by the adversary then the latter instantly wins the game. Here we also have a 10% chance of exposure which gradually increases to 100% at the 10th attempt. The LA scored a winning performance of 49.7% against 46.225% for the adversary, having 4.075% draws in 20 thousand test games that followed after 1.5 million training ones (Figure 5.4). As was expected

from our previous work [26], the LA performed similarly to the baseline case (first experiment, Table 5.3) and learned never to use manipulative actions since they are so dangerous. The LA has so far managed to locate a strategy that is similar to that of our baseline case. The frequencies of the actions are shown in the Table 5.2. According to Figure 5.4 we notice that the LA gradually learns how to reach the baseline policy (i.e never use manipulation). Rejections to the adversary's proposals over the acceptances were in a ratio of approximately 59 to 1, meaning that the LA is again quite eager to reject the adversary's trading proposals as its cautious behaviour is similar to that of the baseline case (first experiment, Section 5.1).

Action number	**Exp.4.1.1** frequency	**Exp.4.2** frequency
1 Do nothing	83563 (28.5%)	109504 (39.32%)
2 Give wheat for rock	10195 (3.47%)	7780 (2.8%)
3 Give wheat for sheep	1985 (0.68%)	16 (0.005%)
4 Give rock for wheat	53684 (18.3%)	78851 (28.31%)
5 Give rock for sheep	29162 (9.94%)	57126 (20.51%)
6 Give sheep for wheat	8700 (2.96%)	1896 (0.68%)
7 Give sheep for rock	27716 (9.45%)	23323 (8.37%)
8 I really need wheat	19826 (6.76%)	**0 (0%)**
9 I really need rock	**31755 (10.86%)**	**0 (0%)**
10 I really need sheep	**26807 (9.14%)**	**0 (0%)**

Table 5.2: *Frequencies of actions of the manipulative learning agent versus the adversary which refuses to trade (Exp.4.1.1, exp.4.1.2 is similar) and versus the adversary which instantly wins the game (Exp.4.2), when exposure occurs. LA actions in 20k testing games, after training.*

5.5 Statistical significance

Similarly to Section 4.4, we performed Z-tests[5] between the winning learning agent sample of our baseline case (Exp. 1) and that of each one of our other cases (Table 5.3). By considering only the wins of the LA from the baseline case (Exp. 1) and those of the other cases we performed Z-tests between their distributions to prove that they are statistically significant. Every result was significant at $p < 0.05$, apart from the cases of restriction of the wheat (Exp. 3.1) and exposure with penalty of instant win (Exp. 4.2), $p = 0.3$ and $p = 0.68$ respectively, with results similar to that of the baseline. Hence we can confidently say that those cases (Exp. 2,

[5]http://www.socscistatistics.com/tests/ztest/Default2.aspx

3.2, 3.3, 4.1.1, 4.1.2) reject the null hypothesis[6] (Table 5.3). In more detail, the total number of (testing) games that were taken into consideration in every case were 20,000. The total number of wins from the Experiment 1 were 9,900, and from the Experiment 2 were 11,834. The LA's wins in Experiment 2 have higher score than those of the LA in the Experiment 1 ($z = 19.4131$, $p = 0$). The total number of wins from the Experiment 3.2 were 10,675. The LA's wins in Experiment 3.2 have higher score than those of the LA in the Experiment 1 ($z = 7.7532$, $p = 0$). The total number of wins from the Experiment 3.3 were 10,675. The LA's wins in Experiment 3.3 have higher score than those of the LA in the Experiment 1 ($z = 25.5961$, $p = 0$). The total number of wins from the Experiment 4.1.1 were 10,172. The LA's wins in Experiment 4.1.1 have higher score than those of the LA in the Experiment 1 ($z = 2.72$, $p = 0.00652$). The total number of wins from the Experiment 4.1.2 were 10,357. The LA's wins in Experiment 4.1.2 have higher score than those of the LA in the Experiment 1 ($z = 4.5704$, $p = 0$). The total number of wins from the Experiment 3.1 were 10,003. The LA's wins in Experiment 3.1 have higher score than those of the LA in the Experiment 1, but they are not significant ($z = 1.03$, $p = 0.30302$). The total number of wins from the Experiment 4.2 were 9,940. The LA's wins in Experiment 4.2 have higher score than those of the LA in the Experiment 1, but they are not significant ($z = 0.4$, $p = 0.68916$).

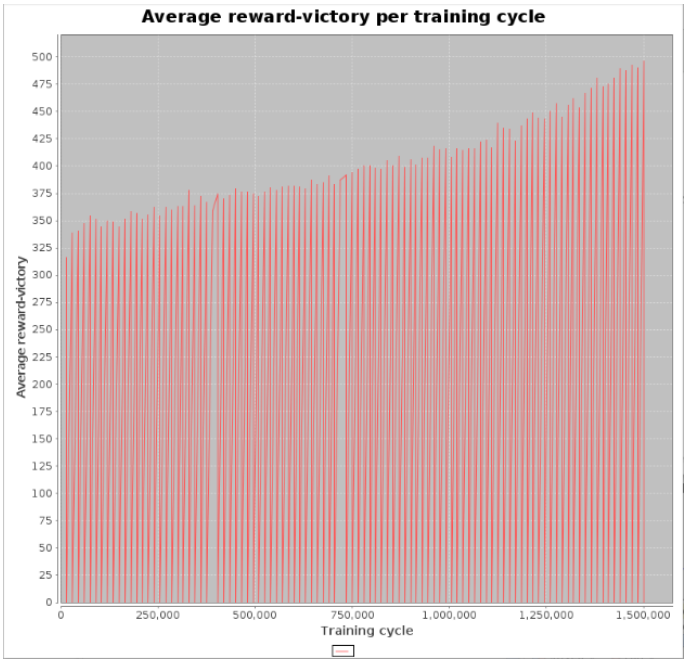


Figure 5.4: *Learning Agent's reward-victory graph in 1.5 million training games of Experiment 4.2.*

[6]The null hypothesis states that the LA's wins from these experiments are not statistically significant compared to those of the LA in Exp. 1.

Exp.	**Learning Agent policy**	**Adversary policy**	**LA win**	**Adversary**
	Random	Baseline (strict)	32%	66%
1	**SARSA**	**Baseline (strict)**	**49.5%**	45.555%
2	SARSA + Manipulation	Baseline (strict)+Gullible	59.17%*	39.755%
3.1	SARSA	Restrict wheat	50.015%	47.9%
3.2	SARSA	Restrict rock	53.375%*	44.525%
3.3	SARSA	Restrict sheep	62.21%*	35.13%
4.1.1	SARSA+Manipulation	Basel.+ Gull.+Expos(10%).(no trade)	50.86%*	46.33%
4.1.2	SARSA+Manipulation	Basel.+ Gull.+Expos(5%).(no trade)	51.785%*	45.595%
4.2	SARSA+Manipulation	Basel.+ Gull.+Expos(10%).(win game)	49.7%	46.225%

Table 5.3: *Performance (% wins) of the discussed learning agents and adversaries, in 20K testing games, after training. (*= significant improvement over baseline [Exp. 1]* ***in bold text****,* $p < 0.05$*)*

5.6 Parameters

We conducted a number of experiments to verify that λ=0.4 is indeed a wise choice as we discussed in Section 3.2.4. Experimenting exclusively with the baseline case and testing different values for λ, we produced the following table which shows the difference in the performance between the learning agent and that of the baseline adversary. Negative numbers indicate that the LA had more losses than wins (Figure 5.5).

The numbers in the parentheses indicate the number of games that we averaged in order to produce the final result, which is in the same cell. On some cases the results were the lowest that have been ever observed so one test was enough to convince us (or sometimes two). We see that λ produces good results in the range between 0.4 and 0.7. As most of our testing cases had already used the value 0.4 though (according to our discussion in Section 3.2.4) and the results were still impressive, we decided to use that value in the future since the next three values didn't have much difference. It was a long process as each test consisted of 3.5 million training games and then 20 thousand more testing games. Each required almost a day to finalise.

5.7 One chance at manipulation

Let's imagine the following example: a human trader negotiates for the first time ever with another trader. An arbitrary use of manipulative actions would not be a wise choice, as it would be risky (based on Section 5.4 too), but instead a cautious (or conservative) use of these actions sounds more appropriate. In this part we

allow only one manipulative action to occur in each game during the exploration phases of the LA's learning process. The main reason that we did that (apart from investigating *when* is the best time to manipulate, Section 5.7.2) is due to human intuition: a human trader would never risk using a manipulative move too much (i.e. more than once) when she is unsure of its chance of exposure due to lack of previous experience (LA's exploration phases). This assumption is related to the dual-mind cognition theory of Section 5.7.1 that we will examine below. She would never risk that especially if she knew that exposure would cost the end of all future negotiations with the particular adversary. That is a case that might happen due to the lack of trust [85] that such an exposed behaviour might have caused. In the experiments of the Section 5.4.1 we considered though that when the adversary exposes the LA then it stops trading in the current training game only, as there was no emphasis on the trust factor there. It resumes trading though in the next one and so on. By making the assumption that Reinforcement Learning resembles the human learning, then we can claim that the RL's exploration phase (when a new action is used for the first time in a particular state) and exploitation phase (when the most successful action in the particular state is used again) are followed by a human during a real life trading scenario.

Hence the idea for our learning agent to use one manipulation (i.e. "I really need wheat") only during the exploration learning phases of a Taikun game[7] was born. There were several purposes that led us towards that direction. One of them was to avoid getting exposed in a real life trading scenario, such as the above, with the human traders and the "trust" factor. There a second manipulative action which is attempted for the first time in a particular state (as in exploration phase) might mean the end of any future trading negotiations with the particular adversary. Exposure might easily occur due to logical contradictions between dialogue moves as

[7]more than one manipulating actions may be used when the LA is exploiting though (e.g. during the testing games after the training ones), as it has "safely" learned how to use them.

BASELINE 3.5m Training Games

Our choice for the value of the λ parameter

λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	4%	3.4%	4.5%	4.7% (4)	4.7% (3)	5% (2)	4.7% (2)	3.9% (2)	2.4% (2)	-1.4% (2)

Figure 5.5: *Table of various tested λ values. The percentages indicate the averaged difference in the performance between the learning agent and that of the baseline adversary. Numbers in parentheses represent the number of experiments, whereas no parentheses and number means that only one experiment has been conducted for that case (i.e. only one difference has been considered). Negative numbers indicate that the LA had more losses than wins.*

we have shown on relevant work [105] and we will discuss in Section 5.8. One contradiction between two manipulative actions may be enough for that. In Sections 5.4.1 and 5.4.2 we have also seen that the manipulative agent was trained against adversaries which were able to expose it based on an accumulative probability which was formed according to the frequency of the agent's manipulative actions. The agent there was using the manipulative actions arbitrarily during the exploration phase, but what if exposure would mean the end of any future trading negotiations with the particular adversary? One manipulative action during the uncertain moments (agent's exploration phases) sounds the safest option that might still learn the agent how to safely (and successfully) trade. Hence it was interesting to see what can RL do in the case where manipulation cannot occur more than once during the exploration phases of training in each game.

Another important reason that we allowed the LA to use only one manipulative action during its exploration phases was to see whether RL would be capable of learning a policy that would be more conservative in using manipulation (keeping a possible risk lower) and on the same time perform successfully. By saying "perform successfully" we mean compared to the case of Section 5.2, where manipulative actions do not have a restriction in number while the agent is exploring during its learning process. Another purpose was to increase the trading proposals and therefore trades that occur in the game, as instead of using a manipulative action more than once, the agent now would be forced to select only an action from the set of the normal trading proposals.

5.7.1 Dual-mind cognition

Dual mind cognition according to Evans [31, 32] suggests that humans have two minds which they use in every day life (i.e. when they take decisions): the old mind and the new mind. Simply stated, the old mind is based on experience (what we also call wisdom, driven by behaviours which succeeded in the past) and the new mind is based on decision making regarding *novel problems* and reasoning about the future. By saying "novel problems" we assume that we don't have a solution ready for them and therefore we need to experiment (by using what we call intelligence). Experimentation though means risk and -according to the human intuition and our above "penalty due to exposure" assumption- only the necessary amount of risk (i.e. one manipulation only) that will allow us to gain sufficient experience for the old mind to hold. In RL we assume that these two minds are the exploitation and the exploration phases of the agent respectively. Thus the agent should manipulate only once during the exploration phases (new mind) as it is unaware about the future consequences (penalties) of this action.

5.7.2 When to manipulate?

By using the above reasoning the agent managed to learn *how* and *when* to effectively use manipulation and keep performing as successfully as it did in Section 5.2, where manipulation would happen arbitrarily during the exploration phases without paying attention to possible penalties that such a behaviour could cause. It is now based on the dual mind reasoning that we examined above. By doing that it also learns how to *safely* manipulate during the exploitation phases (old mind), when manipulation occurs more than once.

5.7.3 Results

We conducted an experiment where a LA, which uses one manipulative action only in each game during its exploration phases, played 350 thousand training games against the adversary of the Section 5.2 and then its policy was tested in 20 thousand games that followed. It managed to win 9.87% more games than those of the adversary. We compared this result with that we obtained from a LA which uses manipulative actions arbitrarily during its exploration phases (as in Section 5.2) and played 350 thousand training games against the same adversary. Its performance was similar, 10.295%. In Figure 5.6 we see that manipulation happens less frequently in the case of the LA which uses one manipulation only in its exploration phases during its training. The LA learned to manipulate at the beginning of the game according to the frequencies of the manipulative actions that we have collected (Figure 5.6). According to the game's mechanisms, this is a wise choice as it will lead to the win faster and safer. That is because the adversary will be manipulated at the beginning and it will lose track of its goal early.

5.7.4 Conclusion

Our above results align with the psychological theory of the Section 5.7.1. In other words, we offer to the RL agent a human reasoning (or even better, constraint) on how to learn to manipulate by one time only during exploration. It learns to do that wisely (it uses it whenever it succeeded in the past, that is at the beginning of the games) in the exploitation phases by resulting in a similar performance with that of an agent which manipulates arbitrarily without considering possible consequences. That is without using its new mind in a "human way", as a human

considers the risk of an action in a new state. Thus in a real life scenario it would risk getting exposed way too much. Hence the question is: why should the agent keep manipulating arbitrarily during the exploration phases (new mind) when we obtain similar performance from an agent which manipulates safely (only once) during the exploration phases and evidently learns how to do that mostly at the beginning of a trading scenario? The answer is: it shouldn't keep manipulating arbitrarily, if there is an ethical bond (e.g. trust) between the traders which plays an important role in the particular negotiation, and it has to be preserved.

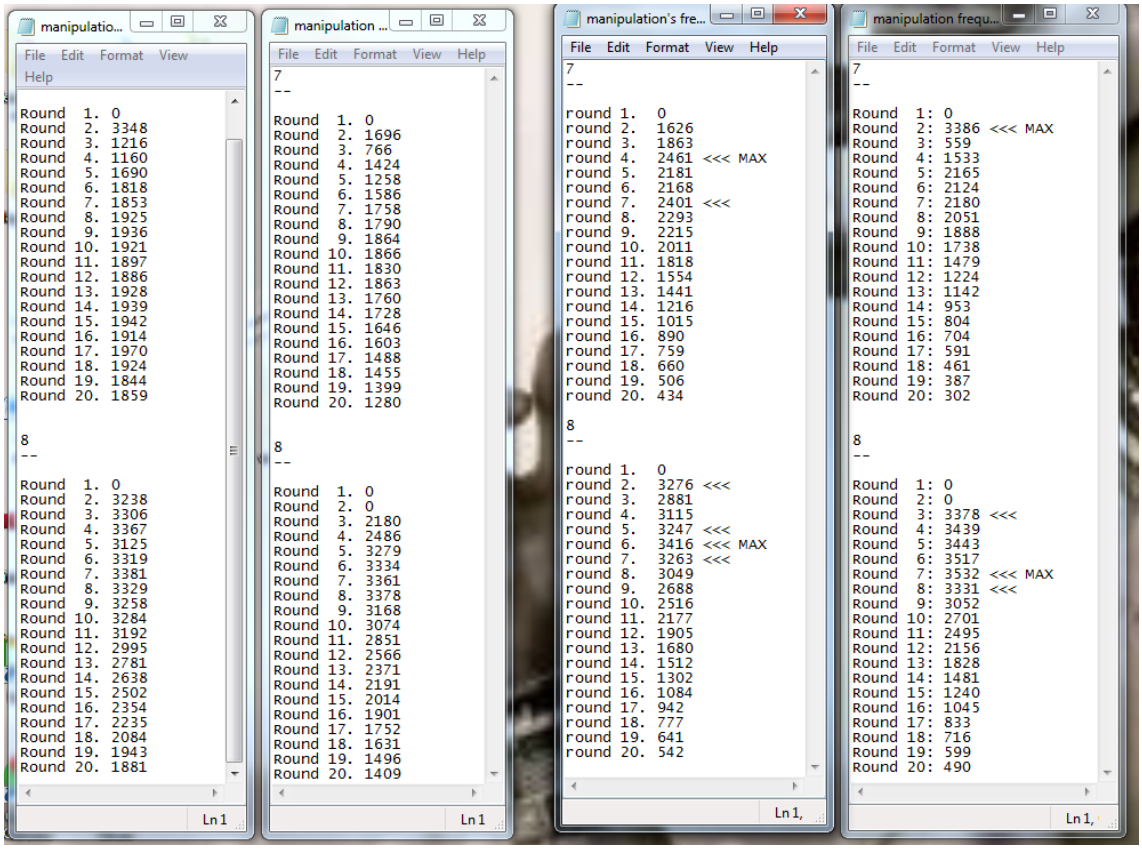


Figure 5.6: *2 lists on the left: frequencies of manipulative actions 7 and 8 in two sessions of 20k testing games each (which occurred after 350k training games) of normal manipulation (Section 5.2). 2 lists on the right: same as left but one manipulation only was allowed in the exploration phases of each training game. The LA learns to manipulate mainly at the beginning of the games, as we can see from the distributions of the frequencies in the 2 lists on the right. These frequencies are high in the first rounds (<<< indicates that) and then they gradually decrease as we can see on the figure.*

5.8 Deception detection

As we mentioned in the Section 5.7, exposure of the manipulative actions (MA) might be also due to identification of logical contradictions. Here we show that by introducing an adversary which detects deception based on those contradictions [105], the performance of our learned policy from the Section 5.2 drops when the penalty is automatic (instant) win for the adversary. In the experiments of the Section 5.2 the LA learned how to win more "Taikun" games against an adversary whose strategy was initially goal-oriented until it would hear the LA's manipulative action (i.e. "I really need sheep"). Then it was acting in a gullible way. That case is now considered our Baseline case for the experiments that will follow. Despite the fact that we did not have the opportunity to re-train that LA against the three adversaries which detect deception based on the three cases that follow, the results are still interesting as we will see below. The experiments that we conducted in the Section 5.4.1 and 5.4.2, where the probability (likelihood) of detection (exposure) by the adversary is increased after each MA, is called in this section Frequency-based approach.

5.8.1 Detection cases

The detection of deception (manipulation) in the experiments that follow are based on a model of semantic inconsistencies amongst the LA's dialogue actions. The following example shows the way that deception can be detected by the adversaries:

1. - LA: I really need wheat. (MA)
2. - ADV: I will give you rock and I need wheat.
3. - LA: OK! (Contradiction)
4. - (Game update)
5. - LA: I will give you wheat and I need sheep. (Contradiction)

According to the above dialogue, the following cases are considered in our experiments by the exposing adversaries:

Case 1: Lies in the same trading phase (Plain lies). In the above dialogue and in particular in 1. the LA falsely announces that it needs wheat (it is a MA) and in its following dialogue turn (3.) it clearly contradicts itself by accepting to

give wheat to the adversary. The adversary in this case detects those lies of the same trading phase.

Case 1+2: Lies in consecutive trading phase (Naive turn-based approach). Here in addition to Case 1, the adversary also takes into consideration logical inconsistencies that occur between a MA and a subsequent LA dialogue action in the next trading phase, as in 5. of the above dialogue.

Case 1+3: Likelihood of consecutive lies (Probabilistic turn-based approach). In this case in addition to Case 1, the adversary takes into consideration the game update (4.) where the players either gain or lose resources at random (as we have examined in the Section 3.1.1) and therefore the probability that the MA is still valid decreases by 1/3 (nothing particularly hinges upon this number). Once a MA is discovered, there are two different ways that the exposure may be penalised by the adversary (as in Section 5.4.1 and 5.4.2): with either refusal to trade for the rest of the game or instant win.

5.8.2 The adversaries and the LA

The three adversaries are all based on the model of the adversary of the Section 5.2, where it starts the game being goal-oriented and when a MA occurs then it becomes gullible. Furthermore, they are all detecting the LA's manipulation based on each of the three cases that we mentioned above. The penalty that is applied after they successfully detect the LA's manipulation, is either refusal to trade for the rest of the game or automatic win. We have conducted experiments for each of these cases. The LA uses the successful trained policy that we have analysed in the Section 5.2 and it has not been re-trained against them.

5.8.3 Results

According to Figure 5.7, we initially notice that all of the LA wins against the three different adversaries are lower compared to those of the Baseline case (as well as to the Frequency-based approach), when the penalty is automatic win. This is due to the fact that with the refusal to trade, the LA still has the chance of winning the game with the updates of the system only, which provides equal opportunities to both of the players though. Hence that case forces both of their

results closer to 50%, as no trades occur due to the adversary's penalty. The most effective of the three adversaries is the one which is based on the Case 1+2 (Naive turn), which is near to the Case 1+3 (Probabilistic turn) where the adversary takes into account environmental uncertainty (game's update) too. Case 1 (Plain lie) has a noticeable difference from the other two, as they introduce additional ways of effectively detecting the deception of the LA.

Our frequency-based adversaries of Sections 5.4.1 and 5.4.2 are better when the penalty is refusal to trade. On the other hand their performance is much worse than that of the adversaries which detect deception based on inconsistencies when the penalty is automatic win. The LA's already trained policy seems to have many weaknesses in that case. It was important for us though to evaluate how successful our trained policy of Section 5.2 was against these adversaries which identify logical incosistencies. Given the fact that the policy was not trained against an adversary with exposing capabilities, the results are positive for the Reinforcement Learning in generalised cases. Surprisingly, the LA trained policy of Section 5.2 performs much better versus the adversaries of this section compared to those of our Frequency based approach when the penalty is refusal to trade. Still there cannot be direct comparison though as the LA of the Frequency based approach was trained against those adversaries (of the Sections 5.4.1 and 5.4.2).

Scenario	LA wins		ADV wins		Draws	
Baseline (no detection)	59.170		39.755		1.075	
Detection by:	Refusal to trade	Automatic win	Refusal to trade	Automatic win	Refusal to trade	Automatic win
Case1: Plain Lies	55.725	39.996	42.295	58.895	1.980	1.110
Case1+2: Naive Turn	54.035	35.950	43.920	62.945	2.045	1.105
Case1+3: Probabilistic Turn	54.275	36.985	43.810	62.025	1.915	0.990
Frequency-based	50.86	49.7	46.33	46.225	2.81	4.075

Figure 5.7: *The winning rates (%) of the different exposing adversaries. All of the LA wins are lower compared to the Baseline case (as well as to the Frequency-based approach) against the three different adversaries when the penalty is automatic win. Please note that the LA uses a trained policy (that of Section 5.2) though. Image taken from [105].*

5.9 Conclusion

We developed a new algorithm based on SARSA(λ) that is able to produce better policies than those of our previous one (Chapter 4), although it requires more memory, in our trading game Taikun. Our environment is non-stationary (discussed in Section 3.3.6), as its dynamics change due to the agent's actions (i.e. manipulation) and due to the game-play mechanisms (i.e. trading proposal phase

- response phase - trading proposal phase etc.). However we produced RL policies which effectively learned in such noise, as the RLA was actually trained in two games in one: the "learning how to trade" game and the "learning how to respond" to the adversary's trades game. Our explicit linguistic manipulation based on scalar implicature dramatically improves the RLA's performance against a hand-crafted strict adversarial strategy with the realistic assumption that the adversary has a reason to be affected by it. Games versus restrictive agents and empirical analysis support that reason (Section 3.3.3).

Furthermore we learned successful RL policies against agents who can expose (i.e. detect) the RLA's manipulative moves and apply severe penalties. Even in such risky environments RL is capable of effectively using manipulation and win more often. When the risk is too high, policies have learned not to use manipulation at all and win without the risk. Apart from *how*, we showed that the RLA learned *when* is the right time to use manipulation and still be successful, thus avoiding excessive use that might result to detection. Finally, we tested our manipulative learned RL policy against more advanced adversaries who detect manipulation based on logical contradictions, and not on its frequency as before. We argued that these adversaries can be very successful at detection, and presented results regarding our trained RL policy's performance that were promising for the future in terms of generalisation. In the next chapter we will evaluate our trained policies against human players, in our game Taikun.

Chapter 6

Evaluating learned trading dialogue policies with humans in Taikun

We conducted experiments using human players in our game "Taikun". Two different kinds of experiments were conducted: the first one included human versus human games and the second one human against two different types of our trained RL agents. Those were either the manipulative one (from the Section 5.2) or the non-manipulative one (from the Section 5.1). Corresponding questionnaires were completed by the human players at the end of each experiment in order to collect valuable information for the future of this research, as we will see in more detail below.

6.1 Human vs. Agent

These experiments were very important because they offered us the opportunity to evaluate how efficient the RL trained policies were against human players. Hence a version of the game Taikun was programmed in Java where a human player can play the game versus either our trained non-manipulative RL agent or versus our trained manipulative one (Figure 6.1). In this experiment 10 people[1] participated and 60 games in total were played. Each of the players played 3 games versus the manipulative agent and 3 games versus the non-manipulative one.

None of the players knew how many different agents they were playing against or which of the agents they were playing against, as the first game was versus an

[1]The participants were all students, undergraduates and postgraduates.

agent at random. After that, the next game would involve the other agent and then the one of the first game and so on. We made sure to have thirty games where the non-manipulative agent would start first, and thirty games where the manipulative one would begin first. At the end of each of the six games the players completed a game questionnaire and then, after all six games were played, they completed an overall questionnaire that we will examine below.

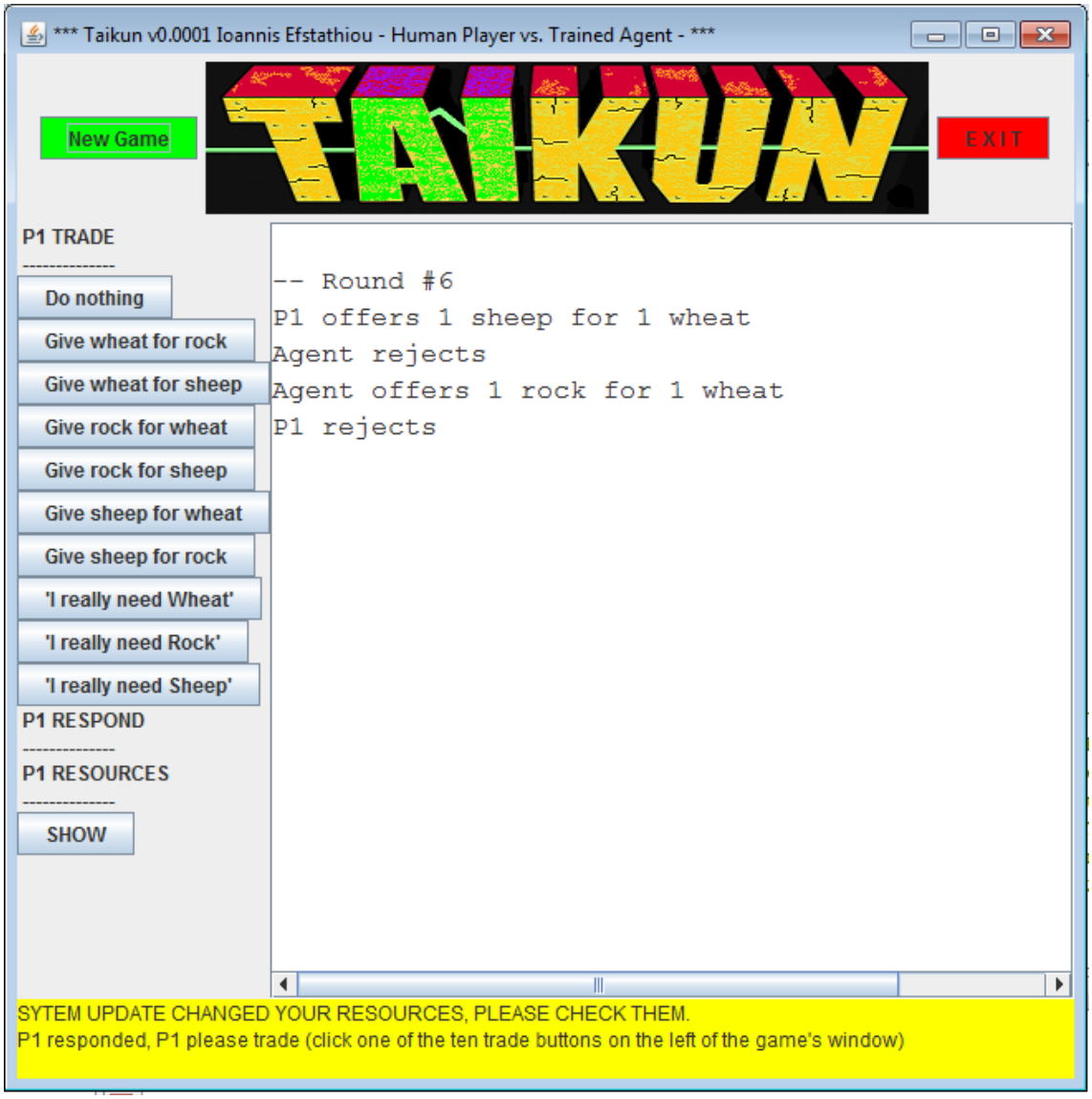


Figure 6.1: *"Taikun" version of human player vs. trained RL agent.*

6.1.1 Game questionnaire

The questionnaire below was completed by each of the players after playing each of the six games:

1. Did you win the game? (Yes/No)
2. Please rate the difficulty of this opponent (1 for easiest to 10 hardest):
3. How would you describe this opponent in terms of the strategy that it used?
4. Did you use any of the announcements (i.e. "I really need sheep") and why?
5. Please explain your strategy during the game (e.g. initially I was focused on reaching my goal resources, then I would never give a specific resource etc.)
6. Did you understand what goal resources your opponent needs? If yes, please write them.
7. Would your strategy be different in a future game with the same goal resources and opponent? If yes, what would you change?

6.1.2 Overall questionnaire

The questionnaire below was completed by each of the players after playing all of the six games:

1. Did you believe your opponent when it said that it really needs a resource?
2. If you thought it was lying, why did you think that?
3. When (if ever) did you start to disbelieve your opponent?
4. Did the "I really need <resource>..." announcement of the adversary cause you a dilemma as to whether you should keep following the same strategy or perhaps change it in order to win?
5. If the answer to question 1 was "yes", did you change your strategy? Please explain how.
6. Do you think that by exclusively aiming to reach your goal is an equally successful strategy with that of exclusively boycotting one specific resource? If yes, please write which resource you think that is. Why do you give this answer?
7. Any other comments:

6.1.3 Questionnaires' results and discussion

According to the questionnaires that the ten players completed while participating in the experiment, the results can be found below.

Objective measures:

• Agents have 31 wins and humans 31 (2 draws included).

• Manipulative agent has 15 wins and non-manipulative has 16.

• No one guessed correctly what kind of resources the agents really need (wheat and rocks) (humans though seemed to infer easier what their human opponents needed as we will see later) according to their answers (Question 6. Section 6.1.1). This is an advantage point for the agents as the manipulative one seems to confuse the human players. That might be one of the reasons for their successful (50%) win rate. Frustration has been previously observed by Kraus et al. [63], as people do not always follow equilibrium strategies and the "fixed" behaviour of the agent (in our case caused by following the RL optimal policy) sometimes confused them and resulted to the negotiation's termination with no agreement. In our manipulative case, the frustration that was caused due to repetitions of particular actions (as in Kraus case) and contradictions (for the sake of manipulation) was a significant reason that the humans lost games, as they failed to follow and exploit the agent's trading patterns.

• 4/10 people thought that the agents need wheat (Question 6 of Section 6.1.1 and 3/4 of them won at least 50% of the games (2/4 won 66%).

Subjective measures:

• The total score[2] of the manipulative agent is 157 and of the non-manipulative one 157 ((Question 2. Section 6.1.1)).

• 1/10 people thought the manipulative agent never lied (Question 1 of Section 6.1.2).

• 7/10 people thought the manipulative agent sometimes lied (Question 1 of Section 6.1.2).

• 2/10 people thought the manipulative agent always lied (Question 1 of Section 6.1.2).

• 3/6 (50%) people answered "yes" to the dilemma question of the adversary's manipulation and 3/6 answered "no" (Question 4 of Section 6.1.2). This result motivates and explains the behaviour of the gullible adversary of Section 5.2, as we have discussed in Section 3.3.3 (2nd reason).

[2]This rating was given by the human players to each of the agents at the end of the game (Question 2. of the game questionnaire in Section 6.1.1) and indicates how hard each of them was to play against.

• 2/4 people thought that the available announcements (i.e. "I really need wheat") actually helped them to win games and 2/4 people didn't (Question 4. Section 6.1.1).
• 4/10 people thought that by aiming only to reach your resources is an equally successful strategy with that of only boycotting one resource. 6/10 answered "no" (Question 6 of Section 6.1.2). This result relates to those of the restrictive adversaries in Section 5.3 and shows that 60% of the people find it hard to believe that the boycott of a resource can be as powerful as aiming only to reach the resources. Our results there show that it is true though, agreeing with the 40% of the people who believed that the two different strategies are equally powerful. These percentages, which are close to each other (1 person made the difference), suggest again that the the gullible adversary of Section 5.2 has a valid reason (like humans) of being in a dilemma after it hears the manipulative dialogue action (e.g. "I really need sheep") and choosing to restrict (boycott) a particular needed resource.
• 8/10 of the people thought that the game is based on skill and 2/10 thought that it depends entirely on luck (this information was mainly inferred from Questions 5 and 7 of Section 6.1.1 and Questions 3, 5 and 7 of Section 6.1.2). Taikun is heavily based on luck (due to the random update of the resources turn) and it is surprising that the human players did not realise that. It requires skill though too.
• 3/10 of the people in most of the cases thought the agent needed wheat and sheep (Question 6 of Section 6.1.1, sheep was frequently used as a lie by the non-manipulative agent so they are evidently affected by the manipulative agent).

6.1.4 Human comments on the manipulative agent

• **"Confusing"** (7 people)
• **"He would mostly not trade or respond "no" to most trading proposals"** (6 people)
• "Persistent on its requests" (4 people)
• "Good strategic player with meaningful actions (3 people")
• "Focused on goal" (3 people)
• "Obstinate" (3 people)
• "Simple and normal" (2 people)
• "Not willing to sacrifice resources" (2 people)
• "Cheat" (2)
• "He was giving resources that (author: they thought) it did not need"
• "He seems to give in to opponent faking risks"
• "Willing to give a win"

- "Difficult to cheat"
- "More lucky than me"
- "It asks for specific resources"
- "Difficult"
- "Announced and offered a lot of times"
- "It stole my two resources in the first two turns"

6.1.5 Human comments on the non-manipulative agent (goal-oriented only)

- **"Rejects all proposals"** (7 people)
- **"The least active on offers"** (6 people)
- "Clear strategy" (3 people)
- "Quite honest" (3 people)
- "No strategy" (3 people)
- "Goal-oriented" (2 people)
- "Relies on the random element" (2 people)
- "Unpredictable " (2 people)
- "Persistent"
- "No announcements"
- "Active on offers"
- "Normal"
- "Offering different things to what I propose"
- "Clever"
- "Repetitive"
- "Cannot be tricked"
- "He is meant to win"
- "Suggesting to trade same resources as I did"
- "Would not trade anything else than rock"
- "Wait for random gains"
- "It offered each of the resources at the beginning"

6.1.6 Discussion and conclusion

The positive side from the above experiments was that the two agents were as successful as the human players. The manipulative agent was characterised by 70% of the human players as confusing and that indicates that its implicatures (i.e. "I really need sheep") were successful against humans. The fact that none of the players managed to guess correctly its resources suggests that too. The manipulative agent was characterised as a good strategic player by 30% of the people, while the non-manipulative one was characterised as having a clear strategy by the same amount of people. Those results suggest that a significant amount of people understood that there was intelligence behind both of the agent's actions. The non-manipulative agent showed its real goal-oriented behaviour as more than one participant characterised it with "Clear strategy", "Quite honest", "Goal-oriented". The manipulative one, despite the fact that it mainly won by confusing the human players (and that was intended as a result of the manipulation), managed to also show its underlying sensibility that its implicature is based on, in some cases, by receiving characterisations such as "Persistent on its requests", "Obstinate", "Focused on goal".

On the negative side, it seems though that people who tried to win, lost most of the time and then they soon decided to do nothing (i.e. stop trading) because that was 50% successful (and they realised that quite early by human intuition). Taikun is highly based on luck so even when someone plays a couple of games at random versus our manipulative agent she/he will probably end up winning by 50%. Only if they will play a significant amount of games we will be able to see difference in the winning rates that would indicate that skill matters too. This is due to the high noise (uncertainty) that the random resource update turn creates. We also wanted to show that the manipulative agent wins the (gullible) human players more often than the non-manipulative one: in this case the humans should behave in exactly the same gullible way as that of the rule-based adversary that we have in Section 5.2.

From the questionnaires' answers of Section 6.1.3 we saw that the above assumption was not always the case though, as 60% of the people thought that being goal-oriented and restricting a particular resource are not equally successful strategies and 50% were actually in a dilemma as to what they should do after they saw the agent's manipulative action. However, even if that was the case and all of the human players would behave in the gullible way and would start restricting resources, luck highly affects the game and only after a significant amount of games we could prove that. Unfortunately the (full) skill of the agents (manipulative one and non-manipulative one) is not evident by just playing a few games and especially when there is no trading. Furthermore the random resource update at the beginning

of each new round is noisy and reinforces the luck factor. It was necessary though because the initial number of resources was insufficient to reach the goal and it also promotes trading. To conclude, more games should be played, we had to find more ways of motivating the players to trade and also reduce the noise of the random resource update (i.e. restrict the luck factor). These conclusions were taken into deep consideration in Chapter 10.

6.2 Human vs. Human

Another version of the game was programmed with the aim of using it between two human players only for data collection. We then decided though that it would be even more natural and inspiring for the players to play between them using real cards, dice and a human referee. In that way we were planning on transcribing the human utterances. Ten human players[3] took part in this experiment and five sessions of 30 minutes each (they usually consisted of 2-3 games) were played between each two of them. All of their games were recorded as we wanted to investigate what human players do and say while playing Taikun. The instructions that were given to the subjects were the rules of the game (Section 3.1.1).

6.2.1 Questionnaire

The questionnaire below was completed by each of the ten players after playing a number of games in thirty minutes:

1. How many games did you win and how many games did you play in total?

2. Did you make any remarks such as "I really need wheat", "I've got lots of rock" etc. other than only trading during the game? Please mention them.

   If no, please proceed to question 3.

2a. If yes, why did you use them? Please explain.

2b. Do you think they had some kind of effect to your opponent's strategy after using them? Did his/her play style change?

2c. Is there anything else important that you would say to your opponent and the above announcements do not express? If yes, please give examples.

[3]The participants were all students, undergraduates and postgraduates.

3. Please now explain your overall strategy during the games (i.e. initially I was focused on reaching my goal resources, after that I would never give a specific resource etc.):

4. Did you understand what goal resources the other player needs? If yes, please write them.

5. Would your strategy be different in a future game with the same goal resources and opponent? If yes, what would you change?

6.2.2 What did people say during trading?

Here we have a list of interesting (from a linguistic perspective) human utterances that were extracted from the recordings of each session:

Session 1

- Would you like to trade *resource* for *resource*? (This type of utterances inspired us to use implicit manipulation as we will see in Chapters 8 and 9. That means manipulation through normal trading actions but not in a form of a question though, as it is in the example.)
- I don't have *resource* (I don't have any *resources*)
- Multiple *resources* for multiple *resources* (usually people ask for more and give less. This information was used in the trading actions of the agents in Chapters 8 and 9 where we include "give 1 for 2" actions.)
- I have nothing to trade with you

Session 2

- The same as before (repeats offer)
- I still want *resource*
- Any *resource* suggestions? (any trades that would involve *resource*?)
- I will give you *resource* for *resource* and when I will have an extra *resource* I will give that extra to you, if you *trust* me...

Session 3

- What resources would you like?
- Would you like a *resource*?

- Not now (response to a trade proposal)
- How about *resource* for *resource* and *resource*?
- Sounds good but I will skip it this time (reply to a trade proposal)
- How about *resource* and *resource* for *resource*?
- Maybe next time (reply to a trade proposal)
- 2 *resources* for 1 of each

Session 4

- Do you have anything you want to give me for free..? I don't have anything to give you back though...
- I am willing to give more than one *resource* for *resource* and/or *resource*
- I'll give you *resource A* for *resource* B. Answer is "No". Ok, then I'll give you *resource* C for *resource B*?
- I'll give you *resource* for anything you might want to give me?

Session 5

- I don't think I want to ask for a trade
- Would you trade anything for *resource*? (same as the 2nd above) ...if you CAN give me a *resource*...?
- Response: I can't do that
- We both want *resource* by the sound of it..?

Most of the people mainly offered one resource for one other, they frequently did nothing on their trading turn and in most of the games they tried to infer what their opponents need. The above utterances were very helpful for the rest of this work and some of the information was used in the experiments of Chapters 8 and 9 that follow, as we mentioned above.

6.2.3 Conclusions on human game-play in Taikun

Humans seemed to be very cautious when they play with each other and they frequently did nothing. They avoided trading and especially accepting trading proposals. They instead mostly preferred to be based on the system's random factor (luck), that updates the resources at the beginning of each round. The reason that we implemented that though was because the initial number of resources was insufficient to reach the goal and it also promotes trading. The resource update slowly

"pushes" the players towards the goal by producing more resources than it takes (discussed in Section 3.1.1). When the humans proposed trades, they lied quite often and tried to infer what the goal resources of their opponents were. The games "versus the agents" had more trades than those between the human players and that shows that humans were more comfortable trading with agents than with other humans. Both humans and agents were more active with their trading proposals and trade acceptances in the games against the agents. The agents have successfully learned how to trade and they won equally with the human players, without using the "Do nothing" action most of the time (as humans did in "human vs. human" games). That required skill. The conclusions from Section 6.1.6 though would hopefully reveal the agents' skill more in another future attempt. In Chapter 10 we had the opportunity to test again trained policies versus humans and use the knowledge that we gained from the experiments of this chapter.

Chapter 7

Main model: Catan

After the completion of our experiments in Taikun we proceeded by experimenting in the (more complex than Taikun) non-cooperative board game "Catan". To investigate non-cooperative dialogues in a controlled setting we used a 2-player version of the game first (experiments will be discussed in detail in Chapter 8), which is a complex, sequential, non-zero-sum game with imperfect information. Our version of the game is focused only on the trading part, as it is the main interest of this research. We call the 2 players the "adversary" and the "Reinforcement learning agent" (RLA) or the "hand-crafted agent" (HCA). The adversary is always used for playing against the RLAs or the HCA.

At the beginning of the board game Catan, the four players place a couple of settlements and roads on the board (map), which consists of hexagons. Each of them represents one of the five different resources of the game, which are wheat, wood (or timber), rocks, sheep and clay (or bricks). Each of the hexagons has its own number (2-12), and by rolling the dice at the beginning of each turn all players receive resources from the the hexagons whose number was rolled, providing that they own settlements or cities which are next to these hexagons. The roads allow the players to build more settlements in other areas of the board in order to expand their controlled resources. The resources are represented by cards that the players hold on their hands. They can trade them with other players. Trades happen through dialogue, as the players usually state the number and type of resources that they offer along with those of the resources that they need. The goal of the game is to reach ten victory points. These are gathered by building constructions and completing achievements (e.g. owning the longest road in the game).

The constructions that the players can build are four: a road, a settlement, a city (which is an upgrade of the settlement) and a development card. The development card is not exactly a construction but an action or feature that the player has available to use in the game (e.g. soldiers, and the players who have the most get extra victory points). The road requires 1 timber and 1 brick to be built, the

settlement 1 wheat, 1 timber, 1 sheep and 1 brick, the city 2 wheat and 3 rocks and the card 1 wheat, 1 rock and 1 sheep. At the beginning of Chapter 9, in Figure 9.1, a screenshot of the game is shown in the JSettlers [99] research environment, which is discussed in detail there.

7.1 RLAs and upgraded SARSA(λ)

From this chapter onwards all of the experiments have been conducted using a new, upgraded SARSA(λ) algorithm for the RLAs. Based on previous experience and results, we decided to keep using gradual decrease of the learning rate from 1 to 0, as it has shown to result in smooth training. Likewise, we also gradually reduce the e-greedy ratio from 20% to 0% exploration for the LA, as it has also shown to result in effective learning. We also decided to create dynamically the policy's state-actions while learning occurs, instead of fully creating them a priori, as that would require a lot of memory for maintaining states which might not ever be used. Furthermore, with the additional state compression methods that we added (Section 7.2.5) and we will introduce from this chapter onwards, we achieved a huge reduction in running times, less demanding memory requirements and policy performances which were nearly as good as those of a numeric tabular RL state representation[1].

7.2 Design

In our version of the game, the RLA or the HCA proposes trades to the adversary sequentially and tries to reach a goal number of resources (in the case of a city: 3 rocks and 2 wheat) within seven[2] trading proposals. A game in our case is considered to be a maximum of seven trading proposals along with the responses. The LAs had to learn how to reach their goal resources (or as near as possible) within those seven attempts. As we have discussed in the beginning of this chapter, there are four different constructions (or mini-goals, as part of the main goal, which is to collect 10 victory points) that can be built in the normal "Catan" game: a road,

[1] After the algorithm's modification, where the states became compressed, brief experiments were conducted. They showed that the difference of the policies' performances between the normal (numeric) states algorithm and the one with compressed states was not important, especially considering the running times and memory requirements which improved dramatically after the modification.

[2] Nothing particularly hinges upon this number. We decided to use seven trading proposals mainly because it would not be very monotonous for a human player to respond and seven attempts seemed to be challenging to reach the goal.

a city, a settlement or buy a development card. Our non-manipulative RLA has learned how to successfully trade in order to achieve all those "goals"[3] but most of our experiments that will follow are based on the example case of the city. As we have seen there are five different resources to trade (wheat, timber, rocks, sheep and bricks) and the adversary only responds by either saying "Yes" or "No" to accept or reject the trade respectively in our case. Initially we assume that the adversary has all of the resources available to give so it is up to the RLA or the HCA to use a successful strategy that will allow it to reach its goal. The learning agents start the game with a random number of resources (up to 7 of each resource) and therefore there are cases where the initial number of resources is insufficient to eventually reach their goal. The agents still learn though how to get as close to the goal as possible (due to the reward function which we will examine in Section 7.2.3).

7.2.1 Actions (Trading Proposals)

Trade occurs through trading proposals that may lead to acceptance or rejection from the adversary, and have deterministic and stochastic effects. In an agent's proposal (turn) only one "give 1-for-1" or "give 1-for-2" trading proposal may occur, or nothing (41 actions in total):

1. I will do nothing
2. I will give you a wheat and I need a timber
3. I will give you a wheat and I need a rock

...

40. I will give you a brick and I need two rocks
41. I will give you a brick and I need two sheep

The agents which use manipulation in the experiments that will follow, manipulate their adversaries through all of the above trading proposals (implicit manipulation) as we will see in detail in Section 8.2.3. They do not use scalar implicature (e.g. "I really need sheep") any more, which is explicit manipulation, as we examined in the previous chapters.

[3]We call the constructions *goals* because in our experiments we focus only on the trading part, where the LA only aims to build one of them. We assume that there is already a build plan to use which aims to gather 10 victory points by indicating which construction to build next, as we will examine later.

7.2.2 The RL Agents (RLA)

The game state is represented by the RLA's encoded (Section 7.2.5) set of resources. The RLA plays the game and learns while perceiving (initially) only its own set of resources. It is aware of its winning condition in as much as it experiences a large final reward when reaching this state. It learns how to achieve the goal state through trial-and-error exploration while playing repeated games. Each game consists of up to 7 trading proposals, but nothing particularly hinges upon this number – we have experimented with a number of different length constraints, and obtained similar results. The agent is modelled as a Markov Decision Process [97]: it observes states, selects actions according to a policy, transitions to a new state (due to the adversary's response), and receives rewards at the end of each game. This reward is then used to update the policy followed by the agent using the SARSA(λ) algorithm (discussed in Section 2.1.6).

7.2.3 Reward function

The reward function used in all the experiments takes into consideration the number of trading proposals made and the distance from the goal, as well as trading success. In detail, the distance is calculated by subtracting the achieved numbers of the goal resources from the goal numbers, and then adding all of the results together. The reward function that is used is: + 10,000 (if trading successful) $-(1,000^*$ proposals) $-(1,000^*$ distance). It was created mainly based on the RL experience that we had accumulated up to that point. The idea was to teach to the RLA to get as close to the goal (i.e. to collect the goal number of resources) as possible, because there were cases where it would start the game with an insufficient amount of resources.

7.2.4 Training parameters

The agents were trained using a SARSA(λ) learning method [97] with an initial exploration rate of 0.2, which gradually decays to 0, and a learning rate α of 1, which also gradually decays to 0 by the end of the training phase. After experimenting with the learning parameters we found that with λ equal to 0.9 and γ equal to 0.9 we obtain the best results for our problem and therefore these values have been used in all of the experiments that follow.

7.2.5 State Encoding

To overcome issues related to long training times and high memory demands, we implemented a state encoding mechanism that automatically converts all of our trading game states to a significantly smaller number states in a compressed representation. The new state representation takes into consideration the distance from goal and the availability of the resource, as well as its quality (goal or non-goal resource) and uses 7 different characters. The agent's state consists of the quantities of the five resources that it currently has available. In the case of the city, it needs wheat and rocks. That means two out of five resources are goal resources and therefore they can be represented by G (goal) when their number is equal to the goal amount, N (null) when their number is 0, M (more) when their number is more than the goal-quantity, and 1 or 2 when the distance from the goal quantity is 1 or 2 respectively. The 3 non-goal resources are represented by Z (zero) when they are 0 and A (available) when they are more than 0.

For example, the state $\langle 1, 4, 3, 0, 2\rangle$ would be encoded to $\langle 1, A, G, Z, A\rangle$, assuming that the numeric goal would be $\langle 2, 0, 3, 0, 0\rangle$. The numeric state space of our problem has 8 x 8 x 8 x 8 x 8 (=32,768) states (we track up to 7 of each type of resource) that are encoded to only 4 x 2 x 5 x 2 x 2 (=160) states. This is reduced to 0.5% of the original size of the state space. With this method and despite the fact that the representation still remains tabular, in all of our experiments of the next chapter 3 million training games required only around 10 minutes to finalize. The performances were very successful too as the logic is still based on the precision of the RL tabular representation.

7.3 Experiments background

The experiments that will follow on the next two Chapters 8 and 9, are conducted on our 2-player version of the game, and JSettlers [99] research environment respectively. In Chapter 8 we initially present the results of experiments with the assumption that the adversary is affected by *all* of the trading proposals of the learning agents, in such a way that it tries to stop the learning agents from getting the resources that they say they need. Intuitively, this is a basic aspect of adversarial behaviour. It is a reasonable and very successful strategy that may originate from sound reasoning as we have shown in our previous chapter with the Restrictive Adversaries (Section 5.3). The reasons of this behaviour have been discussed in detail in Section 3.3.3. We then present the results of experiments between non-manipulative RLAs. The point is to show that the RLAs which maintain the

adversarial preferences in their state space increase their performance compared to an RLA which does not do that. In Chapter 9 we used JSettlers [99], which is a research environment built using Java and it is based on the full version of the Catan board game (multilateral negotiations environment). There we will test learned RL policies from Chapter 8 (including a manipulative one) and show that they are very effective, despite the fact that they were trained on a bilateral negotiations environment.

Chapter 8

Experiments in Catan: Opponent models

All of the experiments of this chapter are based on our 2-player version of the board game "Catan" that we examined in the previous chapter. The adversary only responds by either saying "Yes" or "No" to accept or reject the LA's proposed trades respectively which can be up to seven as we have mentioned before. Hence a game in our case is considered to be a maximum of seven trading proposals along with the responses. The LAs had to learn how to reach their goal resources (or as near as possible) within those seven attempts. Initially we created a non-manipulative (i.e. its actions do not have any manipulating effect) SARSA (λ) RL agent which learned how to successfully trade in the game. It learned how to do that for all of the different goals (road, settlement, city and development card). The adversary always accepts the agent's trading proposals, it has initially an infinite amount of resources and no goal. After that we created two RL agents which -through two different kinds of implicit manipulation- play the game versus an adversary which can be manipulated and hinders their strategies [28]. At the end of this chapter, we present experiments where non-manipulative RL agents include the adversarial preferences in their state space. They play our Catan game versus an adversary which sets a random goal at the beginning of the game, and therefore its preferences change according to that.

8.1 Initial Experiments

Before we examine the cases with manipulation and the adversary's opponent model, we first explore the case of learning a trading policy for adversaries that do not have an opponent model and thus do not try to hinder the learning agent. This

adversary always accepts an agent's trading proposal, and so this serves as an initial proof-of-concept of the extent to which the game is winnable by the learning agents if the adversary is being fully cooperative. We assume that this adversary has an infinite amount of available resources. Here the (non-manipulative) RLA learned how to successfully trade in the full version of the "Catan" game for every goal case. These include building a road, a city, a settlement, or a development card. The different goals are different numbers and types of resources that the RLA needs to gather in order to win.

The RLA has located a successful policy for each one of those cases, showing that the cooperative version of the game is solvable as an MDP problem. As we see in Figure 8.1, in the case where the goal is to build a city it learns to win 96.8% of the time (not 100% due to the cases with insufficient initial resources). It has identified and taken advantage of the power of the "give 1-for-2" over the "give 1-for-1" trades and therefore it uses them much more frequently (with a ratio of around 75% over 25% for the "give 1-for-1"). The adversary that it plays against does not have an opponent model, the learning agent's trading proposals do not affect it, and the adversary always accepts them. Hence we initially show that RL is capable of successfully learning how to trade in this version of the game (with every different goal) while learning to also exploit the "give 1-for-2" trading proposals.

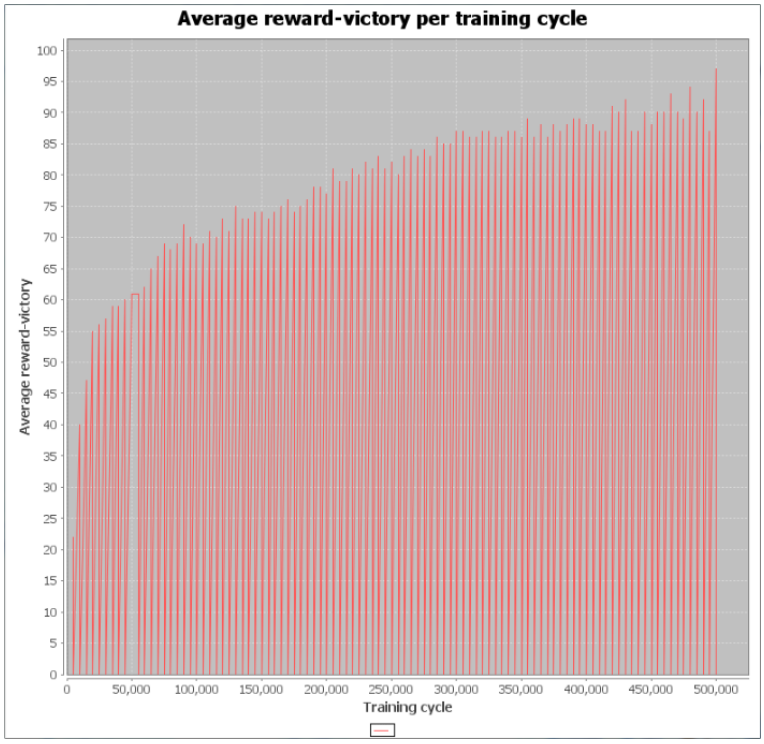


Figure 8.1: *Learning Agent's reward-victory graph in 500 thousand training games of Initial Experiment: building a city, cooperative adversary.*

8.2 Manipulation

In the following experiments we assume that all of the trading proposals (apart from "I will do nothing") affect the opponent model of the adversary (implicit manipulation). Hence a trading proposal may or may not lead to a trade (the action's stochastic effect), but it will definitely affect (action's deterministic effect) the adversary's belief model (opponent model). Here we will discuss each action's deterministic effect. Each of the trading proposals consists of two parts: the offered resource and the wanted one(s). The adversary's opponent model is affected by both of these parts – for example the more often the agent insists on asking for wheat, the less the adversary will be eager to give it in an attempt to hinder the LA (discussed in Section 8.2.3). Hence the agents need to learn how to appropriately use this effect in order to successfully manipulate the adversary and reach the goal number of resources. Manipulation changes the probabilities (weights) of actions in the way that we will examine in Section 8.2.3 (which is similar to that in Section 5.2). However here the offered resource and the wanted resource of each of the LA's trading proposals increase and reduce respectively the corresponding weights of the adversary's responses.

8.2.1 Corpus analysis

An example of the type of human non-cooperative dialogue behaviour which we are generating in this part is given by our (dishonest) trading player agent A in the following dialogue:

A1: "I will give you a wheat and I need 2 clay"[A lies - it does not need clay but it needs wheat]
B1: "No"
A2: "I'll give you a rock and I need a clay"[A lies again and it actually needs rocks too, but it does not have any rocks to give]
B2: "No"
A3: "I'll give you a clay and I need a wheat
B3: "Yes"

Here, B is deceived into providing the wheat that A actually needs, because B believes that A needs clay (A asked for it twice) rather than wheat and rock (that it offered). Similar human behaviour can be observed in the Catan game corpus [1] a set of on-line trading dialogues between humans playing Catan. We analysed a set of

32 logged and annotated games, which correspond to 2512 trading negotiation turns. We looked for explicit lies, of the form: *Player offers to give resource X (possibly for Y) but does not hold resource X* - such as in turn A2 in the above example.

11 turns out of 2512 were lies of this type. Since this corpus was not collected with expert players, we expect the number to be larger for more experienced negotiators. Other lies such as asking for a resource that is not really wanted, cannot be detected in the corpus, since the player's intention would need to be known.

8.2.2 Actions (Trading Proposals)

Trade occurs through trading proposals that may lead to acceptance or rejection from the adversary, and have deterministic and stochastic effects. The action's stochastic effect is whether or not the trade proposal (action) will be accepted. In an agent's proposal (turn) only one "give 1-for-1" or "give 1-for-2" trading proposal may occur, or nothing (41 actions in total for the case of the dishonest RLA):

1. I will do nothing
2. I will give you a wheat and I need a timber
3. I will give you a wheat and I need a rock

...

40. I will give you a brick and I need two rocks
41. I will give you a brick and I need two sheep

In contrast to the case of the dishonest RLA, the cases of the honest RLA and the naive HCA that we will examine consist of 17 of the above actions because they ask only for goal resources (rock and wheat). The adversary responds by either saying "Yes" or "No" to accept or reject the learning agent's proposals. Each of these actions affects the adversary's opponent model as described below.

8.2.3 The Adversary and its Opponent model

The adversary remains the same in all of our experiments. We assume that it has an infinite amount of available resources, like the adversary which is discussed in Section 8.1. It is in fact the same adversary, with a *further* implementation of the intuition that a rational adversary will act so as to hinder other players in respect of their expressed preferences. Opponent models (OM) with hindering abilities

have previously been shown to be important in games such as the "Machiavelli" card game [8]. Hence our adversary is using an opponent model that is based on hindering the LA's preferences, as the LA expresses its preferences through trading proposals and this is the only information that the adversary receives. Since opponent modelling is focused on using knowledge about other agents to improve performance, the adversary therefore hinders the LA's announced preferences (trading proposals).

Our model is based on this approach to OM and uses knowledge (from the LA's announcements) in an effort to improve its performance. Unlike the OM [16, 49, 50] or the PrOM search model of [24] though, it does not explicitly predict the moves of the LA, but the history of those moves are used to direct the adversary's future responses. The adversary therefore uses an opponent model which directs its responses to the other agent's (RLA or HCA) trading proposals. Every time that an agent utters a trading proposal, probabilities of the adversary giving resource types change accordingly (as we will see below), and therefore the adversary becomes more or less eager to give some resources than others. It does this because it tries to hinder the other players from acquiring the resources that they ask for. For instance, if an agent insists on asking only for wheat then the probability that it will be given becomes very low (the adversary considers it now as valuable), but the relative probability that it will get one of the other four resources increases.

However here the adversary also takes into consideration what the agent offers to *give*, so the more an agent keeps offering a resource the more likely it becomes for the adversary to give it too (it considers the resource as less valuable). In detail, at the beginning of each game the probabilities (weights) that represent the adversary's willingness to give (i.e. through responses to the LA's trading proposals) each of the five resource types start equally (each is 20%). When the LA asks for a resource then the adversary's probability to give that particular resource (through its responses) is reduced by either 4% or 6% (if it is a "give 1-for-1" or "give 1-for-2" trade proposal respectively), and the probabilities of giving the four other resource types increase accordingly (i.e. equally, each increases by either 1% or 1.5% respectively). The probability of giving the LA's *offered* resource also increases by 4% and those which give the other resources decrease accordingly (i.e. equally, each one decreases by 1%). As the adversary responds only to the LA's trading proposals, these probabilities actually apply to the various adversary's responses (i.e. "yes" or "no") to the LA's proposals. We experimented with a variety of different increments, and similar results were obtained to those presented below, so nothing particularly hinges on the 4% figure[1]. Due to this opponent model, it is possible to manipulate the adversary into eventually giving resources that are needed, if the right trading proposals are made.

[1]Manipulation would be much more "rapid" if the percentage was for example 30%, instead of 4%.

8.2.4 The Honest Reinforcement Learning Agent - "The Good"

The honest RLA only asks for resources that it really needs (therefore it is restricted to 17 out of the 41 actions). It is a sincere RLA and it only proposes a trade after it has checked that the offered resource is indeed available. However, the fact that it still learns how to successfully manipulate (legitimately persuade) the adversary under those honest constraints, and in a continuous non-stationary MDP environment due to the ever-changing adversarial belief model (i.e. the environment's dynamics can change after an action is selected), makes the outcome surprising. In the experiments that follow we will see that it locates a honest way of persuading its adversary (e.g. it says "I will give you 1 timber for 1 rock", and it needs rock as its goal is to gather 2 wheat and 3 rocks and build a city).

8.2.5 The Dishonest Reinforcement Learning Agent - "The Bad"

The dishonest RLA can ask for resources that it does not need (therefore it uses all of the 41 actions). It can also propose trades without checking if the offered resource is available. If such a deceitful trading proposal gets accepted by the adversary, the RLA then refuses to actually make the trade. Thus its learning process is a harder Reinforcement Learning task than that of the honest RLA (since it has more actions). However, it still learns how to successfully manipulate (deceive) the adversary under those dishonest conditions, and in a continuous non-stationary MDP environment due to the ever-changing adversarial opponent model as above, resulting on a surprisingly equal performance with that of the honest RLA. As we will see in the experiments that follow, its strategy is based on the use of lies (e.g. it says "I will give you 1 timber for 1 sheep", but it does not need sheep as its goal is to gather 2 wheat and 3 rocks and build a city).

8.2.6 The Naive Hand-Crafted Learning Agent - "The Ugly"

This agent is not a learning agent but instead uses a hand-crafted naive strategy. In detail, it uses a reasonable way of proposing trades by checking the availability of the resources that it does not need and offers them for those that it needs in an equi-probable manner. The reason that we call it naive (as well as "ugly") is

because it does not take into consideration the fact that its trading proposals affect the adversary's opponent model and -instead of learning that- it just keeps following the same naive rule-based strategy. This agent is a baseline case and despite the fact that its strategy is quite sensible, we show that it is significantly worse than that of the two manipulative RLAs.

8.2.7 Naive HCA vs. Adversary: Experiment 1 (Baseline)

The naive HCA played 3 million games against the Adversary in Experiment 1. This is our baseline case for comparison. The agent's trading proposals affect the opponent model of the adversary but the agent is unaware of that and therefore it does nothing about it. It just keeps playing the game based on the naive but reasonable strategy discussed in Section 8.2.6.

8.2.8 Honest RLA vs. Adversary: Experiment 2

In this experiment we trained the honest RLA against the adversary in 3 million games. The RLA's trading proposals affect the opponent model of the adversary and we show that, despite the honest constraints, the honest RLA can learn how to successfully manipulate the adversary. Ultimately we show that the performance is better than that of the baseline case in Experiment 1. The performance of the Honest RLA before training (i.e. random action selection) is about 21% (win rate).

8.2.9 Dishonest RLA vs. Adversary: Experiment 3

In this experiment we trained the dishonest RLA against the adversary in 3 million games. The RLA's trading proposals again affect the opponent model of the adversary and we show that the dishonest RLA can learn how to successfully manipulate it. As above, we show that the performance is better than that of the baseline case in Experiment 1. Furthermore, we explore how well this deceitful RLA performs compared to the previous honest one, which legitimately persuades. The performance of the Dishonest RLA before training (i.e. random action selection) is about 4% (win rate).

8.2.10 Results

The RLAs were trained on 3 million games against the Adversary. Their policies were then tested in 20,000 games. The HCA played 3 million games too against the same adversary. As there was no learning in this case, no testing games were played because its performance remained stable throughout the 3 million games as we will see below.

8.2.11 Naive HCA: Experiment 1 results

The naive HCA has a win rate of only 25.3%. This is shown by a yellow horizontal line in the graphs below. Its strategy, 50% of the time focuses on asking for wheat by offering each one of its available unwanted resources in turn, and 50% of the time asking for rocks using the same technique.

8.2.12 Honest RLA: Experiment 2 results

The honest RLA scored a winning performance of 35.8%, see Figure 8.2, starting from 21.1% (which is the performance of random action selection). Its strategy focuses on asking initially for either wheat, until it gathers rocks, or for rocks until it gathers wheat that needs to build a city (2 wheat and 3 rocks are required). It also mainly offers resources that it needs (goal ones) -and has available- instead of non-goal ones as it will become then easier to get them back. This honest persuasive strategy proved to be very effective against the adversarial hindering policy.

8.2.13 Dishonest RLA: Experiment 3 results

The dishonest RLA scored a winning performance of 36.2% after 3-million training games (see Figure 8.3), starting from only 4.2%. That clearly shows that its task was much harder than that of the honest RLA in Experiment 2, which started from 21.1%, as it has to understand how to effectively manipulate through all of the 41 actions (rather than the 17 honest actions which ask for goal resources only). Nevertheless its very effective learned strategy mainly focuses on the use of lies. It asks especially for resources that it does not need only for the sake of manipulation (deception) and it offers resources that it does not have for the same purpose. The

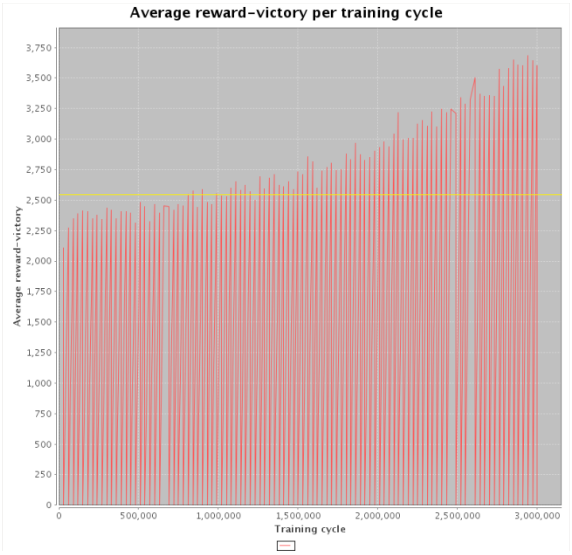


Figure 8.2: *Honest RLA's reward-victory graph in 3 million training games (experiment 2). Yellow horizontal line = Baseline performance (Naive HCA).*

type of the offered resources in this case are mainly goal ones again (as above) and the fact that this RLA can lie about their availability makes such offers even more frequent than before. This dishonest strategy proved to be equally effective with that of the honest RLA though.

Both of the RLAs (as we saw in Experiment 2 too) managed to learn successful strategies despite the fact that there are cases where the initial resources are insufficient to reach the goal within 7 trading proposals. They both realized again (as in our Initial Experiment, Section 8.1) the power of the "give 1-for-2" over the "give 1 for-1" trades and they used them more often. Hence, in some cases they managed to approach their goals even with insufficient initial resources. By comparing the two manipulative cases to that of Experiment 1 we show that manipulation (through legitimate persuasion [Experiment 2] or deception [Experiment 3]) can be successfully learned by our RLAs and outperform by 11% a naive but reasonable strategy.

Exp.	**Learning Agent policy**	**Adversary policy**	**Agent's wins**
Initial	SARSA	Accepts every trade	96.8%
	Random Honest actions	Hinders agent's preferences	21%
	Random Dishonest actions	Hinders agent's preferences	4%
1	**Hand-Crafted Naive Honest (Baseline)**	**Hinders agent's preferences**	**25.3%**
2	SARSA + Honest actions	Hinders agent's preferences	35.8%*
3	SARSA + Dishonest actions	Hinders agent's preferences	36.2%*

Table 8.1: *Performance (% wins) of the discussed learning agents in 20 K testing games, after training. (*= significant improvement over baseline [Exp. 1]* ***in bold text****, $p < 0.05$)*

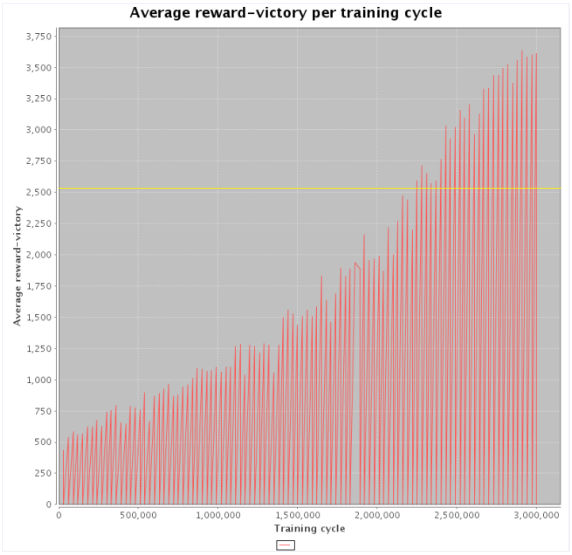


Figure 8.3: *Dishonest RLA's reward-victory graph in 3 million training games (experiment 3). Yellow horizontal line = Baseline performance (Naive HCA).*

8.2.14 Discussion: a Non-Stationary MDP problem

Our Experiments 2 and 3 also show that RL is capable of learning successful policies even in the case where the environment's dynamics change (maximum of 7 times per game due to the maximum number of trading proposals which is 7) and each action (trading proposal) has a stochastic effect (that of a possible trade) and a deterministic effect (that on adversary's opponent model because of the manipulation). Every time the honest or dishonest RLA proposes a trade, the opponent model of the adversary changes as we have seen. That means the environment changes too (as the adversary is a part of it according to the RLA's perspective) and therefore makes our problem a non-stationary MDP [91]. Despite the fact that only the RLA's actions are responsible for those changes and so the problem may be solved by recasting it into a stationary one through state augmentation [17], our case is more complex. This is because our RLA's actions affect the environment in two different ways (through their stochastic and deterministic effects). We also chose not to enrich the state representation because we wanted to see if RL is capable of handling our non-cooperative case with no further augmentations. Enriching the state is not always a good solution as we will see in Section 8.4, where the CP-NET RLA whose state representation consists of 25 slots, will not manage to outperform the NPRLA whose state representation has 10 slots. The state may include too much noise for the RL to handle and result in a good policy. In the above example of the CP-NET RLA, we will see that the state augmentation makes its task harder. However we have enriched in the past (e.g. Chapter 4) the LA's state representation with a feature that maintains information about the turn of the game (i.e. trading proposal turn or response turn). When necessary, state augmentation is advised.

In our case, the environment (adversary) responds to trading proposals based on the history of the deterministic effects of the actions (trading proposals' manipulative effect on adversary's belief) up to that point. In other words, the same action (trade) may have different effects due to the deterministic effects on the environment (changes of the adversary's opponent model) of the actions that preceded it. There are successful combinations between these two different kinds of effects that the RLA has managed to identify and learn how to effectively use, originating from the multi-dimensions (manipulative dimensions) of the problem. It is therefore an interesting multi-dimensional non-stationary MDP case that we have shown to be solvable by RL, which suggests that trading proposals in dialogue evoke non-stationary beliefs in our everyday negotiations. We demonstrated that phenomenon with the realistic assumption that the adversary's opponent model is affected by all normal trading actions.

8.2.15 Discussion: Discourse Studies

Our results also bring an important argument of Van Dijk [20] to light, according to which there is an everyday conventional inference of dishonesty from manipulative acts. That negative effect cannot be taken for granted though as manipulation according to Dillard and Pfau [21], as well as O'Keefe [78] also occurs through legitimate persuasion. This is what our RL work suggests too. Hence we emphasize the significance of Attardo's perlocutionary cooperation [6] as before (Section 3.1.3).

8.3 Preferences

The experiments that follow in this section and the next one (Section 8.4) show that trading dialogues are more successful when the learning agent builds an opponent model – an estimate of the (hidden) goals and preferences of the adversary – and learns how to exploit them. We explore a variety of state space representations for the preferences of trading adversaries, including a representation based on the Conditional Preference Networks (CP-NETS) (in the next section 8.4) that have previously been discussed in Section 2.5. We will show that representing adversary preferences leads to significant improvements in trading success rates.

8.3.1 Actions (Trading Proposals)

Trade occurs through trading proposals that may lead to acceptance or rejection from the adversary, In an agent's proposal (turn) only one "give 1-for-1" or "give 1-for-2" trading proposal may occur, or nothing (41 actions in total) as we have seen previously:

1. I will do nothing
2. I will give you a wheat and I need a timber
3. I will give you a wheat and I need a rock
4. I will give you a wheat and I need a sheep
5. I will give you a wheat and I need a brick
6. I will give you a timber and I need a wheat

...

40. I will give you a brick and I need two rocks
41. I will give you a brick and I need two sheep

The adversary responds by either saying "Yes" or "No" to accept or reject the learning agent's proposals.

8.3.2 The State Encoding Mechanism

As we have seen in Section 7.2.5, to overcome issues related to long training times and high memory demands, we implemented a state encoding mechanism that automatically converts all of our trading game states to a significantly smaller number of states in a compressed representation. The new state representation consists of 10 slots (only the first 5 slots are used for the baseline LA which does not represent estimated adversary preferences) and takes into consideration the distance from goal and the availability of the resource, as well as its quality (goal or non-goal resource), and uses 7 different characters (A, G, Z, N, M, 1, and 2 described below) for the first 5 state slots. These represent the five resources that it currently has available. For example, in the case of building a the city, it needs 2 wheat and 3 rocks. That means that two out of five resources are goal resources and therefore they can be represented by G (goal) when their number is equal to the goal amount,

N (null) when their number is 0, M (more) when their number is more than the goal-quantity, and 1 or 2 when the distance from the goal quantity is 1 or 2 respectively. The 3 non-goal resources are represented by Z (zero) when the agent has 0 of them and A (available) when the agent has more than 0. For instance, the game state $\langle 1, 4, 3, 0, 2\rangle$ would be encoded to $\langle 1, A, G, Z, A\rangle$. The state space of the agent's resources therefore has 8 x 8 x 8 x 8 x 8 (=32,768) states that are encoded to only 4 x 2 x 5 x 2 x 2 (=160) states. This is reduced to 0.5% of the original size of the state space. With this method all of our experiments (apart from those with the CP-NETs that we will see later) require only a few minutes to run on a multi-core CPU with 128GB RAM.

8.3.3 State representing adversary's preferences

The next 5 state slots are used to refer to the adversary's preferences, based on the history of the adversary's acceptances and/or rejections of particular resources. Information regarding previous interactions has been taken into consideration in the past by agents such as the AutONA negotiation agent of Byde [13], who used a rule-based agent though, or those of Katz [58, 59], where RL is applied and three categorical databases (one general and two additional gender-oriented) are used to hold information about previous interactions in "Cliff Edge" environments (e.g. simultaneous auctions). Our 5 state slots are used only by the two RLAs which keep track of the adversarial preferences.

One of the two RLAs uses 2 different characters to represent whether or not the adversary wants a particular resource (i.e. accepts it), and the other RLA uses 3 characters because it also considers resources that the adversary does not want (i.e. it rejects that resource). Hence in these two cases we use 3 different characters, E (empty character) for a resource that we do not know if the adversary wants, X for a resource that we know that the adversary wants (it has accepted it before in the current game) and O for a resource that the adversary has rejected. This mechanism results in a series of 5 additional characters in the state representation for the two RLAs who estimate the adversary's goal. The RLA must figure out for itself the relationship between these 5 state variables and the first 5 (which represent the cards/resources that it holds). It therefore has to learn how these state variables relate to possible successful trading actions that it can take.

Based on the previous example, where the 5 first state slots represent the LA's current resources, an example state would be: $\langle 1, A, G, Z, A, X, E, O, E, X\rangle$. The 5 last characters of this example mean that the adversary has a preference for wheat, we don't know if it wants timber or sheep, it does not have a preference for rocks, and it also wants bricks. Note that a full representation of the *conditional preferences*

(using CP-NET) of the adversary (e.g. "I will give sheep for rock, but not for wheat") is used in Section 8.4.

8.3.4 The Adversary and the Preference RLAs

The challenging non-cooperative **Adversary** sets a random goal, which is always to either build a city, or a settlement, or a development card, or a road, at the beginning of each new trading game. Hence its preferences (and therefore its responses to the learning agent's trading proposals) will change according to the current goal. As before, we assume that the adversary has an infinite amount of resources to give.

We investigate several different learning agents playing against this Adversary:

- The Baseline Reinforcement Learning Agent (BRLA).
- The Positive Preferences RLA (PPRLA)
- The Negative-Positive Preferences RLA (NPRLA)

The Baseline RLA plays the game against the adversary who sets a random goal at the beginning of the new game. It does not keep track of the adversarial preferences and therefore it does not estimate the adversary's goal. Its state representation consists only of the resources that it currently has available, as we discussed in Section 8.3.2.

The PPRLA plays the game against the same adversary (as above) who sets a random goal at the beginning of the new game. It keeps track of the adversarial preferences and estimates the adversary's goal based on what resources the adversary has accepted in the trading dialogue so far. Its state representation consists of the resources that it currently has available plus the adversarial preferences, as we discussed in Sections 8.3.2 and 8.3.3. Those preferences though do not represent rejected resources. Hence this RLA infers the adversarial preferences based on the accepted trades only.

The NPRLA keeps track of the adversarial preferences as above but it represents rejected resources too. Hence this RLA infers the adversarial preferences based on the accepted and the rejected trades.

8.3.5 Experiments and Results

All agents are compared in respect of their win rates (or to rephrase this, trade success rate), which is the percentage of trading games in which they achieve their

goal (in this case, to get the resources required to build a city), within a sequence of 7 trading proposals (moves). A city is quite difficult to achieve, since it requires 3 rocks and 2 wheat. The y-axes of the graphs that follow represent the trade success rate (which we also refer to as "success rate", "reward-victory", or simply "win rate").

8.3.6 BRLA vs. Adversary: (Baseline)

The Baseline RLA (i.e. with no opponent modelling) played 250 thousand training games against the Adversary. This is our baseline case for comparison. The agent then played 20 thousand testing games and scored a 28.1% success rate (Figure 8.4, see Table 8.2).

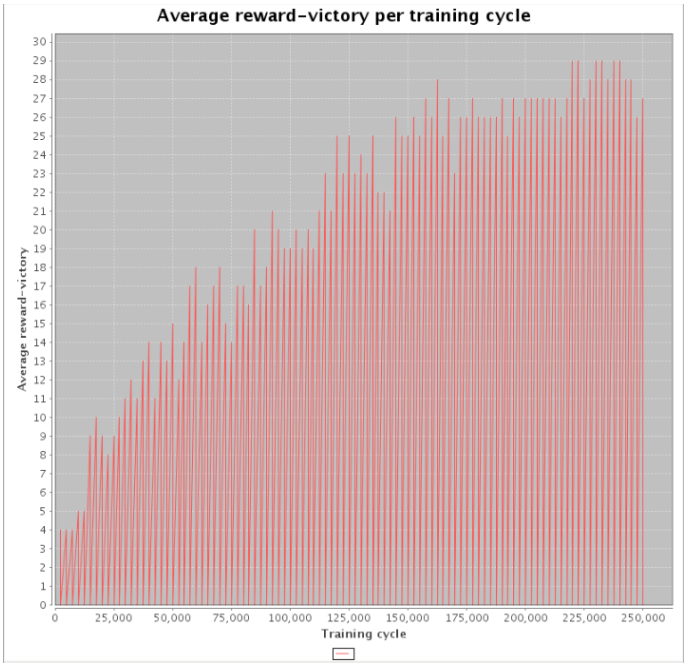


Figure 8.4: *Baseline Agent's reward-victory graph in 250 thousand training games.*

8.3.7 PPRLA vs. Adversary

Here we also trained the Positive Preferences RLA against the adversary over 250 thousand training games. The agent then played 20 thousand testing games and had a 44.2% success rate (Figure 8.5, see Table 8.2).

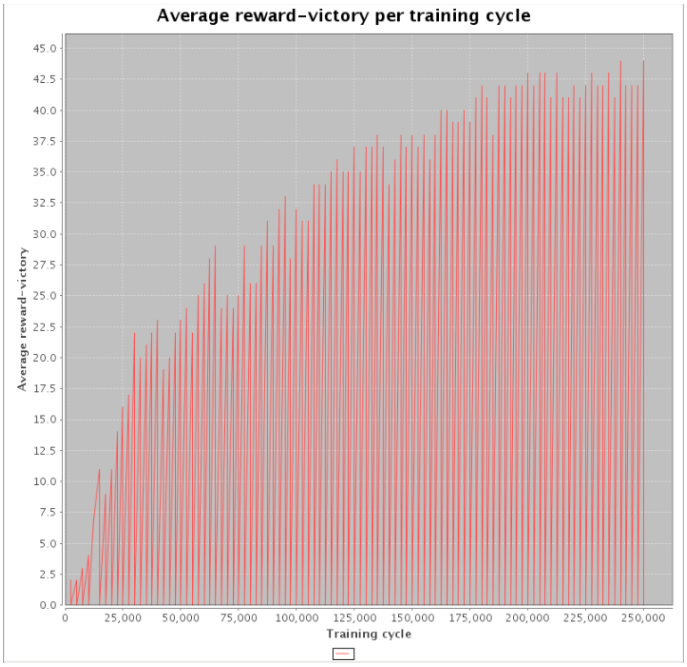


Figure 8.5: *PPRLA's reward-victory graph in 250 thousand training games.*

8.3.8 NPRLA vs. Adversary

Here we trained the Negative-Positive Preferences RLA against the adversary over 250 thousand training games. Over 20 thousand testing games the agent achieved a success rate of 52.5% (Figure 8.6, see Table 8.2).

8.3.9 Conclusion

In this part we showed that a RLA which keeps track of adversarial preferences outperforms one which does not by 24.4%. Furthermore, a RLA which takes into consideration the accepted and rejected trades in opponent modelling (i.e. positive and negative preferences) outperforms one which considers only the accepted trades by 8.3%. Note that these preferences are inferred during learning since the beginning of each new game and they are not given to the RLA a priori. Thus the RLA is learning how to estimate and exploit the preferences of its opponent in its trading behaviour within 7 trading proposals. In the next section we will extend this approach to use conditional preference information [14]. In this chapter we also presented a novel way of encoding the state space for RL of trading dialogues that reduces the state-space size to 0.5% of the original, and so reduces training times dramatically.

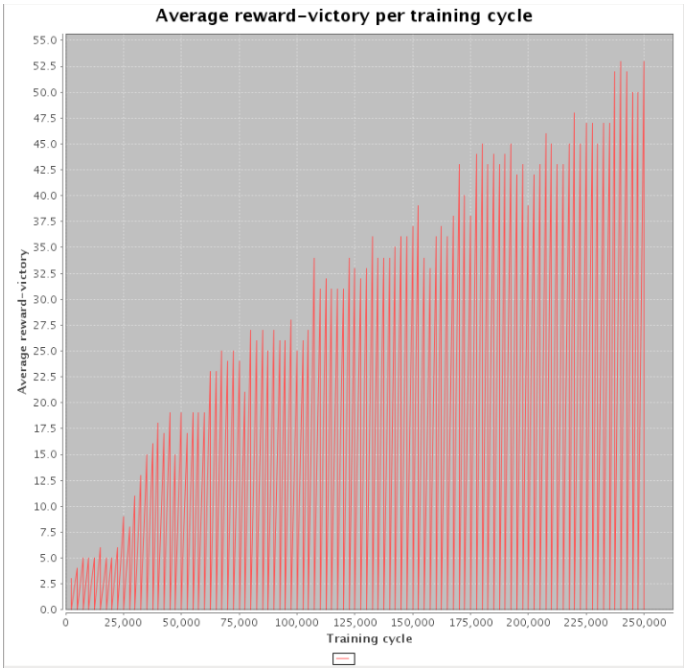


Figure 8.6: *NPRLA's reward-victory graph in 250 thousand training games.*

LA Name	Learning Agent opponent model	Success Rate (infinite)	Success Rate (finite)
BRLA	**No preferences considered**	**28.1**%	**15.4**%
PPRLA	Trade acceptance considered	44.2%*	19.1%*
NPRLA	Trade acceptance and rejection considered	52.5%*	22.5%*

Table 8.2: *Success rate of the Learning agent who considers the adversary's preferences. Adversary with infinite/finite resources. Performance (% wins) in 20K testing games, after training (*= significant improvement over baseline [BRLA cases]* ***in bold****,* $p < 0.05$*)*

8.3.10 Adversary with finite resources

So far we have assumed that the Adversary's resources are infinite, so the experiments were conducted again against the same Adversary which now has a finite random number of each resource (between 0 and 7, like the learning agents) at the beginning of each new game. We suspected that the task would be much harder for the agent to learn but we wanted to see exactly what the difference would be and approach real scenarios, because human players have a limited number of resources to trade. Hence there are cases where the opponent refuses to make a trade for two reasons: he/she does not need the resource or he/she does not *have* the resource to give. The ratios of the results are similar with the above, but the actual numbers are all much lower. In more detail, in 250k training games and after 20k testing games the performances of the agents were 15.4% for the BRLA (versus was 28.1% before), 19.1% for the PPRLA (versus 44.2% before) and 22.5% for the NPRLA (versus 52.5% before) – see Table 8.2.

8.4 CP-NETs

We now extend the previous experiments to the cases where the RLA uses a Conditional Preference Network – CP-NET [11] – to model the adversary's preferences. The Adversary and the actions remain the same. The state encoding mechanism is extended, to encode the CP-NET reasoning in an MDP state. As far as we know this is the first time CP-NETs have been used with RL [29]. We will show that representing adversary preferences with CP-NETs leads to over 24% improvement in win rate.

8.4.1 New extended state representation

A CP-NET expresses conditional preferences between events in a *ceteris paribus* paradigm (i.e. all else being equal) in a very informative, hierarchical manner which allows easy interpretation of complex preferential dialogue acts as we have discussed in the Background Chapter 2. We want to see whether RL is capable of inferring how to effectively use the CP-NET's functionality to increase the agent's trading performance. In our case we use it to keep track of the adversary's preferences, through the history of its accepted and rejected trades during a game. The history consists of a maximum of 7 RLA trading proposals and the corresponding adversary's responses as we have seen previously. The CP-NET resides in the RLA's state representation and consists of all those pairs that are formed by expressing the adversarial preference of one giveable resource over a receivable other one (there is a total of 5 resources). For example "I will give you wheat for sheep" expresses a preference for receiving sheep over giving wheat. Hence, the RLA has now a state representation which consists of 25 features, that is: 5 for the resources (which are encoded as we have seen before in Section 8.3.2) and 20 for the CP-NET adversarial preferences.

An example of such a feature (preference) might be $s \rightarrow r$, that is the adversary's preference for sheep over rocks. This would be inferred from the acceptance of the RLA's proposal: "I will give you sheep and I need rocks". Rephrasing this in CP-NET terms would be an acceptance to: "Given that I will give you sheep I need rocks", or from the adversary's perspective: "Given that I will receive sheep from you, I will give you rocks". So in other words we capture the adversarial conditional preferences between a giveable resource over a receivable one. We use again 3 different characters to represent the RLA's knowledge about the adversary's preference for each of the CP-NET's 20 possibilities, with E (empty preference) for a preference that we do not know if the adversary has, X for a preference that

we know that the adversary has, and O for a preference that the adversary has rejected. Thus the RLA must figure out again for itself the relationship between these 20 state variables and the first 5 (which represent the cards/resources that it holds). It therefore has to learn how these state variables relate to possible successful trading actions that it can take as we have examined before.

8.4.2 CPNET RLA vs. Adversary

Initially we trained the CP-NET RLA against the adversary (with infinite resources) over 250 thousand games. After 20 thousand testing games this RLA achieved a success rate of 36.1% (Figure 8.7 and Table 8.3). The performance is significantly better than that of the BRLA (28.1%, discussed in the previous Section 8.3), proving that the CP-NET improves the Reinforcement Learning procedure, but it is worse than that of the PPRLA (44.2%) and that of the NPRLA (52.5%) for the same number of training games. We presume that the reason was the significantly larger state representation that the CP-NET RLA uses, which consists of 25 slots rather than 10 for the PPRLA and NPRLA. It would probably require many more training games to achieve higher performances than those of its rivals (PPRLA and NPRLA). Hence we ran another experiment for 1.5 million training games (Figure 8.8), and in the following 20k testing games the CP-NET RLA's performance increased to 46.9%.

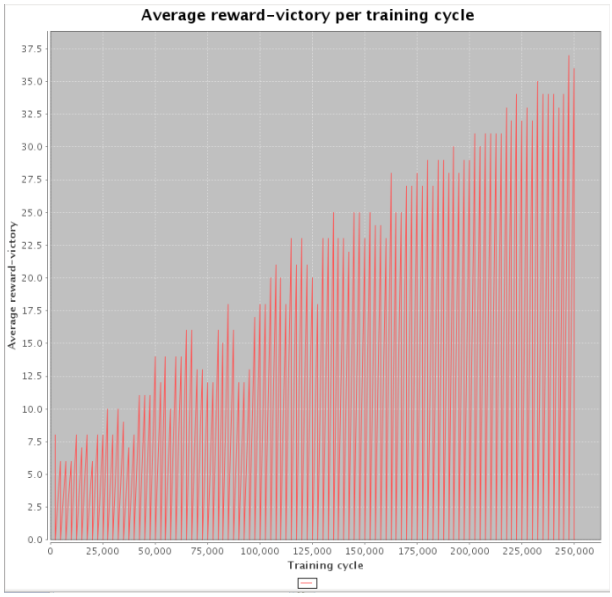


Figure 8.7: *CPNET RLA's reward-victory graph in 250 thousand training games.*

That result was better than that of the PPRLA in 250K testing games but not as high as that of the NPRLA in 250k testing games. Thus we ran another experiment for 2.5 million training games (Figure 8.9) to verify whether the improvement in

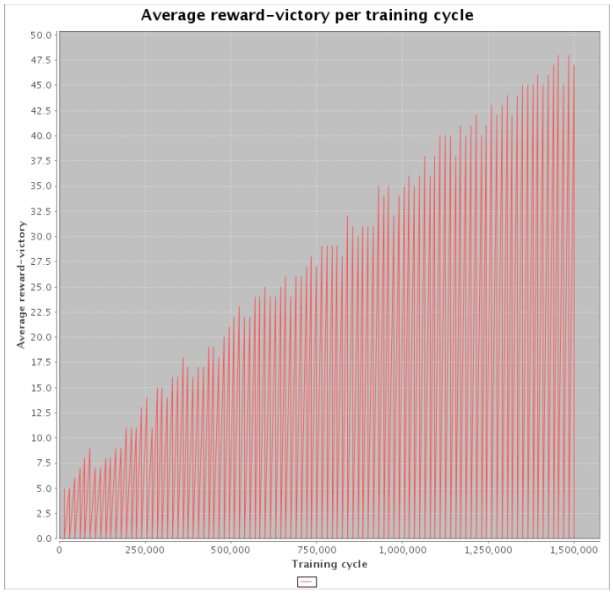


Figure 8.8: *CP-NET RLA's reward-victory graph in 1.5 million training games.*

performance would be as important as previously. In 20k testing games the performance of the CP-NET RLA was 50.3%. A final experiment (which required a very long running time) for 5 million training games (Figure 8.10) and 20k testing ones resulted in 52.4% performance. Even within this long training range the CP-NET RLA did not manage to perform better than the NPRLA which was trained for 5m games, resulting to a similar performance with that of the NPRLA trained for 250k games. See Table 8.4 for a summary of the results for longer training times.

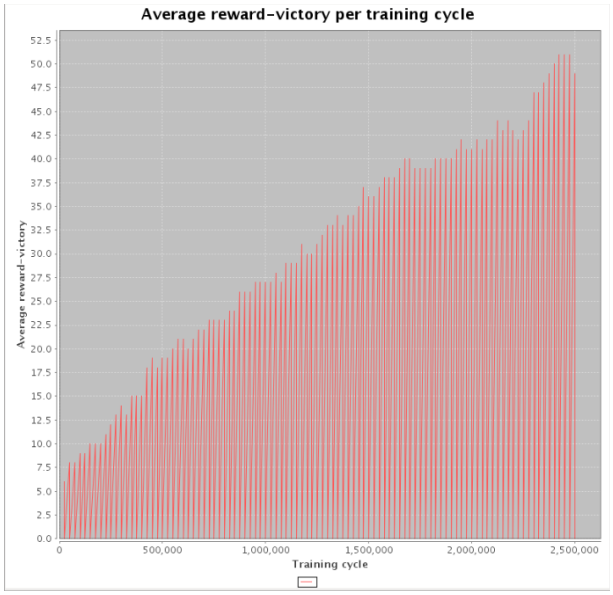


Figure 8.9: *CP-NET RLA's reward-victory graph in 2.5 million training games.*

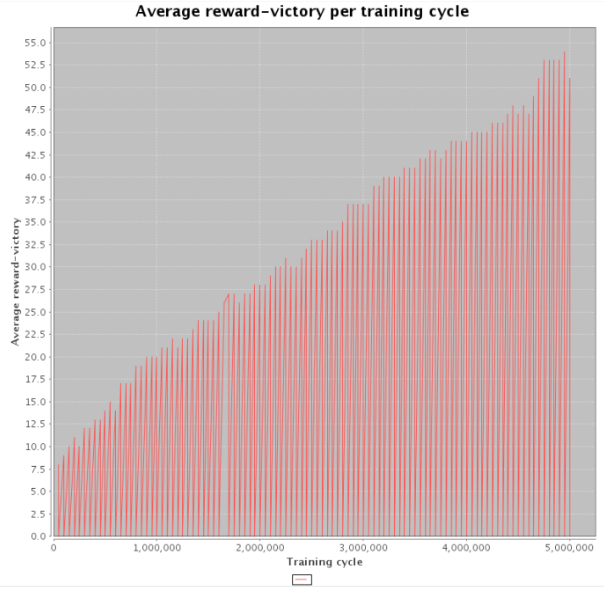


Figure 8.10: *CP-NET RLA's reward-victory graph in 5 million training games.*

Experiment	**Learning Agent policy**	**Adversary resources**	**Agent's wins**
BRLA (1)	**SARSA (250k games, baseline)**	∞	**28.1%**
CP-NET RLA	SARSA + CP-NET (250k games)	∞	36.1%*
BRLA (2)	**SARSA (250k games, baseline)**	**finite**	**15.4%**
CP-NET RLA	SARSA + CP-NET (250k games)	finite	18.1%*

Table 8.3: *Wins of the CP-NET Learning Agent. Performance (% wins) in 20 K testing games, after training. (*= significant improvement over baseline [BRLA cases] **in bold**, $p < 0.05$)*

Experiment	**Learning Agent policy**	**Adversary resources**	**Agent's wins**
CP-NET RLA	SARSA + CP-NET (1.5m games)	∞	46.9%*
CP-NET RLA	SARSA + CP-NET (2.5m games)	∞	50.3%*
CP-NET RLA	SARSA + CP-NET (5m games)	∞	52.4%*

Table 8.4: *Training for longer (infinite resources): Wins of the CP-NET Learning Agent. Performance (% wins) in 20 K testing games, after training. (*= significant improvement over baseline [BRLA (1) case with infinite adversarial resources, Table 8.3], $p < 0.05$)*

8.4.3 CP-NETS and finite resources

As we did in Section 8.3.10, we initially assumed that the Adversary's resources are infinite. The above experiment for 250k training games was conducted again against the same Adversary but now with a finite random number of resources at the beginning of each new game. As expected the result is now much lower than before. In 250k training games and after 20k testing games the CP-NET RLA scored a performance of 18.1%, compared to 36.1% against the Adversary with an infinite amount of resources (see Table 8.3). The performance was still better than that of

the baseline though with a difference of 2.7%.

8.5 Statistical significance

Similarly to Sections 4.4 and 5.5, we performed Z-tests[2] between the winning learning agent samples of our baseline cases (agent's wins) and that of each one of our other cases (Tables 8.1, 8.2, 8.3 and 8.4). By considering only the wins of the agent from the baseline cases and those of the other cases we performed Z-tests between their distributions to prove that they are statistical significant. Every result with an asterisk ($*$) on the above tables was significant at $p < 0.05$. Hence we can confidently say that these cases reject the null hypothesis[3].

In more detail, the total number of (testing) games that were taken into consideration in every case were 20,000. According to Table 8.1 the total number of wins from the Exp. 1 were 5,060, and from the Exp. 2 were 7,160. The LA's wins in Exp. 2 have higher score than those of the LA in the Exp. 1 ($z = 22.7954$, $p = 0$). The total number of wins from the Exp. 3 were 7,240. The LA's wins in Exp. 3 have higher score than those of the LA in the Exp. 1 ($z = 23.6208$, $p = 0$). According to Table 8.2, for the infinite cases, the total number of wins from the BRLA were 5,620, and from the PPRLA were 8,840. The PPRLA's wins have higher score than those of the BRLA ($z = 33.5113$, $p = 0$). The total number of wins from the NPRLA were 10,500. The NPRLA's wins have higher score than those of the BRLA ($z = 49.7451$, $p = 0$). According to Table 8.2, for the finite cases, the total number of wins from the BRLA were 3,080, and from the PPRLA were 3,820. The PPRLA's wins have higher score than those of the BRLA ($z = 9.7932$, $p = 0$). The total number of wins from the NPRLA were 4,500. The NPRLA's wins have higher score than those of the BRLA ($z = 18.1166$, $p = 0$).

According to Table 8.3, for the infinite cases, the total number of wins from the BRLA (1) were 5,620, and from the CP-NET RLA were 7,220. The CP-NET RLA's wins have higher score than those of the BRLA (1) ($z = 17.1357$, $p = 0$). For the finite cases, the total number of wins from the BRLA (2) were 3,080, and from the CP-NET RLA were 3,620. The CP-NET RLA's wins have higher score than those of the BRLA (2) ($z = 7.2304$, $p = 0$). According to Tables 8.3 and 8.4, the total number of wins from the BRLA (1) (Table 8.3) were 5,620, and from the CP-NET RLA for 1.5m games (Table 8.4) were 9,380. The CP-NET RLA's wins for 1.5m games have higher score than those of the BRLA (1) ($z = 38.8331$, $p = 0$). The total number of wins from the CP-NET RLA for 2.5m were 10,060. The CP-NET RLA's

[2]http://www.socscistatis tics.com/tests/ztest/Default2.aspx

[3]The null hypothesis states that the LA's wins from these experiments are not statistically significant compared to those from the corresponding baseline cases.

wins for 2.5m games have higher score than those of the BRLA (1) ($z = 45.4735$, $p = 0$). The total number of wins from the CP-NET RLA for 5m were 10,480. The The CP-NET RLA's wins for 5m games have higher score than those of the BRLA (1) ($z = 49.5512$, $p = 0$).

8.6 Conclusion

In this chapter we showed that implicit linguistic manipulation through deception or persuasion, based on all of the normal trading proposals, can be successfully learned via RL and significantly increases the performance of agents in a more complex trading environment than Taikun. We brought to light philosophical arguments by demonstrating through RL that manipulation is not (and should not) be only translated as deception, it can be performed through persuasion too, as our honest (persuasive) agent resulted in a similar performance with that of the dishonest one versus the same hindering adversary. At this point though, interesting future work might cover cases where there will be detection of multiple refusals to make a trade. These refusals may occur by the dishonest RLA after it has made a trading proposal which has been accepted by the adversary. We have seen that sometimes the dishonest agent does not really have the proposed resource to give, it lies when it offers it and therefore it refuses to trade it if its offer gets accepted. Detection in such cases could occur by agents similar to those of Sections 5.4 and 5.8.1. Also in our version of the game Catan, the agents learned how to successfully trade through larger state spaces than before, using many more actions that made the environment non-stationary due to their deterministic effect (i.e. that of manipulation). Furthermore we saw that RLAs which include the adversarial preferences (e.g. through CP-NETs) in their state representation based on the trading history can learn how to win more often than RL agents which don't represent that information. Finally we presented novel compression methods for representing the states in ways that reduces the size to 0.5% of the original, and therefore reduces the training times dramatically. In the next chapter our findings will be used in the more complex, multi-agent, full version of the game Catan.

Chapter 9

Evaluating learned trading dialogue policies in the JSettlers environment

The experiments of this chapter are all conducted using JSettlers [99], a research environment built using Java that captures the full multi-player (4-player) version of the game Catan (see Figure 9.1). The goal of the game is to collect 10 victory points, mostly by building different pieces on the board (discussed in the beginning of Chapter 7). The completion of each goal from the building plan[1], that is the construction of a piece (e.g. settlement), gives a particular number of victory points (e.g. city=2, settlement=1). JSettlers was another step, higher in complexity than that of our simpler generic version of the game that we examined in the previous chapter, where the players were two (instead of four[2]) and all of the agents were implemented by us. In addition, the goal is now to win the overall game (via trading and building), rather than succeed in each trading dialogue (i.e. a series of trading proposals and responses) only.

Here we had the opportunity to test the trained trading policies of the Negative-Positive RL agent (NPRLA) of the previous chapter, for both of the cases with infinite and finite adversarial resources, against other hand-crafted rule-based agents [40] (which we call "Bots") which use complex heuristics to trade and to build pieces on the board. Further modifications were also made to those agents in order to verify other previous findings, such as the successful use of linguistic manipulation through deception for instance. Despite the fact that our tested policies were trained in our simpler generic version of Catan, most of the results met our expectations as we will see. That proved that an explicit model of the full game is not required for

[1]The building plan indicates the next piece to build (e.g. city) and is used by the original STAC robot (Section 9.1.1).

[2]In this chapter we made the transition from bilateral to multilateral negotiations [67], therefore making our task more challenging in an attempt to generalise our learned policies.

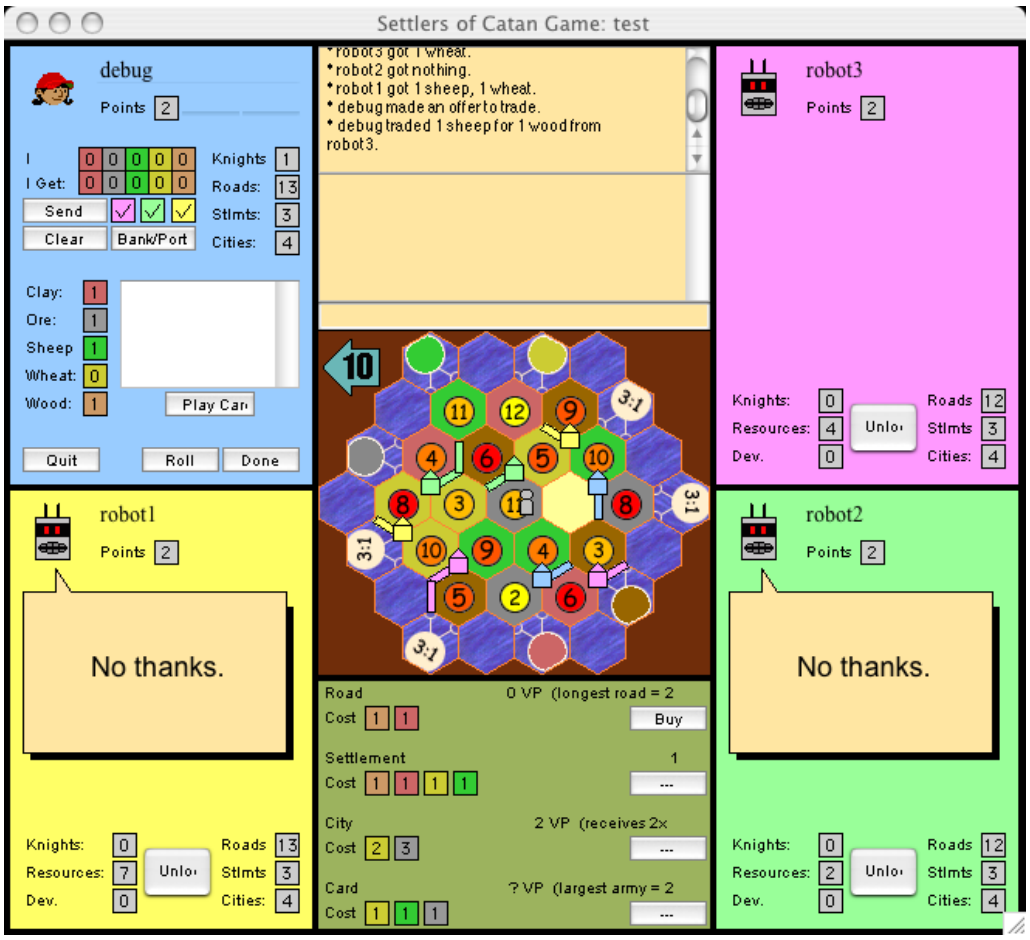


Figure 9.1: Example board of the game "Settlers of Catan" using the JSettlers interface [99].

successful RL trading policies to be learned. Ultimately, it suggested that a generic model of a complex trading scenario may be enough for effective RL, providing that efficient selection of the state representation and of the actions has been made.

9.1 Initial Experiments

In our first set of experiments we used a trained agent based on our four (for the development of city, road, development card and settlement) non-manipulative Reinforcement Learning policies of Section 8.1 against the Bots, discussed in the Section that follows. After that we used the agent against Bayesian agents [61] (also called "Bayes"). Each one of our agent's policies was trained for each of the four different goals (road, settlement, city, development card) of the game as we have discussed earlier.

9.1.1 The original STAC Robot ("Bot")

The Bot is based on the original expert rule-based agent of the research framework JSettlers [99] which is further modified to improve its winning performance. This agent (the Bot), which is the "benchmark" agent described in [40], uses complex heuristics to increase performance by following a dynamic building plan according to its current resource needs and the board's set-up. Hence its trading proposals, responses to offers, building plan and placement of constructions on the board along with other movements (such as the placement of the robber) are all affected based on those needs and set-up. This is a rule-based, hand-crafted agent, developed by experts in the Catan game. It is therefore interesting to find out how well an agent trained in a simulation environment can perform against it.

9.1.2 Our trained RL agent (goal-oriented)

Our trained RL agent is in fact an original STAC Robot (Bot) which has been modified to make offers based on one of our four learnt policies of Section 8.1. That means it has already learnt how to make trading proposals towards a particular goal in order to reach it as soon as possible (goal-oriented). For example, if the goal is to build a city, that is to gather 2 wheat and 3 rocks, and the agent has only 1 wheat and 1 sheep then it has learned that it should offer the sheep for either wheat or rock(s). It has also learnt through those policies to mainly use trading proposals that ask for two resources rather than one. However it has not learned how to respond to opponents' offers, as modifying the Bot takes a long time and it was not necessary for our experiments. Hence in the current experiment our trained RL agent (which is a modified Bot) responds in exactly the same way as the Bots. In more detail, whenever it proposes trades then it follows our corresponding policy depending on the current goal, that is a part of its building plan. In other words, apart from the trading proposals, everything else such as the responses, the building plan, the placement of constructions on the board and other movements are those of the original STAC Robot that are followed.

9.1.3 Trained RL agent vs. 3 Bots

Initially we evaluated our trained RL agent versus three identical hand-crafted agents, the Bots that we mentioned above. Our trained RL agent played 10 thousand

games versus the Bots and resulted in a performance of 32.66% (Table 9.1), while those of the Bots were 22.9%, 22.66% and 21.78% respectively. The results show that our trained RL policies by exploiting the power of "give 1 for 2" trading actions and by being exclusively goal-oriented perform better than the best Bot by nearly 10%. It also suggests that if our trained agent had learnt how to respond towards a goal too, i.e. by additionally taking the advantage of proposals that offer two resources in exchange of one, then its performance would be even better. This result was interesting because it proved that our generic 2-player version of the game (Chapter 8) was enough to train a successful policy for the multi-player version of the game, by considering all three opponents as one. Hence our RLA proposed only public trades (e.g. "I give 1 timber and I need 2 sheep, anyone?"), without referring to someone in particular. Furthermore the 32.66% performance of our RL agent was around 7% better than that of [61] (discussed in Section 2.4.2), who trained it in the real multilateral negotiations environment (i.e. full Catan game in JSettlers) and resulted in a performance a bit higher than 25% against the same opponents (i.e. Bots). This result provided evidence that it was much more effective to train a RLA in a bilateral negotiations environment and then test it on the multilateral one, as noise from the full game probably made RL a harder task there.

9.1.4 The Bayesian agent ("Bayes")

This agent [61] is a Bot whose trading proposals are modified based on the human corpus that was collected from Afantenos et al. [1]. As we have seen before, the trading proposals, responses to offers, building plan and placement of constructions on the board along with other movements are those of the Bot. However the trading proposals are based on Bayesian sampling of 60 human games which were played during a tournament that was arranged, as part of the STAC research. In more detail, the Bayesian agent was 65.7% accurate with human moves (F-Measure score = 0.656). The human players though were not expert "Catan" players.

9.1.5 Trained RL agent vs. 3 Bayes

In this experiment we used our trained RLA (Section 8.1) versus three of the above Bayesian agents. After playing 10 thousand games, our trained agent scored a performance of 36.32% (Table 9.1), which is much higher than those of the three Bayes agents. Their performances were 21.43%, 21.02% and 21.23% respectively. The difference of our agent's performance, which is nearly 15% higher than the best of the Bayes agents, suggests that the human players' trading skill is lacking com-

pared to the trained RL policies. They have probably not taken the full advantage of trades which ask for more than they offer and they were not as focused on goal as our trained RL agent. That could be explained by considering the beginner status of the human players, which is also reported in the Afantenos et al. work [1]. By introducing trained RL policies which have learnt how to respond to offers, we expect the result to become much higher, as the current responses of our trained RL agent are those of the Bot as we have mentioned. The Bayes agents' responses though are also those of the Bot.

9.2 Manipulation

In this part we evaluated our trained manipulative Reinforcement Learning policies[3] from Section 8.2.9 (dishonest) against the Bots and the Bayes agents. The Bots were modified in such a way that every time the manipulative policy makes a trading proposal then the Bots are affected (manipulated). That means the weights of the resources that they offer and ask for change according to the trained manipulative RL proposals, in the same fashion as we have previously examined in Section 8.2.3. We modified only the adversarial trading proposals to be affected by the policy's manipulation and not the responses, as modifying the Bot takes a long time and it was not necessary for our experiments. On the other hand the Bayesian agents (Bayes) remained unmodified. The purpose was to verify whether they would be affected on their own by the trained manipulative policies or not, as we were suspecting that human players might be affected and the same might happen then to the Bayes agents too.

9.2.1 Trained Dishonest RL agent vs. 3 Manipulated Bots

The policy of the trained Dishonest RL agent (Section 8.2.13) played 10 thousand games against 3 Bots which were affected by the agent's trading proposals (in exactly the same way as discussed in Section 8.2.3). At the beginning of the games the probabilities (weights) that represent the Bot's willingness to give (i.e. through trading proposals) each of the five resource types start equally (each one is 20%).

[3] A policy for the city (2 wheat and 3 rocks) only was trained for the experiments of Section 8.2.9, as we have discussed. For the experiments of this chapter, we also trained three more dishonest policies in our 2-player generic version of the game, for settlement, road and development card, as the full version of the game in JSettlers requires all of them. The results were similar to those of the city and therefore they have not been reported again.

Whenever the RLA makes a trading proposal the algorithm first considers the giveable resource. The Bot's probability of giving the LA's *offered* resource increases by 4% (therefore the Bot's trading proposals which have this resource as giveable become slightly more likely to be selected by the Bot) and those which give the other resources decrease accordingly (i.e. equally, each one decreases by 1%). Then the algorithm takes into consideration the receivable resource(s) of the RLA's trading proposal. The probability for the Bot to give that particular resource (through its trading proposals) is reduced by either 4% or 6% (if it is a "give 1-for-1" or "give 1-for-2" RLA's trade proposal respectively), and the probabilities of giving the four other resource types increase accordingly (i.e. equally, each increases by either 1% or 1.5% respectively). In the current experiment, the Bot's receivable resources of its trading proposals are not taken into consideration when it comes to propose a trade. It chooses only based on the giveable resources. Hence the Bot will probably choose one of the trading proposals at random whose giveable resource's weight is the highest. Furthermore, the responses of the manipulated Bots are not affected. That means they are exactly the same as those of Section 9.1.3, which are those of the Bots. The results were: the 3 Bots won by 21.44%, 20.79% and 21.42% respectively. Our trained Dishonest RL agent won by 36.35% (Table 9.1), having a positive difference of nearly 15% from the best Bot, showing that manipulation is very effective towards the manipulated Bots of the JSettlers environment (i.e. full multi-player version of Catan).

9.2.2 Trained Dishonest RL agent vs. 3 Manipulated Bots (weights based on building plan)

The policy of the trained Dishonest RL agent played 10 thousand games against 3 Bots. The probabilities (weights) that represent the Bot's willingness to give (i.e. through trading proposals) each of the five resource types start equally (each one is 20%), as we mentioned in the previous section too, but they are then adjusted further according to the building plan (BP) in this case. That means the Bots are initially biased towards specific resources, as the BP indicates the next piece to build (e.g. city). These resources depend on the BP and every time the BP changes then the weights change accordingly too. An example would be: let's assume that the Bot was planning on building a city (that's what the BP indicates). The goal resources of a city are 2 wheat and 3 rocks. In that case the manipulated Bot will start the game with all of the weights that represent the Bot's willingness to give (i.e. through trading proposals) each of the five resource types equally (each one is 20%) but those which give wheat and rocks will be further slightly decreased (by

6%[4] in the case of giving wheat and 7% in the case of giving rocks). All of the rest[5] will be further slightly increased accordingly (i.e. equally - for those four which are affected by the wheat each one will increase by 1.5%, and then for those four which are affected by the rocks each one will increase by 1.75%). That makes the job of the Dishonest RL agent's policy even harder than before. The results of this experiment were still satisfying: the 3 Bots won by 22.53%, 21.47% and 21.8% respectively. Our trained Dishonest RL agent won by 34.2% (Table 9.1), having a difference of nearly 12% from the best manipulated Bot of this case, which were clearly a bit harder now.

9.2.3 Trained Dishonest RL agent vs. 3 Manipulated Bots (weights based on building plan and resource quantity)

In this experiment, the policy of the trained Dishonest RL agent played again 10 thousand games against 3 Bots. This case is identical to the above (Section 9.2.2) but the weights (probabilities) are additionally adjusted according to the goal resource quantity. For example, we saw previously that in the case of a city 2 wheat and 3 rocks are needed. These two goal resources adjust the weights in a way that the Bot initially offers them less often. What happens now is that the Bot also checks to see whether or not the current goal resource quantities are more than the goal quantities. If yes, then it slightly increases (e.g. by 4% if it has one more than needed, by 6% if it has two and by 7% if it has three or more) the probability that represents the Bot's willingness to give the particular resource whose quantity is more than the goal one, in order to offer it more often, because it has more than it needs. On the other hand, the weights of offering the other resources decrease accordingly (i.e. equally - as described in the previous section). If the current goal resource quantities are not more than the goal quantities, then nothing else happens.

The results of this experiment for the trained Dishonest RL policy were as good as the above: the 3 Bots won by 21.72%, 21.5% and 22.47% respectively. Our trained Dishonest RL agent won by 34.33% (Table 9.1), having a positive difference of nearly 12% again from the best manipulated Bot. This result, along with the two above ones (Sections 9.2.1 and 9.2.2), suggested that the RL agent's dishonest manipulative policy was very effective against the Bots of the multi-player version

[4] 1 needed goal resource is represented by 4%, 2 resources by 6% and 3 resources are represented by 7%. As before, all these small % changes reflect that the adversary is not *dramatically* influenced by manipulation. If the percentages were much larger (e.g. 30%) then the results presented below would be more extreme.

[5] The rest of the resources will be: timber, rocks, sheep and bricks which are affected by the wheat, and then wheat, timber, sheep and bricks which are affected by the rocks.

of the game, showing that our transition from a bilateral negotiation environment to a multilateral one was effective. Furthermore, by comparing the results of the trained dishonest agent against the manipulated Bots (Sections 9.2.1, 9.2.2 and 9.2.3) with those of the trained non-manipulative agent against the non-manipulated Bots (Section 9.1.3), we show that linguistic manipulation is still successful even versus multiple opponents, as the positive differences are in the range of 1.5%-3.5%.

9.2.4 Trained Dishonest RL agent vs. 3 Bayes

The policy of the trained Dishonest RL agent played 10 thousand games against the 3 Bayes agents that we examined in Section 9.1.4. We suspected (and that was our hypothesis) that the human players[6] might have been affected by their opponents' manipulation (if any) in their games[7], and therefore we wanted to verify that by using our Dishonest policy. The results of this experiment proved our hypothesis: the 3 Bayes agents won by 21.97%, 20.58% and 21.64% respectively. Our trained Dishonest RL agent won by 35.81% (Table 9.1), having a positive difference of nearly 14% from the best Bayes agent. That result was an evidence of the fact that the Bayes agents were indeed affected by manipulation (and now by the Dishonest RL agent's manipulative policy too) and its success resulted to almost 14% more winning games. This difference is similar (nearly 15%) to that of the Section 9.1.5. The policy there does not use any manipulation at all though and aims for the goal as soon as possible. Despite the fact that the policy of this section uses manipulation, through repetition or false use (lies) of the trading proposals for example, and does not exclusively aim for reaching the goal as soon as possible, like the non-manipulative policy there does, it is still nearly as successful. This is interesting because the manipulative policy of the dishonest RLA is "wasting" time (rounds) for the sake of manipulation. That verifies that the manipulation is effective on the Bayes agents, showing that it might be successful on human players too.

[6]Their behaviour is simulated by the Bayes agent.

[7]As we have discussed in Section 9.1.4, these games were played during a tournament that was arranged as part of the STAC research.

Exp.	**LA's policy**	**Adversaries**	**LA's wins**
Baseline 1	**Bot**	**3 Bots**	**25% (approx.)**
Baseline 2	**Bayes**	**3 Bayes**	**25% (approx.)**
Section 9.1.3	Bot + SARSA (No Manip.)	3 Bots	32.66%*
Section 9.1.5	Bot + SARSA (No Manip.)	3 Bayes	36.32%*
Section 9.2.1	Bot + SARSA (Dishonest)	3 Manip. Bots	36.35%*
Section 9.2.2	Bot + SARSA (Dishonest)	3 Manip. Bots (BP)	34.2%*
Section 9.2.3	Bot + SARSA (Dishonest)	3 Manip. Bots (BP & resource quan.)	34.33%*
Section 9.2.4	Bot + SARSA (Dishonest)	3 Bayes	35.81%*

Table 9.1: *Wins of our RL trained policies in the JSettlers environment. The baseline performance is 25% for both Baseline 1 and 2 which are* ***in bold text****. The performances (% wins) above are after 10K games (*= significant improvement over baseline, which is 25%, $p < 0.05$). The above cases which involve Bots as adversaries are compared to Baseline 1 and those which involve Bayes as adversaries are compared to Baseline 2.*

9.3 Statistical significance

Similarly to previous chapters, we performed Z-tests[8]. By considering only the wins from the baseline cases (Baseline 1 and 2) and those of the other cases we performed Z-tests between their distributions to prove that they are statistically significant (Table 9.1). Every result with an asterisk ($*$) on the above table was significant at $p < 0.05$. In general, the above cases which involve Bots as adversaries are compared to Baseline 1 and those which involve Bayes as adversaries are compared to Baseline 2. Furthermore, regarding the cases with manipulation, the results of Sections 9.2.1, 9.2.2 and 9.2.3 are significant compared to those of the baseline case of Section 9.1.3, and show that manipulation gives an advantage (Table 9.1). However the results of Section 9.2.4 compared to those of the baseline case of Section 9.1.5 are not significant (Table 9.1). Hence we can confidently say that all of the other cases reject the null hypothesis[9].

In more detail, the total number of (testing) games that were taken into consideration in every case were 10,000. According to Table 9.1 the total number of wins from the Baseline 1 were 2,500, and from the Experiment of Section 9.1.3 were 3,266. The LA's wins in Experiment of Section 9.1.3 have higher score than those of the Baseline 1 ($z = 11.9576$, $p = 0$). The total number of wins from the Experiment of Section 9.2.1 were 3,635. The LA's wins in Experiment of Section 9.2.1 have higher score than those of the Baseline 1 ($z = 17.4038$, $p = 0$). The total number of wins from the Experiment of Section 9.2.2 were 3,420. The LA's wins in Experiment of Section 9.2.2 have higher score than those of the Baseline 1 ($z = 14.2508$, $p = 0$). The total number of wins from the Experiment of Section

[8]by using a web tool (http://www.socscistatistics.com/tests/ztest/Default2.aspx)

[9]The null hypothesis states that the LA's wins from the above experiments are not statistically significant compared to those from the corresponding baseline cases.

9.2.3 were 3,433. The LA's wins in Experiment of Section 9.2.3 have higher score than those of the Baseline 1 ($z = 14.443$, $p = 0$). The total number of wins from the Baseline 2 were 2,500, and from the Experiment of Section 9.1.5 were 3,632. The LA's wins in Experiment of Section 9.1.5 have higher score than those of the Baseline 1 ($z = 17.3601$, $p = 0$). The total number of wins from the Experiment of Section 9.2.4 were 3,581. The LA's wins in Experiment of Section 9.2.4 have higher score than those of the Baseline 2 ($z = 16.6169$, $p = 0$).

The total number of wins from the Experiment of Section 9.1.3 (baseline) were 3,266, and from the Experiment of Section 9.2.1 were 3,635. The LA's wins in Experiment of Section 9.2.1 have higher score than those of the Section 9.1.3 (baseline) ($z = 5.4887$, $p = 0$). The total number of wins from the Experiment of Section 9.2.2 were 3,420. The LA's wins in Experiment of Section 9.2.2 have higher score than those of the Section 9.1.3 (baseline) ($z = 2.3083$, $p = 0.02088$). The total number of wins from the Experiment of Section 9.2.3 were 3,433. The LA's wins in Experiment of Section 9.2.3 have higher score than those of the Section 9.1.3 (baseline) ($z = 2.502$, $p = 0.01242$). The total number of wins from the Experiment of Section 9.1.5 (baseline) were 3,632, and from the Experiment of Section 9.2.4 were 3,581. The LA's wins in Experiment of Section 9.2.4 do not have higher score than those of the Section 9.1.5 (baseline) ($z = -0.751$, $p = 0.45326$).

9.4 Preferences

Here we had the opportunity to test the trained trading policies of the Negative-Positive RL agent (NPRLA, Section 8.3.4), for both of the cases with infinite and finite adversarial resources, against the Bots (discussed in Section 9.1.1) [40]. We chose to evaluate the NPRLA policies in JSettlers because it was our best-performing trading agent which considers adversarial preferences. The NPRLA agent's policies were trained for each of the four different goals (road, settlement, city, development card) of the game, for both of the cases with infinite and finite adversarial resources, as we discussed earlier.

9.4.1 Our trained NPRLA Settlers agents

Both of our trained NPRLA agents[10] are in fact a JSettlers Bot modified so as to make offers based on the NPRLA learned policies, instead of using those of the Bot. Responses and all of the other moves (such as positions on the board) are those of the Bot though and the Bot also computes a building plan which indicates which piece (e.g. city) to build next as we have discussed. The NPRLA maintains a history of all the opponents' (Bots) resource preferences during the game, therefore treating them all as one adversary[11], using the representation that we discussed in Sections 8.3.2 and 8.3.3, based on the accepted and rejected trades that occurred in the past. Whenever the NPRLA is about to propose a trade, it matches the game's trading history and its own current resources with a particular state of one of its learned policies (according to the current goal, e.g. city) and retrieves the dialogue action to take. This dialogue action is a trading proposal that aims for the current goal, that is a part of the Bot's building plan. We investigate whether RL preferential policies trained in a generic bilateral negotiations environment would increase the Bot's performance in JSettlers, which is a multilateral negotiations environment and captures all the game-play details of the board game Catan.

9.4.2 Trained NPRLA (infinite resources) vs. 3 Bots

In this experiment we used our NPRLA agent which was previously trained for one adversary with infinite resources (Section 8.3.8), versus three identical hand-crafted agents (Bots). Our trained agent played 50 thousand games versus the 3 Bots and resulted in a win rate of 25.19% (Table 9.2), while those of the 3 Bots were 24.96%, 25.24% and 24.61% respectively. In 10 thousand games the performances were 24.98% for the NPRLA, and 24.8%, 24.4% and 25.82% for the 3 Bots. The similarity of the results suggests that all of the agents know how to trade towards a particular goal. It also shows that the preferences that were collected from all of the opponents in this case, and were matched to those of our trained policies, offered no important advantage.

[10]These agents were trained against the same adversary (discussed in Section 8.3.4), which had either infinite or finite available resources, for all the four different goals of the game. These goals are to build a city, a settlement, a road, or a development card and they require a different number and types of resources to be gathered. Hence we had 4 trained NPRLA policies to use for each of the two agents.

[11]It was important for us to see here whether the learned preference policies from one opponent would be successful in a multi-opponent environment, as it would hopefully hold important information such as the overall resource values (due to the board's initial resource placement) for the particular game.

9.4.3 Trained NPRLA (finite resources) vs. 3 Bots

The NPRLA agent of this experiment was previously trained against the adversary which had finite resources (Section 8.3.10). It was tested versus three of the hand-crafted agents (Bots). In 10 thousand games the results were 25.11% (Table 9.2) for the NPRLA, and 24.98%, 25.15% and 24.79% for the 3 Bots. Again we see that the trained agent has a very similar win rate, in terms of the overall game, with the 3 Bots, even though it was not trained specifically to beat them. However, the trained NPRLA in this case is not as successful as the trained agents in table 9.1. The reason was that its previously learned policies in the bilateral negotiations environment did not hold accurate enough preference information in regard to the preferences of the JSettlers Bots.

Exp.	**LA's policy**	**Adversaries**	**LA's wins**
	Bot	3 Bots	25% (appr.)
Sect. 9.4.2	Bot + SARSA(NPRLA) (trained vs. infinite res.)	3 Bots (50k games)	25.19%
Sect. 9.4.2	Bot + SARSA(NPRLA) (trained vs. infinite res.)	3 Bots (10k games)	24.98%
Sect. 9.4.3	Bot + SARSA(NPRLA) (trained vs. finite res.)	3 Bots (10k games)	25.11%

Table 9.2: *Wins of our NPRLA trained policies in the JSettlers environment. The baseline performance is 25%.*

9.4.4 Related work

RL applied to strategic negotiations includes the work of [80], who proposed hierarchical RL for automatic decision making on object-placing and selecting trade actions in Catan. This work uses built-in knowledge for learning the behaviours of the game quicker, and suggests that the combination of learned and built-in knowledge is able to beat human players. More recently, as we have discussed in Section 2.4.2, [61] implemented an MDP model for selecting trading proposals, trained and tested in the JSettlers environment (where 4 players play the full version of Catan) but this work did not take into account any preference modelling or manipulation.

9.5 Conclusion

We found that our trained RL trading policies performed as well as (i.e. the case of preference agents) or even better (i.e. the cases of goal-oriented and manipulative agents) than the hand-crafted agents (Bots) in JSettlers, suggesting that data-driven policy training can result in very competitive trading strategies, without expert hand-crafting. Furthermore we showed that an explicit multilateral model of the game "Catan" is not required for successful RL trading policies to be learned. Indeed, our RLA (Section 9.1.2) which was trained in our bilateral negotiations environment was around 7% better (Section 9.1.3) than that which was trained in the real tested multilateral negotiations environment (i.e. full Catan game in JSettlers), according to the results of [61]. To our knowledge this is the first time that policies trained in bilateral negotiations performed better in multilateral negotiations than those which were trained on them, by considering all opponents as one. Ultimately, it is suggested that bilateral training environments (e.g. our generic version of Catan in Chapter 8) may suffice for complex multilateral non-cooperative trading scenarios (e.g. full version of the game in JSettlers) for effective RL, providing that efficient selection of the state representation and of the actions has been made. In this chapter our work showed that RL with (or without) preference learning, or with manipulation, trained versus a single opponent in a generic two-player version of a non-cooperative trading game (i.e. Catan), is also effective versus many opponents, treating them all as one, in the more complex full version of the game. In the next chapter we will evaluate some of our learned RL trading policies against human players in trading scenarios of the game Catan.

Chapter 10

Evaluating learned trading dialogue policies with humans in Catan

In this chapter we will discuss the experiments that we conducted with human players in trading scenarios of the game Catan. Having the knowledge from human trials of Chapter 6, we were now planning on restricting the luck factor that "obscured" the agents' skill, play more games than before and focus on motivating the players to trade. In general, the goals that we set were: i) to investigate how successful our trained RL policies were against humans, ii) to verify whether the manipulative policy is better than the goal-oriented (non-manipulative) policy or not, and iii) to investigate further if the current manipulative agent[1] is effective versus humans and how. In detail, the humans played a game that we implemented and focuses only on the trading dialogue part (as this is where the main interest of this research is on) of Catan against our two trained policies from Sections 8.1 (non-manipulative goal-oriented RLA policy) and 8.2.5 (manipulative dishonest RLA policy).

We compare these two learned trading dialogue policies because they were the best performing policies that we had. Furthermore, we wanted to investigate the effect of implicit manipulation, which is used by the dishonest agent in the form of deception, on humans. The honest agent which uses persuasion would be interesting to be evaluated too, but the deception (as well as its detection) are of higher priority (due to ethical concerns, discussed in Section 1.2) and therefore we decided to use the dishonest agent against humans. A session consisted of 2 games during our pilot experiments and 4 games during our main experiments. Each game consisted of 14

[1]In Chapter 6 we discussed experiments with human players in Taikun. The manipulative agent there used explicit manipulation (e.g. "I really need sheep"). In this chapter the manipulative agent uses implicit manipulation (i.e. through normal trading proposals) and it was important for us to see how humans would react to that.

rounds during the pilot experiments and 20 rounds during the main experiments. A round includes the agent's trading proposal and the human player's response. More details about the mechanisms of the game are given below.

10.1 Pilot sessions

In order to play only the trading dialogue part of Catan we developed a game in Java which is called "Trading in Catan" (Figure 10.1). In this game the players only trade the five resources of Catan, which are wheat, timber, rocks, sheep and bricks. We implemented this game to evaluate the agents' trading abilities. The human only responds to the trading proposals of each agent (which tries to build a city). They both started with 1 of each resource during the pilot sessions. The player did not know what resources the agent currently had and what its goal was. During our pilot sessions, each of the experiments included two sequential games between a human player and the non-manipulative agent, and two sequential games between the same human player and the manipulative agent. The human players did not know with which agent they played against each time. Also two of them played with the manipulative agent first and two played with the non-manipulative one first. When the application begins, the human was informed about the rules of the game that we will examine in depth below.

10.1.1 Information given to the player

The human plays two games against each trained agent. She responds to its trading proposals in order to exchange resources. The goal is to collect more score points than the agent. The player collects a particular number of score points every time she builds something (Figure 10.2). To build something she needs to collect a specific number of resources. Information regarding the constructions to build along with the required resources are given to her[2]. The agent's resources are unknown to the human and vice versa. It tries to get a number of resources to build something. The agent may eventually build through the player's trades or trades that it may do "secretly with others" (i.e. each of its resources randomly change in the range of -2 to +2 at the end of each round). The human only interacts (i.e. trade resources) with the agent. When the agent builds something the game ends and then the next

[2]Road requires 1 timber and 1 brick and gives 20 points. Card requires 1 wheat, 1 rock and 1 sheep and gives 30 points. Settlement requires 1 wheat, 1 timber, 1 sheep and 1 brick and gives 40 points. City requires 2 wheat and 3 rocks and gives 100 points.

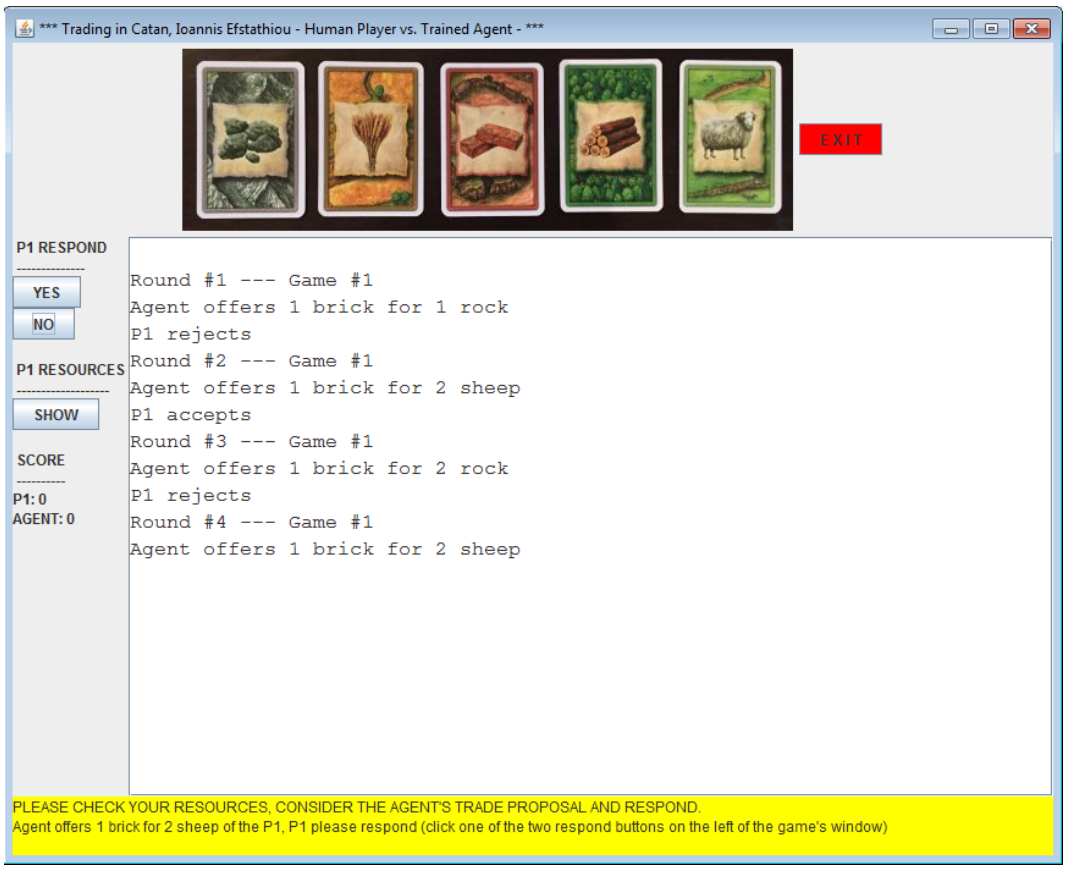


Figure 10.1: The trading phase of the game "Trading in Catan". We notice that the manipulative agent in this example lies (i.e. asks for sheep.) in order to manipulate the human player.

one begins. The game also ends when 14 rounds have been played and the agent did not manage to build anything. The player can build constructions at the end of each game. She can keep her resources and build everything at the end of the second game if she likes. In the beginning of the first game the player and the agent both start with one of each resource. In the beginning of the second game, they both gain 1 of each resource.

10.1.2 The score

The agent tries to build a city only. Every time it manages to do that it scores 100 points. The human on the other hand can build all 4 different constructions (road, city, settlement, development card) and gains a corresponding amount of score points every time she does that. In the case of the road the human gets 20 points, 30 points for a card, 40 points for a settlement and 100 for a city (same for the agent). The score for a city was previously 50 but we changed it to 100 to promote trading and "penalise" the players for not trading. The problem was that

the players were rejecting everything and were still able to build roads and cards and gather 100 points total with the resources they had (at the end of the second game they had 2 of each resource which were given to them by the two games). The agent, only with random changes on its resources and no trades, sometimes would get 50 points (if lucky) and it would lose with no trades ever happening. As we will discuss later, the change of the city's points to 100 was not capable of solving the problem though as it introduced others. The human can build a construction only after the end of each game. The game ends when either the agent manages to build a city or 14 trading proposals have been made and the agent did not manage to build. The score points that have been gained in the first game are transferred to the second one. Whoever gathers more total score points at the end of the second game (which includes the score points of the first) wins.

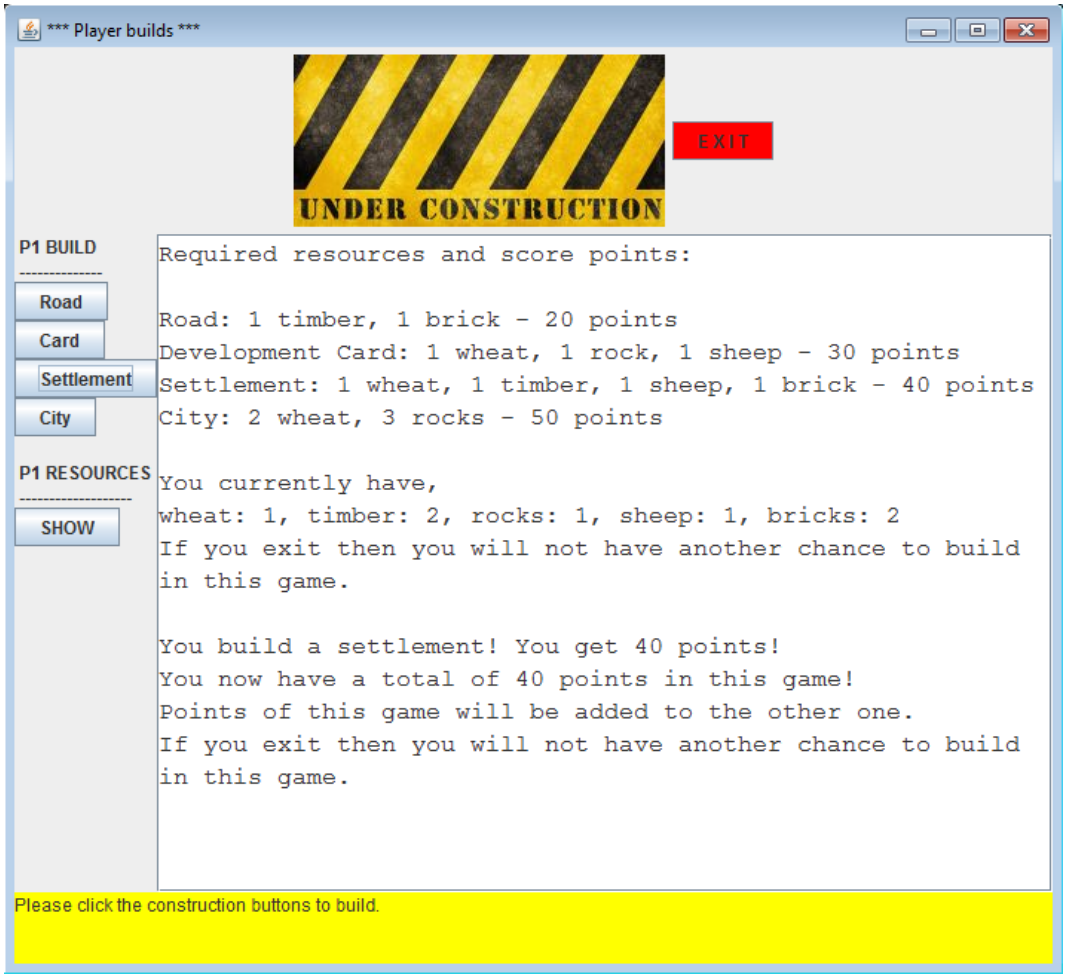


Figure 10.2: The building phase of the game "Trading in Catan".

10.1.3 Lessons from the pilot study

Based on the above information, first we conducted four pilot experiments between the two agents and four human players. The feedback that we obtained from the history of the games that they played and the questionnaires (similar to those of Sections 10.2.2 and 10.2.3) that they completed was very useful. We saw that with the current set-up of the evaluation, the players mostly rejected the agents' trading proposals. This is an issue that we confronted in Taikun experiments of Chapter 6 too. However the game mechanisms there promoted and allowed more trades. The players mostly rejected offers because they are quite cautious (we saw that in Taikun's case too) and because the initial number of resources allowed them to keep them and achieve a decent number of score points (50 in each game by building a road and a card). Even in the cases where the agent achieved 100 points by building a city from its random resource update only, the players would still not trade after that. Thus we had to change the number of initial resources for both of the players by randomizing them (each one 0, 1, or 2) in order to promote trading. We also reduced the points of the city to 50 (again) because 100 was too tempting for the players. That was another reason that they mainly aimed for building that and they rejected all of the agents' trades too. By reducing the points of the city to 50 we now offered again the same "magnitude" to each resource, as for example a road which requires two resources gives 20 points, a card which requires three resources gives 30 points and so on. Thus the game became more balanced.

We also examined the case where it would be easy for the players to understand whether or not the agents lie. In other words, we wanted to see what would happen when the lies were obvious. Two of the four players were informed at the beginning of the session that the agent wants to build only a city (therefore by asking for anything else apart from rocks and wheat would be a lie). The problem then was that they were rejecting the agent's proposals which were asking for wheat and rocks to prevent it from building that. Hence in this case, where the player knows that the agent wants to build a city at the beginning of the session, the agent's lies may be obvious but their effect was useless because the player knows that they are lies (as she knows that the agent wants to build a city only and therefore it should ask for only wheat and rocks). Our assumption on manipulation though was that the human player might believe the lie, thinking that it is the truth, and she might start boycotting (restricting) the corresponding resources and give others that the agent really needs. Thus we decided in our main experiments *not* to inform the player at the beginning of the game about the fact that the agent only aims to build a city, as the obvious lies offered nothing interesting. However we were still planning on checking whether the players think the agents lied or not, without making their lies obvious though. We wanted to see if the manipulative agent's deceitful strategy

would be still identified by some players (hence the questions 6. and 7. of the session's questionnaire of Section 10.2.2).

Another issue which promoted the rejections was that the agents had previously (Chapter 8) successfully learned to ask for two resources but the human players often did not have two to give them. Hence that caused even more rejections. That was another reason that we decided to initialise the resources at the beginning of the first game at random and give 0, 1, or 2 of each resource to both. Finally, the experiment's goal was to have the humans evaluate the agents' trading skill. That initially suggested a plain evaluation with no rules, no game in other words. Due to the fact that it might be monotonous for the players to only evaluate the agents' trading skill though we decided to do that through playing a game. However we had a few complaints that the game was not balanced and fair. That might be true but it was outside the scope of evaluation. The players though expected fair play, no matter what the goal of the experiment was. Hence at some point we were thinking of making an evaluation without having the form of a game.

10.2 Main experiments

By taking into consideration all of the above we decided to leave the evaluation in the form of a game, as it would be entertaining for the participants, and focus on making it as fair as possible (Section 10.2.1). 20 of them[3] played 2 sessions each, which consisted of 4 sequential games against an agent. The goal, as announced to the players, was to gather as many score points as possible and -if possible- more than the agents. To motivate them further we decided to give to the highest scorer a special gift, apart from the normal compensation they would receive for participating. At the end of each of the two sessions, each player completed a questionnaire about the agent that she played against (Section 10.2.2). At the end of the experiment each participant completed an overall questionnaire (Section 10.2.3).

10.2.1 Revised rules

The conclusions of Section 10.1.3 led us to make some important changes to the rules of the game as well as to its overall mechanisms. The revised rules, as announced to the twenty participants, were the following:

[3]The participants were all students, undergraduates and postgraduates.

- You will play 4 games against one opponent (agent). You will respond to its trading proposals and exchange resources. Your goal is to collect more total score points.
- You collect a particular number of score points every time you build something.
- To build something you need to collect a specific number of resources.
- The constructions to build along with the required resources will be given to you.
- The agent's resources are unknown to you.
- The agent wants to get a number of resources to build something. This can be done through your trades. Each of its resources may further change at random by -1 or +1.
- When the agent builds something the game continues normally.
- The game ends when 20 rounds have been played.
- You only build constructions at the end of each game. You can keep your resources and you can build everything at the end of the session (i.e. the end of the 4th game) if you prefer.
- You both start the first game with either 0, 1 or 2 of each resource (random).
- In the beginning of a new game, you both gain 1 of each resource.

An interesting question that needs to be answered before we proceed to the questionnaires is: is the game now fair? Considering that both of the players (humans and agents) start the game with random resources and at the beginning of each new game they both gain 1 of each type of resource, the answer so far would be "yes". However there are some features that the agent only has and some others that the player only has. In detail, the human player only responds to the agent's trading proposals and that makes his role harder. However the participant has the advantage of building any of the four constructions (i.e. road, card, settlement and city) while the agent on the other hand aims to build only cities. This is a big disadvantage for the agent, especially considering that it builds them during the game and the player soon realises that it aims only for that and can exploit that[4].

By considering the fact that the agent builds during the game while the player can build only at the end of the game we cannot find a reason to say that the agent has any advantage in this case, as the score points are eventually carried over to the end of the 4th game anyway. Finally the random update of the agent's resources[5] is

[4]This information was collected from the players themselves.

[5]Each resource may change in a uniformly distributed way by -1 or +1 or not change at all.

not an advantage either. The reason that we did that was to offer a variety of trading proposals to the RLA, because if we would keep the state constant it would always make the same trading proposals providing that no trades have happened[6]. Hence we believe that the game is fair, despite the fact that it is not balanced (analogous) in terms of game-play for both of the sides. With the following questionnaires we wanted to see what the results were and what do people think about it too. As we will see in Section 10.2.5, people found many ways to win.

10.2.2 Session's questionnaire

This is the questionnaire that each participant completed at the end of each session (i.e. after playing 4 sequential games with an agent):

1. Was your score higher than that of your opponent?
2. What was the reason that made you win/lose?
3. "This was a difficult opponent"

   (not agree) 1 2 3 4 5 6 7 8 9 10 (agree)
4. Please describe the opponent's strategy (e.g. It mainly asked for timber, I think it wanted to build a road...)
5. Please describe your strategy (e.g. I accepted everything because I needed all of the resources...)
6. "The opponent always asked for resources that it needed"

   (not agree) 1 2 3 4 5 6 7 8 9 10 (agree)
7. If you think that the opponent lied (i.e. asked for resources that it did not need), please explain what you did then. Did you change your strategy?
8. Would your strategy remain the same if you would play again versus the same opponent? If no, what would you change?

[6]Another reason that we implemented the random update of the agent's resources was to motivate the players to trade because in some games they rejected all of the agent's trading proposals. In this case though the agent still managed to build a city sometimes, therefore making them think that they should probably start trading.

10.2.3 Overall questionnaire

The two questions that follow consist the overall questionnaire that each participant completed at the end of the experiment:

1. By comparing the first with the second opponent, which one do you think was the smartest trader?
2. Why do you think that?

10.2.4 Comments on agents

In this section we will examine what the players thought about the agents. This will offer a "taste" about their overall strategies. We begin by mentioning first the most frequent comments based on the participants' questionnaires.

About the *manipulative* agent:

- Changed its proposals very often compared to the other opponent
- Hard to understand what the goal is
- Trading proposals always benefited the opponent more
- It accumulated its resources smarter than the other one
- It was too lucky
- It asked for 2 resources
- Sometimes it asked for something other than rocks or wheat
- It would keep itself from offering more often
- It did not give a good offer for the resources
- It was better than the other because it offered more viable trades (1-for-1)
- The opponent's offers were unfavourable trades but sometimes it wants 1-for-1
- I hoped it would propose one of the trades I wanted
- It cheated
- It had all that he needed yet faked by asking for what he needed not

- I rejected hoping that he would make a good offer yet he never did
- It had everything he needed and was only tricking me to keep his resources at the end of the game for maximum points
- It offered less valuable resources (sheep and bricks) for more valuable ones (rocks wheat)

About the *non-manipulative* agent:

- Aggressive / persistent
- Too greedy
- Trading proposals always benefited the opponent more
- It asked for 2 resources
- It did not give a good offer for the resources
- Offered more 1-for-1 to increase chances of trade
- Offers not worthwhile
- Offered me more realistic options / things I might need
- It always goes for equal or unfair trades, it usually demands more res for an item that you want
- It offered more favourable deeds
- It asked for the same resources lots of times
- It tried harder to get what it wanted offering alternative trades for the items it needed

10.2.5 How did the players win?

In this part we will examine the methods that the players think they used to beat the agents. In other words, we will see what the players answered in question 2 of the session's questionnaires (Section 10.2.2) in the case of win:

- I won because I was considering other things than cities

- I won because I aimed for equal resources distribution[7]
- I won because I aimed to accept only 1-for-1 trades and only those which moved me closer to balanced resources (i.e. same as above)
- I won because I hoarded resources until the last round and then I made notes of my resources and constructions (thus finding which resource I had an abundance of) and only accepted trades which advantaged me
- I worked options to trade
- I hoarded resources until the last round and then traded resources which I had abundance of
- (I understood that every resource is worth the same) + (not accepting unfair trades) + (luck)
- Lucky
- With one good trade
- With good initial resources
- I tried to hinder my opponent by rejecting most of his offers
- Once I gathered resources to build something I did not trade them

10.2.6 Results and discussion

According to the questionnaires that the twenty players completed during the experiment and the analysis of their logged games, the results are discussed below.

Objective measures:
• Out of 20 participants no one managed to beat both of the agents. This shows that the agents were skilful.
• 40 sessions (4 games each) were played and agents won 29 (72.5%) of them. Players only won 10 and there was 1 draw. As above, these results strongly suggest now that the agents were skilful.
• The manipulative agent had 14 wins (out of 20 sessions) and the non-manipulative (i.e. exclusively goal-oriented) had 15 wins (out of 20 sessions). These results suggest that the agents were (nearly) equally skilful.

[7]Surprisingly only a few people noticed that the road and the card (which both give 50 score points) both require only one resource of each type to be built. Hence a strategy that aims for a balanced number of resources is a very successful one, as it results in many score points by targeting roads and cards only.

• 5 players only (out of 20, 25%) managed to have a higher total score (from both of the sessions) than the agents. This result also suggests that the agents are skilful.
• 8 human players accepted more trades of the manipulative agent than those of the non-manipulative. 6 of them (75%) lost. The manipulative agent seemed to promote trading, compared to the non-manipulative one where 5 people accepted its trades more, and most of the players who traded lost. That indicates skill and a possible effect of manipulation as "lies" (i.e. trading proposals which ask for resources which are not really needed) serve the same purpose. The remaining 7 participants accepted the same number of trades from both of the agents.
• 14 human players (out of 20, 70%) thought that the manipulative agent might have lied during the games (6th question of the questionnaire in Section 10.2.2). 11 of those 14 (i.e. 78.57%) lost against the manipulative agent. This result strongly suggests that manipulation might be responsible for their loss. Of those 11 players, 3 of them changed their strategy (7th question of the questionnaire in Section 10.2.2) after they thought the agent lied (motivating our assumption about "gullibility" and hindering strategy as effects of manipulation, discussed in Section 3.3.3), 3 did not change their strategy and the rest 5 did not answer the question. These results show that the effect of confusion that manipulation (lies) causes[8] might have led to their loss.

Subjective measures:
• The manipulative agent was rated with 146 points total (avg. 7.3/10) and the non-manipulative with 147 points total (avg. 7.35/10) from the 3rd question of the session's questionnaires (Section 10.2.2). As we saw above, this result suggests again that the agents are skilful and, in fact, (nearly) equally skilful.
• According to the answers that the participants gave in the 1st question of the overall questionnaires (Section 10.2.3) the agents were equally smart. Half of the human players thought the manipulative agent was smarter while the other half thought that the non-manipulative agent was smarter.
• By considering the participants' ratings to the 6th question of the session questionnaires (Section 10.2.2) we saw that the manipulative agent receives 151 points of "honesty" (out of 200, that is completely agree that the agent always asked for resources that it needed) and the non-manipulative agent received 171 points out of 200. The manipulative agent's deceitful strategy was identified by some people but it was not very obvious[9].

[8]We saw that in Taikun too, people did not know how to proceed as they were in a dilemma, observed and discussed in Section 6.1.3, 7th bullet point.

[9]In some games the manipulative agent never lied, as its policy does not contains only lies, and therefore some people did not see any lies.

10.3 Conclusion

According to the above comments and results, it is evident that the agents were skilful. We restricted the luck factor in this chapter to the point where this can now be easily observed[10]. The luck (random) factor is designed to favour equally all of the interlocutors (agents and humans), in all of our experiments. We have noticed that whenever this factor was low, then the task of RL would become easier, as less noise would be then "captured" during the learning process. The manipulative agent had more trades accepted than the non-manipulative one and this was expected. The goal-oriented strategy of the non-manipulative agent makes it harder for humans to accept its proposals. We observed that in Chapter 6 too. On the other hand, the manipulative agent is more flexible in terms of offering, as it includes lies too (i.e. asks for other resources than only rocks and wheat, Figure 10.1), and therefore people trade with it more often. It was interesting to notice in this point that 75% of them though lost. The effect of manipulation as well as its success is evident here. The same is suggested by the fact that 78.57% of the people who thought that the agent might have lied, lost the games too. These results suggest that the third goal that we set at the beginning of the chapter was met: the manipulative agent through implicit manipulation was successful versus humans and affected them (e.g. confused them). In fact, 3 of them changed their strategy too. As we have mentioned, the two agents were very successful against the human players (won 75% of the sessions) proving that the first goal that we set in the beginning of this chapter was also met. Hence we show that trained (versus agents, Sections 8.1 and 8.2.5) RL trading dialogue agents are capable of being very successful in non-cooperative negotiations against human players.

The results also showed that the two agents were (nearly) equally successful, meeting our second goal that we set at the beginning of this chapter. Their wins, ratings from the participants along with their opinions about their "smartness" strongly suggested that. It was not possible to prove in this chapter though that the manipulative agent's policy was better than that of the non-manipulative one versus humans. However their performances agree with those that were discussed in Sections 9.1.5 and 9.2.4, where the same two policies were tested in the multilateral negotiations environment of Catan against the Bayes agents which simulated human behaviour. This agreement[11] suggests again that both policies are equally successful versus humans, and a further direct comparison between the two might offer no more information than that. It is important to recall at this point that the manipulative policy was tested in 10 thousand games against the Bayes agents (Sections 9.1.5 and

[10] In contrast to Taikun's case in Chapter 6, where the agent's skill was not as obvious due to the higher luck factor.

[11] In Chapter 6 we also saw that the wins of the manipulative agent (which used explicit manipulation there) were (nearly) equal to those of the non-manipulative one (Section 6.1.3).

9.2.4). It would probably require a very large number of games to be played against human players to investigate the full effect of the manipulation, regardless of using the dishonest or the honest agent, like we did with the agents[12]. The findings and the results so far, along with the particular number of games and human participants that were used in this chapter, suggested that we should probably refer to hundreds of participants (if not thousands) to (possibly) show that manipulation affects a significant amount of humans, in the ways that we have shown so far in our previous chapters[13].

[12]Millions of training and thousands of testing games have been played versus agents as we have seen in the thesis so far.

[13]Regardless of implicit or explicit manipulation, probably hundreds of human experiments are required to be played in order to provide solid evidence that linguistic manipulation offers an advantage versus humans, compared to a goal-oriented only (i.e. with no manipulation) strategy.

Chapter 11

Conclusion

This thesis has investigated the learning of trading dialogue policies in non-cooperative negotiations. We developed learning agents which were based on a custom SARSA(0) algorithm (Chapter 4) and mainly on SARSA(λ) (Chapter 5 onwards), as it resulted in more effective policies. Both met our expectations, as the Reinforcement Learning agents managed to successfully learn how to trade, especially through manipulation, with each algorithm having its own advantages and disadvantages regarding the running time, memory demands and success of the learned policy. The learning agents were capable of effectively trading with various adversaries, some of which were simulating human behaviour (Sections 9.1.5 and 9.2.4), in three different non-cooperative environments[1], showing that they are solvable by RL.

A range of research issues was explored:

- Bilateral negotiations (Chapters 3-8).
- Imperfect information about dialogue partners (Chapters 3-10)
- Non-stationary learning environments (Chapters 3-5 & 7-9)
- Use of explicit linguistic manipulation (Chapters 3-6)
- Use of implicit linguistic manipulation (Chapters 7-10)
- Adversaries who can detect manipulation (Sections 5.4 & 5.8).
- *How* and *when* to manipulate (Chapters 3-9 and Section 5.7 respectively).
- Methods of compressing the state space (Section 7.2.5)
- Models of adversarial preferences (Sections 8.3 & 8.4)

[1]These environments were the bilateral negotiations games Taikun and our own version of Catan, as well as the multilateral environment JSettlers.

- Multilateral negotiations (Chapter 9).
- Evaluation of RL policies against humans (Chapters 6 and 10)

In all of these areas, the thesis has demonstrated positive results.

11.1 List of contributions

The contributions of this research are the following:

1. Reinforcement Learning: Successful learning of tabular RL policies in bilateral non-cooperative environments where there is imperfect information about the participants' goals, states and preferences. A variety of opponents where used through the thesis (Chapters 3-5 and 7-8).
2. Reinforcement Learning: Successful learning in the above environments even where their dynamics continuously change, therefore making the problem a non-stationary MDP (Chapters 3-5 and 7-8).
3. Reinforcement Learning and Pragmatics: RL successfully learns to use explicit (i.e. through scalar implicature, Chapters 4 and 5) and implicit (i.e. through normal trading proposals, Chapters 7 and 8) linguistic manipulation and increases its performance in the above non-cooperative environments (Chapters 3-5 and 7-9).
4. Reinforcement Learning, Pragmatics and Psychology: Our explicit and implicit manipulating RL agents have demonstrated that they are capable of learning how to beat manipulated adversaries which demonstrate signs of human behaviour (Chapters 3-5 and 7-9). These adversaries are gullible to the RL agent's manipulation and hinder the stated resources from the agent (based on the reasons of Section 3.3.3).
5. Reinforcement Learning and Pragmatics: We have shown that our RL agents have learned to manipulate by being cooperative on the Attardo's [6] locutionary level and non-cooperative on the perlocutionary level (Chapters 3-5 and 7-9).
6. Reinforcement Learning: We have demonstrated that the above RL agents learn how to successfully manipulate even in cases where there are risks of exposure (detection) where severe penalties apply (Section 5.4).

7. Reinforcement Learning and Psychology: Adversaries which detect manipulation based on logical contradictions have been implemented and have been shown to be harder to beat than the above exposing ones. Manipulating RL policies have been able to beat these adversaries during evaluation, even without previous training against these adversaries (Section 5.8).

8. Reinforcement Learning and Psychology: We have shown that RL trading agents are capable of learning not only *how*, but also *when* is the best time to (safely) manipulate, demonstrating human reasoning according to the dual-mind cognition theory of Evans [31, 32] (Section 5.7).

9. Reinforcement Learning and Philosophy: Our RL agents through implicit manipulation bring an important argument of Van Dijk [20] to light, according to which there is an everyday conventional inference of dishonesty from manipulative acts. That negative effect cannot be taken for granted though as manipulation according to Dillard and Pfau [21], as well as O'Keefe [78] also occurs through legitimate persuasion. This is what our Reinforcement Learning work suggests too with the similar performances of our deceptive and persuasive agents (Section 8.2).

10. Reinforcement Learning: We have presented a novel way of encoding the state space of tabular RL trading dialogues which reduces its size to 0.5% of the original (Section 7.2.5).

11. Reinforcement Learning: We have shown that trading RL agents are more successful when they build an opponent model, an estimate of the hidden goals and preferences of the adversary and learn how to exploit them (Section 8.3).

12. Reinforcement Learning: Conditional preference networks (CP-NETs) [11] have been implemented and used for the first time ever in the RL agent's state representation, resulting in higher performances than those of RL agents which did not use them (Section 8.4).

13. Reinforcement Learning: Our previously trained RL policies in bilateral negotiations environments have shown to perform successfully in multilateral negotiations environments, by treating all of the opponents as one, performing even better than the policies which were trained on these multilateral negotiations environments (Section 9.1.3). Previous efficient selection of states and actions in our 2-player version of the "Catan" game during training assisted in that too. Hence we suggest that bilateral training environments may suffice for complex multilateral non-cooperative trading scenarios, for effective RL (Chapter 9).

14. Reinforcement Learning and Pragmatics: Our RL agent with implicit linguistic manipulation was successful too in the above multilateral environments, and has been shown to beat Bayesian adversaries which simulate human behaviour (Section 9.2).

15. Reinforcement Learning: Our above trained RL policies on bilateral negotiations which maintained preferences in the state representation, resulted in decent performances against the adversaries of the above multilateral negotiations environment (Section 9.4).

16. Reinforcement Learning and Pragmatics: Our trading RL agents were successful versus human players in our tested non-cooperative environments. Explicit (Chapter 6) and implicit (Chapter 10) linguistic manipulation was effective against the humans and our RL agents managed to win more games than them.

11.2 Personal evaluation and reflection

The outcome of this thesis met most of our expectations. There is still important work to be done though in order to enhance the agent's trading capabilities, and especially to reach a level where it will be capable of successfully negotiating in other non-cooperative negotiation domains than that of trading. Frankly, this is what we aim for in the long term. Our results from experiments based on Pragmatics provide evidence that generalisation to other non-cooperative negotiation domains is possible, as we will discuss in more detail in the next section. Our Reinforcement Learning in an MDP framework has shown that it is capable of learning how to effectively negotiate by demonstrating similarity to human behaviour. We have demonstrated for example that our RLA's are able to learn how to successfully use explicit manipulation through scalar implicature, or implicit manipulation through the use of normal trading proposals by being persuasive or deceptive. However a "multi-purpose" non-cooperative negotiations agent would need to consider even more aspects of human behaviour (such as trust), in order to improve its manipulative negotiating performance. By taking into account more human traits it will probably affect a wider range of human behaviours, and therefore become applicable to more negotiation domains.

Instead of using RL code that was already implemented by other researchers we preferred to create our own, always following fundamental principles of RL. The advantage of that plan was (apart from the enjoyment of creativity) that we gained technical knowledge and the flexibility to create algorithms that fully served

our research needs, piece by piece. The disadvantage of that plan was probably wasted time, as time was spend to initially investigate and experiment with trivial problems (e.g. the RL solution of the tic-tac-toe game). However we now have a learning algorithm that can easily be modified according to the problem's needs and after much trial and error has reached a decent level. Our transition from the simple tic-tac-toe game to the complex multi-player Catan game eventually led us to that level and we produced methodologies that made the use of the traditional tabular RL easier (e.g. our compressed state space representation).

Empirical analysis and theories that were used in our work were mainly focused on Pragmatics, some on Philosophy and some on Psychology. At the beginning of this research time was dedicated to study Game Theory, but we soon realised that we were more interested in researching the negotiation part of a game, as this is where the dialogue management mainly occurs, rather than the overall game-play strategies. However Game Theory inspired us to create the non-cooperative games Taikun and our version of the game Catan[2], in a way that could serve our research needs.

11.3 Future work

In this section we will initially discuss how the current findings can be applied to other areas in an attempt to generalise our work. We will also examine which areas can be affected. At the end of this section we will provide an outlook discussing the vision of this work.

11.3.1 RL negotiations generalisation

The findings of the current thesis can be useful and might be applied in many non-cooperative negotiations scenarios and not only trading. Political debates, police investigation, tutoring and court trials are some examples where our findings using explicit linguistic manipulation, as is presented in the form of scalar implicature in this work, could be learned by RLAs in order to lie or hide information. Despite the fact that we were focused on trading, our resource exchange might also be used for basic information exchange in these cases and our learning mechanisms would still remain the same. Implicit manipulation, where the agent has learned how to deceive or persuade through normal trading proposals, might also be applied in

[2]as well as conduct the evaluation of Chapter 10 through our newest game "Trading in Catan".

other forms of negotiation, as the *trading resource* would be translated again to the corresponding *information* that the particular negotiation focuses on.

Apart from *how* to manipulate, information regarding *when* would be also useful to apply to other negotiation areas, as relevant theory in Psychology suggests and we have previously discussed (Section 5.7). As the current thesis proposes though, detecting manipulation is as important as manipulating. Thus, our methodologies and findings on manipulation detection, with agents which identify it based on its frequency or its logical inconsistency, are suggested and can be taken into consideration in the agents' functionality of various other negotiation areas, such as those that we mentioned above. Manipulation and detection *can* be learned and used by intelligent agents in other forms of negotiation, as our trade resources can be translated to any other kind of information exchange. For example, the scalar implicature's effect (Section 3.1.3) from the sentence "I really need *resource*" that the agent has learned to use at the beginning of this work, might also be triggered from "I really need *information*", as in a police investigation negotiation, and used for exactly the same manipulative purpose that we have examined in the thesis. In both of these cases the intelligent agent would use it in order to get **other** *resources* or *information* than the one it says it needs, in such a non-cooperative environment.

11.3.2 Outlook

In this thesis we have provided a first investigation into a whole range of issues in non-cooperative trading dialogue. In future we hope that our results are useful in the new field of "Computational Pragmatics", and could be extended for example to use Deep Learning for non-cooperative dialogue management [18]. Ultimately, the vision of this research is to eventually develop intelligent agents which will be able to assist (or even replace) the user when (s)he comes to take important decisions in non-cooperative negotiations with other humans or robots in various sectors. Examples are the games/video-games sector, financial sector, political and diplomatic sector, healthcare, police investigation, military operations and education.

Bibliography

[1] Stergos Afantenos, Nicholas Asher, Farah Benamara, Anais Cadilhac, Cedric Degremont, Pascal Denis, Markus Guhe, Simon Keizer, Alex Lascarides, Oliver Lemon, Philippe Muller, Soumya Paul, Verena Rieser, and Laure Vieu, 'Developing a corpus of strategic conversation in The Settlers of Catan', in *Proceedings of SemDial 2012*, (2012).

[2] R. Arkin, 'The ethics of robotics deception', in *1st International Conference of International Association for Computing and Philosophy*, pp. 1–3, (2010).

[3] N. Asher, E. Bonzon, and A. Lascarides, 'Extracting and modelling preferences from dialogue'. submitted, 2009.

[4] N. Asher and A. Lascarides, 'Commitments, beliefs and intentions in dialogue', in *Proc. of SemDial*, pp. 35–42, (2008).

[5] N. Asher and A. Lascarides, 'Making the right commitments in dialogue', in *University of Michigan Linguistics and Philosophy Workshop*, (2008).

[6] S. Attardo, 'Locutionary and perlocutionary cooperation: The perlocutionary cooperative principle', *Journal of Pragmatics*, **27**(6), 753–779, (1997).

[7] N. Barr, *Economics of the Welfare State (5th Edition)*, chapter 3.2.2: The relevance of efficiency to different theories of society, 46, Oxford University Press, 2012.

[8] M.H.J. Bergsma, *Opponent Modeling in Machiavelli*, B.s. thesis, Maastricht University, the Netherlands, 2005.

[9] C.F. Bond and M. Robinson, 'The evolution of deception', *Journal of Nonverbal Bahavior*, **12**(4), 295–307, (1988).

[10] V.S. Borkar, 'Stochastic approximation with two time scales', *Systems Control Letters*, **29**(5), 291–294, (1997).

[11] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'CP-nets : A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements', *Journal of Artificial Intelligence Research*, **21**, 135–191, (2004).

[12] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin, 'Heads-up limit hold'em poker is solved', *Science*, **347**(6218), 145–149, (2015).

[13] A. Byde, M. Yearworth, Y.-K. Chen, and C. Bartolini, 'Autona: a system for automated multiple 1-1 negotiation', in *Proceedings of the 2003 IEEE International Conference on Electronic Commerce*, pp. 59–67, (2003).

[14] Anaïs Cadilhac, Nicholas Asher, Farah Benamara, and Alex Lascarides, 'Commitments to preferences in dialogue', in *Proceedings of the SIGDIAL 2011 Conference*, pp. 204–215, Portland, Oregon, (June 2011). Association for Computational Linguistics.

[15] C. Camerer and T.H. Ho, 'Experience-weighted attraction learning in normal form games', *Econometrica*, **67**(4), 827–874, (1999).

[16] D. Carmel and S. Markovitch, 'Learning models of opponent's strategies in game playing', in *Proceedings AAAI Fall Symposium on Games: Planning and Learning*, pp. 140–147. The AAAI Press, (1993).

[17] Samuel P. M. Choi, Dit-Yan Yeung, and Nevin L. Zhang, 'Hidden-mode markov decision processes for nonstationary sequential decision making', in *In Sequence Learning - Paradigms, Algorithms, and Applications*, pp. 264–287. Springer-Verlag, (2001).

[18] Heriberto Cuayahuitl, Simon Keizer, and Oliver Lemon, 'Strategic dialogue management via deep reinforcement learning.', in *NIPS Deep Reinforcement Learning workshop*, Canada, (2015).

[19] Daniel Dennett, 'When Hal Kills, Who's to Blame? Computer Ethics', in *Hal's Legacy:2001's Computer as Dream and Reality*, (1997).

[20] T.A.V. Dijk, 'Discourse and manipulation', *Discourse & Society*, **17**(2), 359–383, (2006).

[21] James Price Dillard and Michael Pfau, *The Persuasion Handbook: Developments in Theory and Practice*, SAGE Publications, Inc., 2002.

[22] Peng Ding and Tao Mao, 'Reinforcement Learning in Tic-Tac-Toe Game and Its Similar Variations', Technical report, Thayer School of Engineering at Dartmouth College, (2009).

[23] R.V. Dodge, *Schelling's Game Theory*, Oxford Scholarship Online, 2012.

[24] H. H. L. M. Donkers, H. J. Van Den Herik, and J. W. H. M. Uiterwijk, 'Probabilistic opponent-model search', *Information Sciences*, **135**, 123–149, (2001).

[25] Ioannis Efstathiou and Oliver Lemon, 'Learning non-cooperative behaviour for dialogue agents', in *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, pp. 999–1000, Prague, Czech Republic, (2014).

[26] Ioannis Efstathiou and Oliver Lemon, 'Learning non-cooperative dialogue behaviours', in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 60–68, Philadelphia, PA, U.S.A, (2014).

[27] Ioannis Efstathiou and Oliver Lemon, 'Learning to manage risks in non-cooperative dialogues.', in *Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2014 - DialWatt)*, pp. 173–175, Edinburgh, Scotland, U.K., (2014).

[28] Ioannis Efstathiou and Oliver Lemon, 'Learning non-cooperative dialogue policies to beat opponent models: "the good, the bad and the ugly"', in *Proceedings of the 19th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2015 - GoDial)*, pp. 33–41, Gothenburg, Sweden, (2015).

[29] Ioannis Efstathiou and Oliver Lemon, 'Learning better trading dialogue policies by inferring opponent preferences.', in *To be presented: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, Singapore, (2016).

[30] I. Erev and A. Rapoport, 'Coordination, "magic", and reinforcement learning in a market entry game', *Games and Economic Behavior*, **23**, 146–175, (1998).

[31] J.S.B.T. Evans, 'Dual-processing accounts of reasoning, judgment, and social cognition', *Annual Review of Psychology*, **59**(1), 255–278, (2008).

[32] J.S.B.T. Evans, 'Two minds rationality', *Thinking & Reasoning*, **20**(2), 129–146, (2013).

[33] D. Floreano, S. Mitri, S. Magnenat, and L. Keller, 'Evolutionary conditions for the emergence of communication in robots', *Current Biology*, **17**(6), 514–519, (2007).

[34] D. Fudenberg and J. Tirole, *Game Theory*, chapter 1.1: Introduction to Games in strategic Form and Iterated strict Dominance, 4–11, MIT Press, 1991.

[35] Kallirroi Georgila, 'Reinforcement learning of two-issue negotiation dialogue policies', in *Proceedings of the 14th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 112–116, (2013).

[36] Kallirroi Georgila, Claire Nelson, and David Traum, 'Single-agend vs. multi-agent techniques for concurrent reinforcement learning of negotiation dialogue

policies', in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 500–510, Baltimore, USA, (September 2014).

[37] Kallirroi Georgila and David Traum, 'Learning culture-specific dialogue models from non-culture specific data', in *Proceedings of Universal Access in Human-Computer Interaction, HCI International, Lecture Notes in Computer Science*, volume 6766, pp. 440–449. Springer Berlin Heidelberg, (2011).

[38] Kallirroi Georgila and David Traum, 'Reinforcement learning of argumentation dialogue policies in negotiation', in *Proceedings of INTERSPEECH*, (2011).

[39] H.P. Grice, 'Syntax and semantics: Speech acts', in *Logic and Conversation*, eds., P. Cole and J.L. Morgan, volume 3, 41–58, New York: Academic Press, (1975).

[40] Markus Guhe and Alex Lascarides, 'Game strategies in *the settlers of catan*', in *Proceedings of the IEEE Conference on Computational Intelligence in Games*, Dortmund, (2014).

[41] A. Hadi, 'A critical appraisal of grice's cooperative principle', *Open journal of Modern Linguistics*, **3**(1), 69–72, (2013).

[42] Christos Hadjinikolis, Yiannis Siantos, Sanjay Modgil, Elizabeth Black, and Peter McBurney, 'Opponent modelling in persuasion dialogues', in *IJCAI '13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 164–170, (2013).

[43] Melita Hajdinjak and France Mihelic, 'Information-providing dialogue management', in *Lecture Notes in Computer Science*, eds., Petr Sojka, Ivan Kopecek, and Karel Pala, volume 3206, 595–602, Springer, (2004).

[44] J.M. Hammersley and D.C. Handscomb, *Monte Carlo Methods*, Methuen and Co., London, and John Wiley and Sons, New York, 1964.

[45] Peter Heeman, 'Representing the reinforcement learning state in a negotiation dialogue.', in *Automatic Speech Recognition & Understanding*, pp. 450–455, Merano, Italy, (2009).

[46] Takuya Hiraoka, Kallirroi Georgila, Elnaz Nouri, David Traum, and Satoshi Nakamura, 'Reinforcement learning in multi-party trading dialog', in *Proceedings of the SIGDIAL 2015 Conference*, pp. 32–41, Prague, Czech Republic, (September 2015).

[47] Takuya Hiraoka, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura, 'Reinforcement learning of cooperative persuasive dialogue policies using framing', in *The 25th International Conference on Computational Linguistics (COLING)*, pp. 1706–1717, (2014).

[48] J. Hu and M.P. Wellman, 'Multiagent reinforcement learning: Theoritical framework and an algorithm', in *Proceedings of the Fifteenth International Conference on Machine Learning*, San Francisco, (1998).

[49] H. Iida, J.W.H.M. Uiterwijk, H.J. van den Herik, and I.S. Herschberg, 'Opponent-model search', Technical report cs 93-03, Universiteit Maastricht, (1993).

[50] H. Iida, J.W.H.M. Uiterwijk, H.J. van den Herik, and I.S. Herschberg, 'Potential applications of opponent-model search. part 1: The domain of applicability', *ICCA Journal*, **16**(4), 201–208, (1993).

[51] Levin Irwin, Sandra L. Schneider, and Gary J. Gaeth, 'All frames are not created equal: A typology and critical analysis of framing effects.', *Organizational behavior and human decision processes*, **76**(2), 149–188, (1998).

[52] H. Ishibuchi, T. Nakashima, H. Miyamoto, and C. Oh, 'Fuzzy q-learning for a multi-player non-cooperative repeated game', in *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, Barcelona, (1997).

[53] S. Ishii, W. Yoshida, and J. Yoshimoto, 'Control of exploitation-exploration meta-parameter in reinforcement learning', *Neural Networks*, **15**, 665–687, (2002).

[54] ISO24617-2, 'Language resource management-semantic annotation frame work (semaf), part2: Dialogue acts', Iso, International Organization for Standardization, (2010).

[55] B. Joy, 'Why the future doesn't need us', *Wired*, (April 2000).

[56] L.P. Kaelbling, M.L. Littman, and A. Moore, 'Reinforcement learning: A survey', *Journal of Artificial Intelligence*, **4**, 243, (1996).

[57] I. Kaplansky, 'A contribution to von neumann's theory of games', *Annals of Mathematics*, **46**(3), 474–479, (1945).

[58] R. Katz and S. Kraus, 'Efficient agents for cliff-edge environments with a large set of decision options.', in *Proceedings of the 5th International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 697–704, (2006).

[59] R. Katz and S. Kraus, 'Gender-sensitive automated negotiators', in *Proceedings of the 22nd National Conference on Artificial Intelligence*, pp. 821–826, (2007).

[60] M. Kearns and S. Singh, 'Near-optimal reinforcement learning in polynomial time', *Machine Learning*, **49**, 209–232, (2002).

[61] Simon Keizer, Heriberto Cuayahuitl, and Oliver Lemon, 'Learning trade negotiation policies in strategic conversation', in *The 19th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2015 - goDIAL)*, pp. 104–112, (2015).

[62] J.F. Kelley, 'Cal, a natural language program developed with the oz paradigm: Implications for supercomputing systems', in *First International Conference on Supercomputing Systems*, pp. 238–248, (1985).

[63] S. Kraus, P. Hoz-Weiss, S. Wilkenfeld, D.R. Andersen, and A. Pate, 'Resolving crises through automated bilateral negotiations.', *Artificial Intelligence*, **172**(1), 1–18, (2008).

[64] H.J. Ladegaard, 'Pragmatic cooperation revisited: Resistance and non-cooperation as a discursive strategy in asymmetrical discourses', *Journal of Pragmatics*, **41**(4), 649–666, (2009).

[65] Michail G. Lagoudakis and Ronald Parr, 'Least-squares policy iteration', *Journal of Machine Learning Research*, **4**, 1107–1149, (2003).

[66] D.S. Leslie and E.J. Collins, 'Convergent multiple-times-scales reinforcement learning algorithms in normal form games', *The Annals of Applied Probability*, **13**(4), 1231–1653, (2003).

[67] R. Lin and S. Kraus, 'Can automated agents proficiently negotiate with humans?', *CACM*, **53**(1), 78–88, (2010).

[68] Michael L. Littman, 'Markov games as a framework for multi-agent reinforcement learning', in *Proceedings on the Eleventh international Conference on Machine Learning*, pp. 157–163, (1994).

[69] T. Meguro, Y. Minami, R. Higashinaka, and K. Dohsaka, 'Wizard of oz evaluation of listening-oriented dialogue control using pomdp', in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop*, pp. 318–323, (2011).

[70] Francisco S. Melo, Sean P. Meyn, and M. Isabel Ribeiro, 'An analysis of reinforcement learning with function approximation', in *Proceedings of the*

25th International Conference on Machine Learning, ICML '08, pp. 664–671, New York, NY, USA, (2008). ACM.

[71] T. Mitchell, *Machine Learning*, chapter 13: Reinforcement Learning, 367–388, The McGraw-Hill Companies Inc., 1997.

[72] J. Nash, *Non-Cooperative Games*, Princeton University, 1950.

[73] J. Nash, 'Non-cooperative games', *The Annals of Mathematics*, **54**(2), 286–295, (1951).

[74] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Bahaviour (3rd Edition)*, Princeton University Press, 1953.

[75] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behaviour*, chapter 11.1.1: The concept of a strategy and its formalization, 79, Princeton University Press, 1953.

[76] B. Norberto, *On Mosca and Pareto*, Librairie Droz, 1972.

[77] Elnaz Nouri and David Traum, 'Initiative taking in negotiation', in *Proceedings of SIGDIAL 2014*, (2014).

[78] Daniel O'Keefe, *Persuasion: Theory and research (2nd Edition)*, SAGE Publications, Inc., 2002.

[79] Alexandros Papangelis and Kallirroi Georgila, 'Reinforcement learning of multi-issue negotiation dialogue policies', in *Proceedings of the SIGDIAL 2015 Conference*, pp. 154–158, (2015).

[80] Michael Pfeiffer, 'Reinforcement learning of strategies for Settlers of Catan', in *International Conference on Computer Games: Artificial Intelligence, Design and Education*, (2004).

[81] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.

[82] V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*, chapter 2: Background, 9–27, Springer, 2011.

[83] V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*, chapter 3: Reinforcement Learning, 29–52, Springer, 2011.

[84] Verena Rieser and Oliver Lemon, *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*, Theory and Applications of Natural Language Processing, Springer, 2011.

[85] W. Ross and J. LaCroix, 'Multiple meanings of trust in negotiation theory and research: a literature review and integrative model', *International Journal of Conflict Management*, **7**(4), 314–360, (1996).

[86] A. Rudnicky and W. Xu, 'An agenda-based dialogue management architecture for spoken language systems', in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, (1999).

[87] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, chapter 2: Intelligent Agents, 34–61, Prentice Hall, 2003.

[88] Jost Schatzmann and Steve Young, 'The hidden agenda user simulation model', *IEEE Transactions on Audio, Speech, and Language Processing*, **17**(4), 733–747, (2009).

[89] N. Sharkey, 'The ethical frontiers of robotics', *Science*, **322**, 1800–1801, (2008).

[90] J. Shim and R.C. Arkin, 'A Taxonomy of Robot Deception and its Benefits in HRI', in *Proc. IEEE Systems, Man, and Cybernetics Conference*, (2013).

[91] B.C. Silva, E.W. Basso, A.L.C. Bazzan, and P.M. Engel, 'Dealing with non-stationary environments using context detection', in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh PA, (2006).

[92] Bruno C. Da Silva, Eduardo W. Basso, Filipo S. Perotto, Ana L. C. Bazzan, and Paulo M, 'Improving reinforcement learning with context detection', in *Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems - AAMAS 2006*, (2006).

[93] R. Sutton, 'Learning to predict by the methods of temporal differences', *Machine Learning*, **3**(1), 9–44, (1988).

[94] R. Sutton and A. Barto, *Reinforcement Learning*, chapter 6.5 Q-Learning: Off-Policy TD Control, 185–188, MIT Press, 1998.

[95] R. Sutton and A. Barto, *Reinforcement Learning*, chapter 6.4 SARSA: On-Policy TD Control, 182–184, MIT Press, 1998.

[96] R. Sutton and A. Barto, *Reinforcement Learning*, chapter 7.5 SARSA(λ), 226–228, MIT Press, 1998.

[97] R. Sutton and A. Barto, *Reinforcement Learning*, MIT Press, 1998.

[98] R. Sutton and A. Barto, *Reinforcement Learning*, chapter 1: Introduction, 14–35, MIT Press, 1998.

[99] R. Thomas and K. Hammond, 'Java settlers: a research environment for studying multi-agent negotiation', in *Proc. of IUI '02*, pp. 240–240, (2002).

[100] M. Tokic, 'Ki 2010: Advances in artificial intelligence, karlsruhe, germany 2010. lecture notes in artificial intelligence', in *Adaptive e-greedy exploration in reinforcement learning based on value differences*, eds., R. Dillmann, J. Beyerer, U. Hanebeck, and T. Schultz, 203–210, Springer, (2010).

[101] David Traum, 'Extended abstract: Computational models of non-cooperative dialogue', in *Proc. of SIGdial Workshop on Discourse and Dialogue*, (2008).

[102] N. Vishnu and D. Tapas, 'A reinforcement learning algorithm for obtaining nash equilibrium of multi-player matrix games', *IIE Transactions*, **41**(2), 158–167, (2009).

[103] Adam Vogel, Max Bodoia, Christopher Potts, and Dan Jurafsky, 'Emergence of gricean maxims from multi-agent decision theory', in *Proceedings of NAACL 2013*, (2013).

[104] Adam Vogel, Christopher Potts, and Dan Jurafsky, 'Implicatures and nested beliefs in approximate decentralized-pomdps', in *Proceedings of ACL 2013*, (2013).

[105] Aimilios Vourliotakis, Ioannis Efstathiou, and Verena Rieser, 'Detecting deception in non-cooperative dialogue: A smarter adversary cannot be fooled that easily', in *Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*, pp. 252–254, Edinburgh, Scotland, U.K., (September 2014).

[106] A.R. Wagner and R.C. Arkin, 'Analyzing social situations for human-robot interaction', *Interaction Studies*, **9**(2), 277–300, (2008).

[107] A.R. Wagner and R.C. Arkin, 'Robot deception: Recognizing when a robot should deceive', in *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA-09)*, Daejeon, South Korea, (2009).

[108] 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).

[109] M. Walker, R. Passonneau, and J. Boland, 'Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems', in *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, (2001).

[110] C.J.C.H. Watkins, *Learning from Delayed Rewards*, Cambridge University, 1989.

[111] C.J.C.H. Watkins and P. Dayan, 'Q-learning', *Machine Learning*, **8**, 279–292, (1992).

[112] Jason D. Williams and Steve Young, 'Partially observable markov decision processes for spoken dialog systems', *Computer Speech and Language*, **21**(2), 393–422, (2007).

[113] Steve Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, 'The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management', *Computer Speech and Language*, **24**(2), 150–174, (2010).

[114] X. Zhang, V. Lesser, and R. Podorozhny, 'Multi-dimensional, multistep negotiation for task allocation in a cooperative system', *Autonomous Agents and MultiAgent Systems*, **10**(1), 5–40, (2005).

Appendix A

Appendix

In this chapter we will initially discuss the preliminary work that preceded the above research and offered practical experience, aim and motivation. The reader will then find various algorithms that were used in the thesis and other draft notes about the tic-tac-toe game, poker and Taikun.

A.1 Preliminary work

In this section we will examine the preliminary work that has been made. The purpose of this work was to understand in depth the Reinforcement Learning programming in an MDP framework. This allowed to initially deal with trivial problems and effectively traverse to more complex ones. One of the trivial problems was to successfully learn how to move in a grid-world environment to reach a goal point. Along with the familiarisation with the Reinforcement Learning, another significant reason that this preliminary work was conducted was to solve trivial problems in non-cooperative games, such as the "Tic-tac-toe". This offered fundamental knowledge and a fist impression of how a more complex non-cooperative trading environment should be created and dealt (i.e. Taikun). Initially we were thinking of working in poker but -as we will see below- we soon changed our mind because the nature of that problem did not fit exactly to our future plans and needs of the current research. Hence we decided to create our own game "Taikun" that we have examined in Chapters 3, 4, 5 and 6.

A.1.1 Q-Learning example on a grid-world

This program was implemented in Java using the NetBeans IDE 7.3 RC2. It is a Reinforcement learning example that was based on Q-Learning in an MDP framework. The knowledge that was gained from that assisted in the development of the algorithms that followed. It used a 3x4 grid world (similar to the examples of the Section 2.1.4) where the agent's goal was to learn how to find the shortest route towards a goal state. At the beginning of each episode (iteration), the agent always "spawned" at the 3-1 (row-column) state and gradually learnt a policy of how to effectively reach the goal state at 1-4 by following the shortest possible route. That was achieved by constantly updating the Q-values of the state-action pairs while it was traversing through the various states, by applying the appropriate formula for the deterministic case (Section 2.1.3). The program starts by applying 2 rewards of value 10 to the actions towards the goal state from the states 1-3 and 2-4, -1 reward to every other action and 0 to the actions of the goal state. Once the agent reaches the goal state then it automatically spawns again at the state 3-1 (beginning of next learning episode) until convergence has been reached (Q-values of the actions do not change any more).

This program covered both the cases (and therefore there are 2 different versions of the program) of deterministic and non-deterministic actions. Parameter α (learning rate) is set to 1 in the former case while on the latter it has to change to 0.1 (recommended value). The non-deterministic case was programmed by applying probabilities to the agent's movements and then stop the execution of the iterations to a number that we are confident with (we can be certain that the algorithm has learnt the optimal policy there). In another case, where there can be no such confidence, the formula that was used to update the Q-values of the state-actions is that of the stochastic case and for the parameter a was used a specific formula too (examined in detail at the end of Section 2.1.4).

A.1.2 Tic-Tac-Toe

This program was implemented in Java using the NetBeans IDE 7.3 RC2. The reason that it was designed and implemented was to experiment with a custom made Temporal-Difference learning agent based on value iteration in a non-cooperative game with perfect information, such as the "tic-tac-toe", in order to pass this knowledge further to our future work. Inspired by the relevant example of [97], the learning agent being in an e-greedy mode with a ratio of 80%/20% (greedy/exploring), initially managed successfully to learn how to play on a satisfactory level (after 40,000

training games) and then how to never lose a game (after 80,000 training games), see Figures A.1 and A.2, against an agent that was playing at random. The result was decent compared to others [22]. The rewards (converted to state values) that were used were 0 for the case of adversary's win, 8,000 for the case of a draw, 5,000 for any other case (of a normal state) and 10,000 for the case of the learning agent's win. The Temporal-Difference Learning method calculates current value estimates using previous learnt ones. In this program, the applied formula that was used and is based on Value iteration is:

$V'(s) \leftarrow V(s) + \alpha * (V(s') - V(s))$ where,

α is the learning rate ($0 \leq \alpha \leq 1$), it reduces gradually to 0 at the end of the training games for the sake of convergence,

s' and s the current and previous states respectively,

V' and V are the current and previous values.

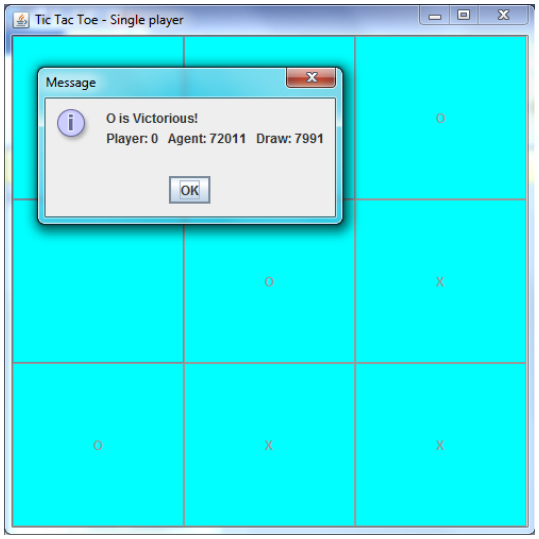


Figure A.1: *Learning Agent's performance after 80,000 testing games of "tic-tac-toe". Trained on 80k games and then tested on 80k games versus an opponent which plays at random (named above as "player"), our learning agent has learnt how to never lose a game.*

Notes about the LA's algorithm (1st PhD year)

The game (environment) presents to the agent new states due to the player's "unpredictable" intervention (the agent is not aware of what the player is going to play). Thus, the adversary is considered to be a part of the learning agent's environment. The states of the player and the agent are all recorded in an array

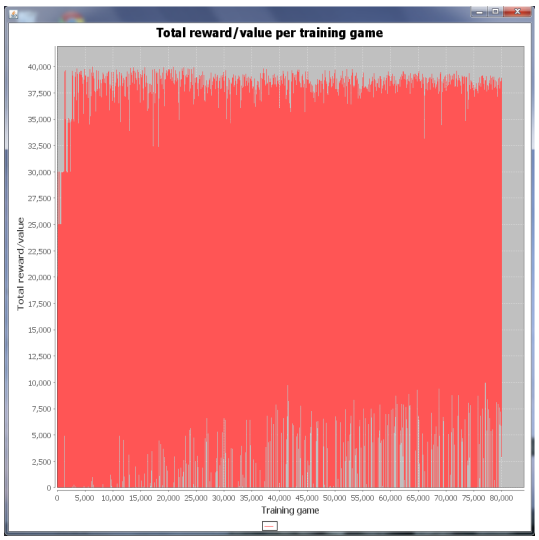


Figure A.2: *Total reward/value graph for each of the 80,000 training games of "tic-tac-toe". The above graph provided to us evidence of the algorithm's correct functionality. The learning agent, while in greedy mode, learnt how to consider only those state-actions that have the highest values, as the number of the training games increases.*

list in the way that is explained in Figure A.3, where X is the state of the game including the action (move) of the player (or adversary) who always starts the game first). O is the state of the game including our learning agent's action.

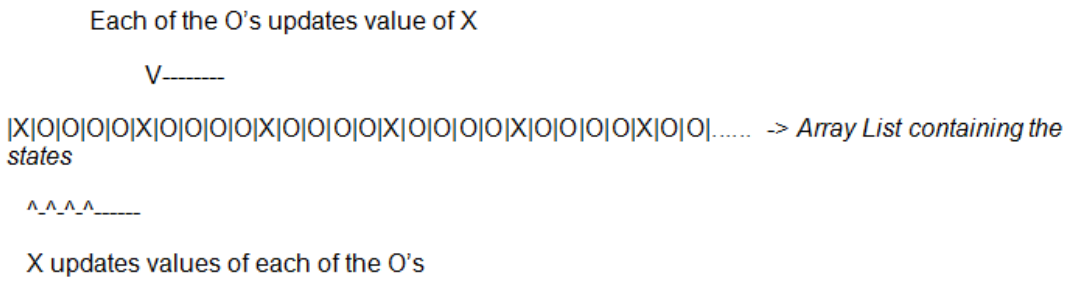


Figure A.3: *X is the state of the game including the action (move) of the player (or adversary, which always starts the game first). O is the state of the game including our learning agent's action (move).*

We focused on storing structures such as $|X|O|O|O|O|$ and not for example $|O|O|X|$ because we always investigate what needs to happen after the adversary has played. When the game starts the adversary plays always first. The array list accepts gradually whole "trees" that consist of a root, that is the state of the adversary (that has just played) and then the alternative states of the learning agent follow (including alternative played moves). In this way when the agent locates a game state that it hasn't seen before and occurred after the adversary played, it directly records it in the list. Then it updates the previous one (that is a learning agent's state along with an action) accordingly, based on the formula which is at the end of Section 2.1.5. The LA's new states are placed next to it.

On the other hand, whenever the agent sees a state that is already recorded in the list then it only updates the previous one by back-propagating its value in the same way as before again. After each of the player's (adversary's) new move (action) the learning agent searches in its array list to locate the next most valuable state in order to act accordingly when it is in greedy mode. When it is in exploration mode, then it just acts at random and if the new state after its recent action does not exist in the list (next to those that are after the player's action) then it inserts it in the appropriate place. It was important for us to know in other words after each player's action (X is the current state of the game including this action) which of the next states (O) that include alternative LA's actions has the current highest value. This is what it selects to play in greedy mode and in this way it learnt how to play optimally and never lose a game. The success of this algorithm allowed us to use a similar, advanced version in the case of the game "Taikun" that we examined in Chapter 3.

A.1.3 Poker

Another game that we considered working on at the beginning of this research was the Texas hold-em poker, as this version is the most popular amongst the various ones of poker. The game offered many elements that were interesting for us, such as its non-cooperative nature and imperfect information, as the player's cards are hidden and the history of the games is not fully observable. Code was implemented in Java for a Texas hold-em version of the game between two players, planning on making one of them a Reinforcement Learning agent. Hence a basic RL algorithm was designed (but it was never tested) and can be found in the Appendix of the thesis (Section A.2.2) along with relevant ideas (Section A.2.2). However due to the fact that the negotiation between the players is not direct, as they do not trade their cards between them and therefore it would not be (linguistically) interesting enough for us, it was soon abandoned.

There was another important reason that made us abandon poker and proceed with the creation of Taikun: the facial characteristics and the body language of the players, that have a very important role in poker, suggested the use of learning techniques that incorporate pattern recognition, visual perception and a high dosage of empirical analysis based on Psychology in order to result to a strong solution. Recent remarkable results in heads-up limit Texas hold'em poker [12] have shown that the game is solvable, by a proposed new algorithm called "CFR". The algorithm uses a regret minimization method which is related to our research area, the Reinforcement Learning. However according to the authors of the above article, with their proposed algorithm the game was *essentially weakly solved*. With the addi-

tional application of the above visual techniques that would take into consideration facial characteristics and body language, blended with theories of Psychology, the solution might become even stronger.

A.2 Algorithms

A.2.1 "Tic-tac-toe": LA's algorithm

PLAYER'S TURN:

if game is not over
Player moves
else
new game

AGENT'S TURN:

if state does not exist
assign state to list
assign value to this state (LOW VALUE if it is one of the losing states or tie)
(if not the 1st state) update the value of the previous state (before the move) based on the new state
(after the move) (LOW/HIGH VALUE if it is one of the winning/losing states) using the formula $V'(s) \leftarrow V(s) + a[V(s') - V(s)]*$
if the game is not over
agent's random move (explore)
assign state to list
assign value to this state (HIGH VALUE if it is one of the winning states)
update the value of the previous state(before the move) based on the new state(after the move) using the formula $V(s) \leftarrow V(s) + a[V(s') - V(s)]*$
// end if game is not over
else (game is over - loss/tie)
new game
// end if state does not exist
else if state does exist
(if not the 1st state) update the value of the previous state(before the move) based on the new state (after the move) (LOW/HIGH VALUE if it is one of the winning/losing states) using the formula $V(s) \leftarrow V(s) + a[V(s') - V(s)]*$
if the game is not over

agent makes move

70% based on state after that which has the next higher value (greedy)

(OR)

30% random move (explore)

if this random move (new state) doesn't exist

assign state to list after the player's move (element), that is the previous state (player's) that we found it exists

assign value to this state

// end if the game is not over

update the value of the previous state(before the move) based on the new state(after the move) (LOW/HIGH VALUE if it is one of the winning/losing states) using the formula $V(s) \leftarrow V(s) + a[V(s') - V(s)]*$

else (if the game is over)

new game

// end else if the state does exist

* The variable a (learning parameter) begins with a value of 0.9 and as long as more games are being played its value gradually decreases to 0 (for the sake of convergence).

A.2.2 Poker: LA's algorithm and notes

ALGORITHM based on Temporal-Difference Learning Method (STOCHASTIC ACTIONS)

GOAL : TO INCREASE THE VALUE OF OUR HAND BY SWAPPING A SPECIFIC NUMBER OF CARDS. WHAT SHOULD THIS NUMBER BE IN EACH OF THE HAND'S INITIAL OCCASIONS?

AGENT'S TURN:

if state does not exist //5 new cards on hand that we haven't had before

-assign state to list

-agent's random action (explore) //change a random number of cards from your initial 5 that aren't in a pair, triplet or quad

-assign state to list exactly after the previous one that we found that it doesn't exist

-update the value of the current state based on the final state (evaluation after the move), as well as on the value of the current hand (valueOfHandCards[0] in code) using the formula $V(s) \leftarrow V(s) + a[V(s') - V(s)]$ (This evaluation state is called $V(s')$ and is $V(s) + 1$ if it has a larger value than the initial or 0 if it has the same)

// end if state does not exist

else if state does exist

-agent makes move

70-based on state after that which has the next higher value (greedy)[1]

(OR)

30-random move (explore)

if this random move (new state) doesn't exist

-assign state to list after the previous state that we just found that it already existed

// end if the game is not over

-update the value of the current state based on the final state(evaluation after the move), as well as on the value of the current hand (valueOfHandCards[0] in code) using the formula $V(s) \leftarrow V(s) + a[V(s') - V(s)]$ (This evaluation state is called $V(s')$ and is $V(s) + 1$ if it has a larger value than the initial or 0 if it has the same)

// end else if the state does exist

Further notes (1st PhD year)

- Each state of the game is my hand of 5 cards
- Actions are the number of cards that I can exchange
- The goal state is a hand with better value
- Thus the goal is to increase the value of my hand
- So the optimal policy would swap a specific number of cards to increase the value of my hand. This number of cards depends on my initial hand (what kind of cards I currently have).
- There is a states' transition between only 2 states in each game. Initial hand, future hand.
- We need to enumerate statistically all the hands with their outcomes before and after each swap to determine how many cards we should change on each case in order to achieve the highest value. This simply can be stated as: We train a number of times (play a number of games) and we store every initial hand, number of cards that have been swapped and their future hands. Each time that our future hand has a higher rank we increase a pointer that

[1]This move is in fact choosing the suitable number of cards to be swapped depending on the hand that has the highest value (afterwards in our list).

indicates how many cards I swapped given my initial hand (state). At the end the learning agent will be able to find each hand (state) and choose the number of cards to exchange based on that counter. In other words, it will be based on the number of cards that I swapped which gave to me a future hand with a higher rank. Value iteration will work here. If my future hand (state) has a higher rank then I will give it the highest reward and then I will update the value of the previous state (hand, including the number of cards that I planned on exchanging).

- Stochastic actions, stochastic rewards (and Q-Learning can be applied).
- After my action of swapping one card there is a ((4 - what I have observed in play)/remaining cards in deck) chance of getting to a desired state (getting the card that I need).
- From the same state all of the actions can lead to a goal state
- Higher future hands will have high rewards, lower future hands will have low rewards and the same will have neutral rewards.

Diagram of the array list:

Current Hand → Current Hand with action (swap 0 cards) → Future Hands → Current Hand with action (swap 1 card) → Future Hands → Current Hand with action (swap 2 cards) → Future Hands → Current Hand with action (swap 3 cards) → Future Hands → Current Hand with action (swap 4 cards) → Future Hands → Current Hand with action (swap 5 cards) → Future Hands

$Q(S, a) \leftarrow (1 - a) * Q(S, a) + a * [r + gamma * maxQ(S', a')]$

For a future hand this will be :

Q(Initial Hand, swap number of cards) ← (1-a)*Q(Initial Hand stored in my list so far, swap number of cards) + a * [immediate reward + gamma * maxQ(Future Hand, no future action)] that in fact is: Q(Initial Hand, swap number of cards) ← (1-a)*Q(Initial Hand stored in my list so far, swap number of cards) + a * immediate reward.

The immediate reward is stochastic and therefore it will be high for a higher hand, low for a lower one.

Similar with the formula from value iteration: $V(s) \leftarrow V(s) + a[V(s') - V(s)]$ where:

$V(s)$ is the Current Hand with action.

$V(s')$ is the Future Hand.

In my array list I can just have each of the states (Current Hands) with their distinct actions followed by a number of outcomes (Future Hands)[2]. My program

[2]The outcomes (future states) don't have to be included in the list as they will grow its size very much for no reason and therefore the algorithm's complexity. Our focus is on the state-action pairs of the current hand anyway.

will search inside this list to locate the current state (Current Hand) and based on all of the values of the states in there (that in fact are state-actions pairs) it will select the one with the highest value. It will be the same recorded state as our game's current state with either the action of swapping 0 cards, 1 card, 2 cards, 3 cards, 4 cards or 5 cards.

A.2.3 "Taikun": LA's algorithm (custom SARSA(0))

if state with id 0 does not exist at all //can be winning state
if game is over //for state 0 because it is indeed a winning state
update value of previous state (state 2)
end game
assign state 0 to list
assign value /id 0 to this state 0
Perform a random action 0
assign this random action 0 to this state 0
<if not the 1st state of the game> update value of previous state (state 2)
if game is over //for state 1 because of action 0
update value of previous state (state 0)
end game
assign new state 1 to the list
assign value/id 1 to the new state 1
update value of previous state (state 0)
listen to adversary's trade and perform random action 1
assign action 1 to state 1
if game is over //for state 2 because of action 1
update value of previous state (state 1)
end game
assign new state 2 to list based on the previous action
assign value/id 2 to this state 2
update previous state (state 1)
//end if state 0 does not exist
else if state with id 0 exists //that means it is definitely not a winning state
if greedy (70%) action //state 0
locate states' 0 action 0 with highest value
perform that action 0
update value of previous state (state 2)
if game is over //for state 1 because of the action 0

update the value of previous state (state 0)
end game
if greedy (70%) //state 1
if this state 1 from the previous action 0 exists
locate states' 1 action 1 with highest value
perform that action 1
update value of the previous state (state 0)
//end if this state 1 from the previous action 0 exists
else //if this state 1 from the previous action 0 does not exist at all
random action 1 generation for state 1
assign this action 1 to state 1
assign state 1 to list exactly after the previous state 0
assign id/value to state 1
update the value of the previous state 0
perform action 1
//end else this state 1 from the previous action 0 does not exist at all
if game is over //for state 2 because of action 1
update the value of previous state (state 1)
end game
if the new state 2 exists
update the value of previous state (state 1)
else //if the new state 2 does not exist
assign state 2 to list (exactly after the previous state 1)
assign value /id 2 to this state 2
update the value of previous state (state 1)
////end if greedy state 1
else // if exploring (30%) state 1
random action 1 generation for state 1
if this state 1 with the new random generated action 1 exists
perform the previous random action 1
update the value of the previous state (state 0)
//end if this state 1 with the new random generated action exists
else //if this state 1 with the new random generated action 1 does not exist
perform the previous random action 1
assign state 1 to list exactly after the previous state 0
assign action 1 to this state 1
assign value /id 1 to this state 1
update the value of previous state (state 0)
//end if this state 1 with the new random generated action does not exist
if game is over //for state 2 because of action 1

update the value of previous state (state 1)
end game
if the new state 2 exists
update the value of previous state (state 1)
else //if the new state 2 does not exist
assign state 2 to list
assign value /id 2 to this state 2
update the value of previous state (state 1)
//end else exploring (30%) state 1
//end if state 0 greedy
else //if exploring (30%) action 0 for state 0
random action 0 generation for state 0
if this state 0 with the new random generated action 0 exists
perform the previous random action 0
update the value of the previous state (state 2)
if game is over //for state 1 because of action 0
update the value of previous state (state 0)
end game
if greedy (70%) //state 1
if this state 1 from the previous action 0 exists
locate state action 1 with highest value
perform that action 1
update the value of previous state (state 0)
//end if this state 1 from the previous action 0 exists
else //this state 1 from the previous action 0 does not exist at all
random action 1 generation for state 1
assign this action 1 to state 1
assign state 1 to list
assign id/value to state 1
update the value of the previous state 0
perform action 1
// end else this state 1 from the previous action 0 does not exist at all
if game is over //for state 2 because of action 1
update the value of previous state (state 1)
end game
if the new state 2 exists
update the value of previous state (state 1)
else //if the new state 2 does not exist
assign state 2 to list
assign value /id 2 to this state

update the value of previous state (state 1)
//end if greedy (70%) for state 1
else //if exploring (30%) action for state 1
random action 1 generation for state 1
if this state 1 with the new random generated action 1 exists
perform action 1
update the value of the previous state (state 0)
// end if this state 1 with the new random generated action exists
else //if this state 1 with the new random generated action 1 does not exist
assign state 1 to list
assign the random action 1 to state 1
assign value /id 1 to this state
perform the random action 1
update the value of previous state (state 0)
//end else this state 1 with the new random generated action does not exist
if game is over //state 2 because of action 1
update the value of previous state (state 1)
end game
if the new state 2 exists
update the value of previous state (state 1)
else //if the new state 2 does not exist
assign state 2 to list
assign value /id 2 to this state
update the value of previous state (state 1)
// end else exploring (30%) action for state 1
// end if this state 0 with the new random generated action exists
else //if this state 0 with the new random generated action does not exist
perform this action 0
assign state 0 to list
assign value /id 0 to this state
assign previous action 0 to this state 0
update the value of previous state (state 2)
if game is over //for state 1 because of action 0
update value of previous state (state 0)
end game
assign new state 1 that was based on the previous random action 0 to the list
assign value/id to the new state 1
listen to adversary's trade and perform random action 1
update value of previous state (state 0)
assign this new random action 1 to state 1

if game is over //for state 2 because of action 1
update value of previous state (state 1)
end game
assign new state 2 to list based on the previous random action 1
assign value/id to this state 2
update previous state (state 1)
//end else this state 0 with the new random generated action does not exist
//end else exploring (30%) action for state 0
//end else state 0 exists

A.2.4 "Taikun": LA's algorithm (SARSA (λ))

Create list with all states actions and initial Q-Values
new game (initialize states of both agent and adversary)
while the adversary's and the agent's current states aren't the goal ones
generate e-greedy behaviour for state 0
if greedy //state 0
locate states' 0 action 0 with highest value
if(round!=1)
update the value of the previous states actions (ALL)
perform that action 0
if game is over //for the new state 1 because of action 0
update the value of the previous states actions (ALL)
end game
//end if game is over for state 1
generate e-greedy behaviour for state 1
if greedy //state 1
locate states' 1 action 1 with highest value
//end if greedy state 1
else //exploring state 1
random action 1 generation for state 1
//end else exploring state 1
update the value of the previous states actions (ALL)
perform that action 1
if game is over //for the new state 2 because of action 1
update the value of the previous states actions (ALL)
end game
//end if game is over for state 2

//end if greedy state 0
else //if exploring state 0
random action 0 generation for state 0
if(round!=1)
update the value of the previous states actions (ALL)
perform that action 0
if game is over //for the new state 1 because of action 0
update the value of the previous states actions (ALL)
end game
//end if game is over for the new state 1
generate e-greedy behaviour
if greedy //state 1
locate states' 1 action 1 with highest value
//end if greedy state 1
else //exploring state 1
random action 1 generation for state 1
//end else exploring state 1
update the value of the previous states actions (ALL)
perform that action 1
if game is over //for the new state 2 because of action 1
update the value of the previous states actions (ALL)
end game
//end if game is over for state 2
//end else if exploring state 0
//end while the adversary's and the agent's current states aren't the goal ones

NOTES: STATES 2 (GAME'S UPDATE) ARE NOT INCLUDED IN THE LIST AT ALL, STATE 1 + ACTION 1 RESULT IN STATE 0 IN THIS IMPLEMENTATION