ABSTRACT

EGER, MARKUS. Intentional Agents for Doxastic Games. (Under the direction of Chris Martens).

Playing games in which communication is a key part of game play with or against human players is a challenge for AI research. Agents that should play such games need to reason beyond purely maximizing their score to account for how the human players expect to communicate. One important aspect of this expectation is the notion of intentionality, or goal-directedness, which was defined as choice plus commitment by Cohen and Levesque. They also note that intentionality is tightly linked with reasoning about beliefs, since, in order to determine how to achieve a goal, an agent has to reason about what they know and how their actions might affect the world. In this document, I first present AI agents that play the cooperative card game Hanabi with human players to study how the agents' goal-directed behavior is understood by them. These agents first choose what they want to happen in the game, and then commit to that goal by performing the corresponding action. However, the goals used by the agents are all focused on short-term effects on the game state, and therefore no longer-term commitment strategy is needed. On the basis of the performance of these agents, though, I then describe a general agent design with a model of intentionality that utilizes a theory of mind to play games with communicative actions. The agents use Dynamic Epistemic Logic as a representation of the beliefs of other players to be able to plan their communicative actions as well as interpret communication they receives from other players. As the second application domain I present features deception and lying, I developed a notion of belief quality to account for the uncertainty of information that is provided by other players. The agents use this measure of belief quality to choose a goal corresponding to what they believe that they can convince other players of. Belief quality as a measure of how well a goal can be achieved is also used by the agents to determine how long they should stay committed to a plan, and when they should adopt a new one. I then present how these agents can play the social deduction game One Night Ultimate Werewolf and the results of two experiments for this application. In the first, different agent types played the game against each other, which revealed that different levels of commitment have an effect on the agents win rate. In the second experiment, human players played with the agents, and were asked to rate their experience. As hypothesized, how committed agents were to their goals had an effect on the participants' perception of the agents. Finally, I will shortly discuss how the insights gained from my work can foster future work, by applying the same principles to other domain.
Intentional Agents for Doxastic Games

by
Markus Eger

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2018

APPROVED BY:

R. Michael Young
Arnav Jhala

Nicholas Taylor
Murray Campbell
External Member

Chris Martens
Chair of Advisory Committee
DEDICATION

To Open Science,
because knowledge wants to be free.
BIOGRAPHY

Markus was born and grew up in the small town of Eggersdorf bei Graz in Austria. As an avid gamer, he was interested in the implementation of games as soon as he learned how to code. When he got his bachelor’s degree in Computer Science from Graz University of Technology his capstone project was the implementation of a new AI component for autonomous soccer robots. He followed this up with a master’s thesis about the procedural generation of 3D structures using shape grammars at the same university, before moving to North Carolina in 2013 to get his PhD at NC State. Apart from games, Markus enjoys cooking, hiking, cats and sloths.
ACKNOWLEDGEMENTS

I would like to thank my advisor, Chris Martens, for all the discussions, support, editing and advice she has given me, without which this dissertation would not have been possible. Before I worked with Chris, I had a different advisor, Michael Young, until his departure from NC State. He was also great mentor in my early years as a graduate student, and had many stories to tell, and I am glad he could be part of my committee. Additional thanks go to my other committee members, Arnab Jhala, who always had an open ear for me, Nick Taylor, who provided valuable insights, and Murray Campbell, who is an inspiration for my work.

Beyond my committee, there are other faculty members that I would like to thank: Blair Sullivan, who always had an open door for a chat and general advice for me, Jon Doyle, who it was always fascinating to have discussions with, and Tiffany Barnes, who TAing for was a joy. The staff at NC State was also most supportive and always on top of things, so I would like to thank Kathy Luca, Linda Honeycutt and Jami Boylan for their help in my fights against bureaucracy. However, my journey did not start at NC State, and several faculty members at Graz University of Technology also have my gratitude: K.C. Posch, who was the most enthusiastic teacher I ever had, Sven Havemann, who was an inspiring teacher, and great advisor for my master’s thesis, Oswin Aichholzer, who introduced me to Hanabi, and Peter Lipp, who gave his TAs a lot of freedom, which inspired me to improve how I teach.

This five-plus year journey would not have been possible without the support and love from my parents Erika and Wolfgang, my brother Jürgen, and all of my extended family, who always welcomed me back home warmly whenever I could make it. Danke für Alles! I also want to thank my friends back home in Austria: Herv, Gudrun, Kathi, Daniel, Namuel, and Johanna, for many game nights and barbecues, Andi, and Mazda, for banter and hijinks since middle school, and my oldest, best friend René. My visits were few and far between, but all of them always made sure that I know why I miss them. And Berni is crazy.

When I came to the US, I did not know anyone, but I met many awesome people along the way: Justus, Rogelio, Stephen and Julio, I thank you for the warm welcome you gave me to the lab, and the role models you were, Camille, premier painter of pheasants, and Trey, a fellow Waffle House connoisseur. Marisa, and Sid, who is the first friend I made in this country, and the worst player of Wizard I have ever met. Mike, Shannon, and Prof. Dr. Thomas who were always up for games, and know where the mans are. Jen, who is the only other person to ever get 24\frac{1}{2} points in Hanabi (one day ...). A special thank you goes to brew master Sean, who does not panic even if his mead does not bubble, and who makes the best birthday gifts. Ana, Leo, German, Erika, Hector and Etna, who helped me improve my Spanish during several joyous gatherings, even if it did not always seem like it. The entirety of
the WYLDKATZ, and trivia host Lee, who does not tire of my complaints, even though Istanbul is definitely not a capital. My lab mates, former and current: Helen, best lunch buddy, fellow $\tau$-ist (I think) and drawer of fine art. Colin, who is the only one that will walk through the snow with me in the middle of Utah. Brandon, Alex, Chinmaya and Owais, one of whom (probably) writes random crosswords and riddles on sticky notes and leaves them in the lab for people to solve. A late mention also goes to Sasha, who is exhaustingly extroverted, and always up for and to something. And spiders. The numerous other people I have met along the way, and who have contributed to all the good times I had here, that I will always remember fondly. And maybe y’all can visit Costa Rica some day!

Speaking of Costa Rica, my last, but by no means least, thanks go to my wonderful girlfriend Marcela, who showed me the way by graduating first, and then showed me the amazing natural splendor of her home country. Thank you for being there for me! I am looking forward to spending my life with you wherever it takes us. Te quiero! I also want to extend thanks to Marce’s family, Saray, Amanda, George and Edgar, who make me feel as part of the family, too! Muchas gracias por todo!

This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.
# TABLE OF CONTENTS

**LIST OF TABLES** ............................................................... ix

**LIST OF FIGURES** .................................................................. x

**Chapter 1 INTRODUCTION** .................................................. 1
  1.1 Motivation and Scope ................................................. 2
  1.2 Research Questions and Thesis Statement .................. 2
  1.3 Typographical Conventions ......................................... 4
  1.4 Open Science ............................................................. 5
  1.5 Outline ...................................................................... 6

**Chapter 2 BACKGROUND AND RELATED WORK** .................. 8
  2.1 Doxastic Games .......................................................... 8
     2.1.1 Hanabi ............................................................. 9
     2.1.2 One Night Ultimate Werewolf .......................... 12
  2.2 Intentionality and Theory of Mind ............................. 15
  2.3 Dynamic Epistemic Logic .......................................... 19
     2.3.1 Syntax .......................................................... 20
     2.3.2 Formula Interpretation ...................................... 21
     2.3.3 Actions and Action Application ......................... 23
     2.3.4 Practical Considerations .................................. 26

**Chapter 3 HANABI** .......................................................... 28
  3.1 An Intentional AI for Hanabi ................................... 30
     3.1.1 Knowledge Representation ............................... 30
     3.1.2 Giving Hints .................................................. 31
     3.1.3 Discarding Cards ............................................. 33
     3.1.4 Interpreting Hints ........................................... 34
  3.2 Results .................................................................... 35

**Chapter 4 OSTARI** .......................................................... 40
  4.1 The Ostari System .................................................... 41
     4.1.1 Syntax .......................................................... 42
     4.1.2 Compilation ................................................... 43
     4.1.3 Execution Environment .................................... 46
     4.1.4 Belief Quality ................................................ 49
     4.1.5 Optimizations ................................................ 51
     4.1.6 Planning ......................................................... 58
     4.1.7 Unreachable Goals ........................................... 62
  4.2 Intentional Agents for Doxastic Games ..................... 63
     4.2.1 Agent Design ................................................ 65
     4.2.2 Interpreting Received Information .................... 66
     4.2.3 Intentions ....................................................... 67
7.2.3 Epistemic States .................................................. 133
7.2.4 Application Domains ............................................. 133
7.2.5 Communication Modes .......................................... 134

BIBLIOGRAPHY ................................................................. 136

APPENDICES ................................................................. 142
Appendix A Ostari ......................................................... 143
   A.1 Ostari Input File ................................................. 143
   A.2 Intentional Agent Deliberation to Generate Next Move .......... 154
Appendix B One Night Ultimate Werewolf ................................ 155
   B.1 Simulation Results ............................................... 155
   B.2 Participant Survey Responses ................................... 158
   B.3 Participant Comments ........................................... 171
      B.3.1 Balanced Agent ............................................ 171
      B.3.2 Capricious agent ......................................... 174
      B.3.3 Fanatical agent ............................................ 175
LIST OF TABLES

Table 3.1 Agents’ knowledge state for a card about which no information is known .......... 31
Table 3.2 Agents’ knowledge state for a card after they have been told that the card is red, and a red 1 and 2 have been played ............................ 31
Table 3.3 Average scores (with sample standard deviation) after 10000 games for each AI pair .......................................................... 35

Table 4.1 Translation of Ostari macros to Baltag actions. Lower case letters are placeholders for properties, conditions or statements, with the upper case equivalents in the right column referring to the compiled version of that property, condition or statement ............................................ 45
Table 4.2 Values the epistemic planning heuristic takes for different weighted qualities and plan lengths, with $\beta = 1$ .................................................. 63
Table 4.3 World weights only considering 3 players after player $a$ claimed to be the Seer, player $b$ claimed to be a Villager and player $c$ also claimed to be the Seer .............................................. 67

Table B.1 Simulation results for a balanced game, in which the Werewolf card can not be exchanged for any other card. As expected does the level of commitment not have any significant effect on agent performance .............................................. 156
Table B.2 Simulation results for a balanced game, in which the Werewolf card can be exchanged for another card. In this case, the level of commitment of the agents does have an effect on agent performance .............................................. 156
Table B.3 Simulation results for a game which heavily favors the non-Werewolf players. The level of commitment does have an effect on agent performance, because it is very likely that cards are switched around, making being overly committed to any particular plan a liability .......................................................... 157
Table B.4 Simulation results for a game which heavily favors the non-Werewolf players, comparing fanatical agents to balanced agents with $\alpha = 0.66$, with different certainty thresholds .................................................. 157
Table B.5 $\chi^2$ statistic values, $p$-values and means for player ratings for how reactive the agents’ behavior was .................................................. 159
Table B.6 $\chi^2$ statistic values, $p$-values and means for player ratings for how much sense the agents’ statements made .................................................. 161
Table B.7 $\chi^2$ statistic values, $p$-values and means for how many statements by an AI agent the participants believed .................................................. 163
Table B.8 $\chi^2$ statistic values, $p$-values and means for how many statements by an AI agent the participants rated as making sense .................................................. 165
Table B.9 $\chi^2$ statistic values, $p$-values and means for player ratings for how well the agents played .................................................. 167
Table B.10 $\chi^2$ statistic values, $p$-values and means for player ratings for how much fun it was to play with the agents .................................................. 169
LIST OF FIGURES

Figure 2.1 A typical Hanabi board during game play. In an actual game, players would hold their hands of cards facing away from themselves. Source: http://www.clevermovegames.com .............................. 10
Figure 2.2 A possible starting configuration for One Night Ultimate Werewolf. The roles are revealed for demonstration purposes only, in an actual game they would be secret. ................................. 14
Figure 2.3 A possible worlds model for a possible epistemic state in a simple card game: Agent \( x \) holds two cards, and knows that they are a club and a spade, while agent \( y \) only knows that the cards could be clubs or spades, but not which is which. Agent \( y \) also knows that agent \( x \) knows which cards they have. ................................. 20
Figure 2.4 A graphical representation of the action \( \text{flip} p \cdot (\text{flip} p + \text{flip} q)^x \). Each node represents an action token containing the content change, and the edges represent the appearance maps for each agent corresponding to the label on the edge. ................................. 25
Figure 2.5 Resulting epistemic state after applying \( (? \text{left}(\text{clubs}))^y \) to the state from Figure 2.3 ................................. 26
Figure 3.1 Algorithm to determine which hint to give to the cooperator, if any ............................. 32
Figure 3.2 A screenshot of my implementation of Hanabi during game play ............................. 36
Figure 3.3 Score distribution after the first game played by each participant, grouped by which agent they played with. ............................. 37
Figure 3.4 Score distribution after all games played by the participants, grouped by which agent they played with. ............................. 39
Figure 4.1 Grammar for action definitions in Ostari ................................. 44
Figure 4.2 A representation of the initial beliefs of the logicians in the joke. Each node represents a world in which the logicians listed in the label of the node want a beer. Note that each world is additionally also considered to connected to itself for all three logicians in this example. ................................. 72
Figure 5.1 Win rate in a typical game for various agent types for different numbers of states that are expanded before the planning process is terminated. ............................. 101
Figure 5.2 Anna’s win rate based on her level of commitment \( \alpha \) against random and capricious Citizens ................................. 103
Figure 5.3 Win rate in an imbalanced game for different certainty thresholds. ............................. 105
Figure 6.1 A system diagram of the deployment of One Night Ultimate Werewolf for the study. ................................. 112
Figure 6.2 A screenshot of the Unity implementation of One Night Ultimate Werewolf (Art for the roles kindly provided by Hui-Yin “Helen” Wu). ................................. 113
Figure 6.3  A summary of the responses to the six survey questions for each of the three treatments. Each question is identified by a descriptor, where the last letter indicates whether the game was played against (f)anatical, (b)alanced or (c)apricious agents. ................................................................. 118
Figure 6.4  Distributions for the number of statements made by an AI agent that participants rated as making sense. .......................... 120
Figure 6.5  Distribution of the individual statements made by an AI agent that participants rated as making sense. ............................. 121
Figure 6.6  Distributions for the number of statements made by an AI agent that participants stated that they believed. ............................. 122
Figure 6.7  Player ratings for how well the different AI agents played, as ranked on a 5 point Likert scale. .......................... 123
Figure 6.8  Player ratings for how much fun it was to play with the different AI agents, among players with moderate or high board game experience. .......................... 126
Figure B.1  Player ratings for how reactive the agents’ behavior was, as ranked on a 5 point Likert scale .......................... 158
Figure B.2  Player ratings for how much sense the agents’ statements made, as ranked on a 5 point Likert scale .......................... 160
Figure B.3  Distribution of the individual statements made by an AI agent that participants rated as being believed. .......................... 162
Figure B.4  Distribution of the individual statements made by an AI agent that participants rated as making sense. .......................... 164
Figure B.5  Player ratings for how well the different AI agents played, as ranked on a 5 point Likert scale .......................... 166
Figure B.6  Player ratings for how much fun it was to play with the AI agents, as ranked on a 5 point Likert scale .......................... 168
CHAPTER

1

INTRODUCTION

The driving force behind advances in AI often comes from application of AI to games, such as chess [Cam02], Go [Sil16], Jeopardy [Fer10], or the video game Defense of the Ancients 2 (DotA 2) [Ope18]. However, a lingering difficulty lies with conveying the AI agent’s reasoning to humans. For example, during the televised Jeopardy game, IBM Watson famously answered Toronto to a question from the category U.S. cities, baffling host and audience alike, even though its internal reasoning was sound and it gave a very low confidence score for the answer. Similarly, while OpenAI’s DotA 2 agents play competently, and beat humans in one exhibition match\(^1\), they still perform erratically in the face of hidden information, and often exhibit unexpected behavior. Especially behavior that goes against human expectations makes it hard to analyze why exactly the agents ended up losing another exhibition match, but it is also a limitation for scenarios in which the AI agents should play on the same team as human players. To improve this interaction between AI agents and human players my work aims to make the AI agent’s actions more understandable for human players. In particular, I am looking at agents that employ intentionality, or goal directed behavior, to more closely align with what human players expect. Because I am interested in human player’s understanding and perception of the AI agents’ actions, I will focus on agents that use intentions to plan and commit to their communicative actions.

\(^1\)For details, read Mike Cook’s summary, that does not require detailed knowledge of the game mechanics, here: http://www.gamesbyangelina.org/2018/08/openai-dota-2-game-is-hard/
1.1 Motivation and Scope

Games provide an ideal test environment for the ideas I am developing for my thesis, because they represent self-contained rule systems that can be formalized easily, and provide built-in performance measures in the form of their scoring mechanics. I will be using the term *doxastic games* to refer to games where communication and belief manipulation are a key part of game play. While my research is not limited to such games, they provide a more focused test environment than games in general. The games I am describing here have also already been studied from a variety of angles, providing research to draw from and results to compare to.

To achieve my goal of having agents that utilize intentional behavior to communicate well with humans, it is necessary for the agent to have a model of what the human, and other agents, believes, including nested beliefs, i.e. what the human believes that the agent believes, etc, called a *theory of mind*. I am using *epistemic logic*, the logic of knowledge and belief, to be able to represent such scenarios.

In this document, I will present an approach to intentional agents that can play games that involve communication in a way that makes sense to human players. In particular, I will focus on the effect an agent’s commitment to a goal, which is a key feature of intentional behavior, has on player perception. In the next section I will discuss the research questions that inspired this research.

1.2 Research Questions and Thesis Statement

The work presented in this document is focused on intentional agents and how a theory of mind can serve as the basis for intentional behavior that can ultimately be used to communicate more effectively with humans in the context of doxastic games. To investigate this relationship between beliefs and intentions, I want to answer several research questions surrounding intentionality and epistemic logic as part of my work. The thesis I am going to prove in this document is:

*Agents can use a theory of mind to exhibit intentional behavior in a way that is understood by humans. This approach can be applied to doxastic games in a way that is believable and understandable for human players.*

The key idea here is that intentionality has an effect on what players consider believable, and on their perception of the AI agents’ actions. As a starting point, I looked at a specialized approach that is geared towards the cooperative card game Hanabi. This work establishes that intentional behavior has an advantage over prior work when playing with human players, and thus answers the following research question:
RQ1: Does an intentional agent perform better when playing with human players in a cooperative doxastic game?

However, while the agents in the Hanabi experiment pursued goals, they did not capture the entirety of what constitutes intentional behavior which also includes agents being committed to achieving their goals. Additionally, the implementation was specific to the game, and could not be easily adapted for other domains. In order to be able to apply one common agent design to a wide range of domains, I developed a system called Ostari that provides a programming language and execution environment for games involving epistemic actions. This system addresses the following research question:

RQ2: How can games and other domains that involve communicative actions be encoded in a high-level language?

This does not mean that agents should never give up pursuing their goals, just that this has to be reasoned about as well. The circumstances under which agents should reconsider their intentions still remains to be researched, though. To test if and what effect different levels of commitment have on agent performance in a doxastic game, I performed an experiment on the game One Night Ultimate Werewolf using simulated agents. This experiment serves to answer this research question:

RQ3: What effect does the degree of commitment to an agent’s intentions have on that agent’s performance in a competitive doxastic game?

However, because I am not only interested in raw performance in games, but also in the effect intentional behavior has on human players, I also tested the effect of intentional agents that plan their actions when playing with human players. The results from this experiment serve to answer the following research question:

RQ4: What effect does the level of commitment have on player enjoyment and agent believability when playing a competitive doxastic game?

All of these questions together serve to answer the bigger question of how using a theory of mind enables intentionally acting agents in environments where they communicate with humans, and if this can be used to communicate in a way that is natural and believable. However, I believe that the techniques presented in this document can also be applied to environments outside of games, and I will briefly outline how they could be used to address the following
research question in future work:

RQ5: Does intentional behavior improve hint-giving capabilities in learning games or support environments?

### 1.3 Typographical Conventions

For the remainder of this document, the following typographical and naming conventions will be used unless otherwise noted:

- Formulas, especially those in Dynamic Epistemic Logic, will be written in math notation, e.g. $\square_x t(AceOfSpades)$.

- Code will be written in typewriter font, e.g. `tell (x): top() == AceOfSpades` or appear as a listing.

- Sequences will be written in standard mathematical notation, e.g. $(a_i)_i$, where the index set of the index variable will not be stated unless necessary.

- Similarly, indexed sets use standard mathematical notation, e.g. $\{a_x\}_x$, and the index set will also usually be omitted if clear from the context.

- The notation $\cdot_f$ is used to refer to a function that is designated by the subscript $f$. An application of this function to a value $k$ would be written as $k_f$.

- Arbitrary formulas will be designated with Greek letters, e.g. $\phi, \psi$.

- Actions will be designated with lower case letters from the start of the alphabet, $a, b, c$.

- Agents will be referred to using lower case letters from the end of the alphabet, $x, y, z$.

- Generic predicates will use the letters $p, q, r$.

- Worlds, in the possible worlds sense, will use lower case $w$, when necessary with subscripts to indicate which world among several possible worlds is being referred to.

- Sets, in particular those of states or worlds, will be represented by upper case letters such as $S$ and $W$.

- Epistemic states will use math calligraphy, such as $\mathcal{M}$

Note that these conventions are used in explanations of general concepts. In any specific scenario, names of predicates, agents, etc. will be chosen to align with the domain in question.
1.4 Open Science

While not specific to the work presented in this document, I want to use this space to discuss the importance I place in Open Science, and the steps I took to make my research openly accessible. Open Science, as a movement, aims to make scientific work more readily available to the public. While several funding agencies have policies that the results from research funded by public money have to be made public, Open Science aims to make the entire process available to interested parties [Gra10]. This includes not just other researchers, but also the general public. This is done through several different means:

- **Open Access** means that publications are not hidden behind paywalls, or have restrictive licensing terms that make them hard to access or use for future work [Sub07]. Since publications are the primary medium researchers use to communicate their methods and results, this allows a broader audience access to the work, making it easier for more people to potentially build on it. Fortunately, the main conferences I publish my work at, AIIDE, and CIG provide free pdf copies of all papers on their websites.

- **Open Data** refers to the publication of the data that was used to obtain the results in order to allow others to reproduce the results, or gain related or unrelated insights from the same data [MR08]. It has been argued that this leads to better scientific progress overall, because data constitutes the backbone of science, and open access to it helps foster further research [Mol11]. There are some challenges with publishing data, especially when it is collected from human subjects, because participant confidentiality is also critical. In both of my human subject studies I provided participants to opt into having the data recorded from them to be published in a public repository. In order to encourage participants to use this option, I kept the collection of demographic information to a minimum, to make sure no data point can be linked back to an actual person, and most participants ended up choosing to have their data published. The result is that, for example, for the Hanabi study presented in chapter 3, over 1000 game logs from games played with actual human players are openly available ².

- **Open Source**, in the context of Open Science, is the practice of releasing the code used to obtain research results publicly and with a permissive license [Alt05]. For researchers that want to build on, improve or simply compare their work with other researchers’, this saves them from having to duplicate the efforts that were already undertaken. Furthermore, despite our best efforts, descriptions of approaches in papers may lack subtle details that make a reimplementation difficult or behave differently in edge cases that were not covered

²https://github.com/yawgmoth/HanabiData
in the publication. In an effort to make my work not only as accessible as possible, but also make its availability more visible, I have, in addition to releasing my code under a permissive open source license, also submitted the code artifacts to appropriate venues. For the study on Hanabi this was as a technical demo at FDG 2017 [Ege17a], while the implementation of the One Night Ultimate Werewolf agents was submitted as an artifact to the Artifact Evaluation Track at AIIDE 2018. My implementation of Hanabi, being more accessible to someone without a background in Dynamic Epistemic Logic, provides a framework that makes it very easy to define new Hanabi agents for anyone interested in it, and comes with the entire setup for the study.

I strongly believe that opening up access to scientific progress helps fostering further research, but it also has benefits that go beyond helping the established scientific community. Particularly in computer science, hobbyists and tinkerers can integrate cutting-edge research into their own work, leading to interesting new insights and applications, and thus heightened interest and exposure for research. As mentioned above, the scientific community with which I share my work is also working towards these goals. The conferences AIIDE and CIG both provide free access to their papers, with the Experimental AI in Games workshop that is traditionally held at AIIDE even providing video recordings of the talks. AIIDE has also started an Artifact Evaluation Track to encourage authors of accepted papers to share their code artifacts. A new textbook on AI in games, written by Georgios Yannakakis and Julian Togelius, is also available for free in pdf format [Yan18]. I understand that not all funding agencies are as permissive with the rights to research done with their funding, and that not every scholar is in the position to be able to share their work as freely, but I also believe that the knowledge obtained through the scientific process should not be a privilege of those at prestigious universities, or richer countries, and I am pleased that the community I am a part of shares this vision.

1.5 Outline

In this document I will present answers to research questions 1 to 4 outlined above, and briefly discuss how research question 5 could be approached in future work. To frame this work, chapter 2 provides an overview of prior work, especially focusing on intentional agents, and the particular games I am treating. I will also provide a description of the most important aspects of the particular flavor of Dynamic Epistemic Logic that I use in the rest of my thesis. Chapter 3 will describe my initial work on agents that play the game Hanabi in a way that is more understandable for human players when compared to prior work. This includes a description of the design of the agent, as well as the experiment setup used to validate my work, and the results obtained from the experiment. To build on the success with the specialized Hanabi agents,
I set out to build a system that allows the definition of arbitrary doxastic games in a concise manner. The result of this work is a system called Ostari which is presented in chapter 4. Ostari is basically a programming language for actions that manipulate agent beliefs, which can be compiled to Dynamic Epistemic Logic. It also provides an execution environment with several performance enhancements over the compiled version. Finally, to enable game play, Ostari also contains components to define games based on the epistemic actions, and a planning based approach for agents to play these games. Chapter 5 then describes how the game One Night Ultimate Werewolf can be implemented using Ostari, including goals for the agents to pursue, and presents how different Ostari based agents perform against each other. The particular focus of this chapter lies in the difference between agents with different levels of commitments to the plans they come up with to achieve their goals. Following that, chapter 6 describes the adaptations that were made to the agents to improve them for play with human players, the setup for the human subject experiment I performed, and the results and insights from this experiment. This chapter serves to show that agents with different commitment strategies are perceived differently by human players. Finally, chapter 7 will serve as a summary and conclusion of the work, and present opportunities for future extensions and improvements.
Before I describe my approach for intentional agents and their application to doxastic games, I will give an overview over existing work. I will start with doxastic games, including a definition for the term, and the two sample games that constitute the focus of my work. Because I will treat these games in more detail I will explain their rules, but also report on findings by other researchers. I will then provide a selection of prior work on intentionality, especially on logical formulations thereof. Because this is a topic of great interest to many researchers, from philosophy, to psychology to computer science, I can only discuss the most directly relevant work. Because these logical formulations typically include a model of agents’ beliefs, I will also briefly talk about related work on that topic. The final part of this overview over related work is a discussion of a particular logic for belief modeling, Dynamic Epistemic Logic. Because my work makes heavy use of one particular flavor of Dynamic Epistemic Logic, I will describe it in detail.

2.1 Doxastic Games

As already mentioned, games provide a constrained enough environment to be formalized easily and come with built-in performance measures, while still being an interesting and generalizable application area. However, not all games are created equally, and for the purpose of this thesis, I will focus on what I call doxastic games. I coined this term to refer to games in which the communication and manipulation of players’ beliefs is an integral part of game play, and therefore
defined in the rules. For example, while an agent playing a game like poker might benefit from being able to reason about or manipulate other players’ beliefs, there is nothing intrinsic to the rules of poker that defines how that would be done. In contrast, some cooperative games, like Hanabi, which is described below, actually require the communication between players to follow strict rules. There are also competitive games that require players to exchange information, often permitting them to lie. Because my thesis is fundamentally about communication between AI agents and human players, I will be focusing on this subset of games. This does not constitute a limitation of my approach, but merely provides a more focused demonstration of its merits, and also helps with making attribution of effects to features of the agents clearer.

2.1.1 Hanabi

Hanabi [Bau10] is an award-winning [Spi13] cooperative card game in which players cooperate to build fireworks represented by cards in five colors with ranks from 1 to 5. Unlike in most other card games, players hold the cards in their hand facing away from themselves, such that every other player sees the cards, but they themselves do not. Cards are played on the table in five stacks, one for each color. These stacks start out empty, and the goal is to play the cards of each color in ascending order on the corresponding stack, starting with the 1 of each color. Figure 2.1 shows a game state from a typical game, with the five stacks in play, hint tokens, draw pile and discarded cards.

Game play proceeds in turns, and on each turn a player has to perform one of three possible actions:

- **Play a card.** To perform this action, the player chooses a card from their hand and places it face up on the table. If the stack corresponding to its color has its immediate predecessor on top (or is empty, in case the played card is a 1), it is placed as the new top card of that stack. Otherwise it is removed from the game and a mistake is noted.

- **Give a hint to another player.** This action can be used to give information to other players, either to tell them about all cards in their hand that have a particular color, or a particular rank. For example, a player may tell another player which cards in the second player’s hand are red, but they have to point out all of them at once. Hints come at the expense of a hint-token, of which there are initially and maximally 8 available.

- **Discard a card.** This is done by a player choosing a card from their hand, and removing it from the game face up. Doing so recovers one hint token, up to the maximum of 8.

After playing or discarding a card, players draw a new card to replace the one they just played or discarded. The game ends when all cards were drawn, followed by one extra turn for each
Figure 2.1 A typical Hanabi board during game play. In an actual game, players would hold their hands of cards facing away from themselves.
Source: http://www.clevermovegames.com
player, or when the third mistake is made by the players collectively. The score obtained by the players is equal to the number of cards they played, for a maximum of 25 points, if all 5 cards in each of the five colors were played in the right order.

Strategically the game poses the challenge of when to perform which of the three actions, given that players will likely always have incomplete information about the cards in their hand. Playing cards poses the risk of making a mistake, whereas hints are a limited resource and must be given with care. Additionally, there are only three copies of the 1s in each color, two of each 2, 3 and 4 and only one copy of each 5, making discarding also a risky action. If, for example, both 2s of a color are discarded, none of the 3, 4 or 5 in that color can be played anymore either, because they would require the 2 to be played first. When human players play this game, they usually discover that the cooperative nature of the game allows them to draw certain conclusions from hints they are given. For example, in a four player game, if the first player tells the second player that they have a 1, the second player is guaranteed to be able to play that card, because all 1s are playable at the beginning of the game. When the third player then tells the fourth player that they also have a 1, the fourth player can not, strictly from the hint, deduce that that card is playable. However, because the game is cooperative, and assuming reasonably rational players, the third player would not have given that particular hint if it was not helpful in some way. This concept of “helpfulness” is what I will develop into intentionality in chapter 3.

Hanabi is an interesting application for AI research [Wil16], because it not only involves reasoning about partially observable states, but also requires effective communication between players to coordinate their behavior by using only the hint actions to communicate. Because of this, there has been a significant amount of prior work on the game, but most of it focused on improving the performance of agents that play with other agents of the same type. For example, Cox et al. use strategies for hat-guessing games\(^1\) to define a protocol for the agents to follow [Cox15]. Using such a protocol, they encode information into a single hint such that, in a 5 player game, after one agent gives a hint, all 4 other players know what they should do on their turn. This results in agents that perform extremely well, reaching the perfect score of 25 in about 75\% of all games. This was further improved to reaching the perfect score in 90\% of the games by Bouzy [Bou17] by using a tree search approach to determine what each player should do with their cards instead of having a fixed priority list. However, these hat-guessing game-based strategies involve many logical and arithmetical operations that are hard to carry out for a human player, and not particularly intuitive. Additionally, any player not following the exact same strategy, or making a simple mistake during the calculations, will cause massive miscommunication. In summary, while this strategy performs well when played with AI agents,

---

\(^1\)These are logic puzzles, in which \(n\) agents each have a hat of \(k\) possible colors, with some limitations on which hats they can see, e.g. agents can only see other agents’ hats, but not their own. The goal for each agent is typically to deduce the color of their own hat [But09].
it is unsuitable for human players.

Other researchers have focused more on the strategies followed by actual human players. Van den Bergh et al. analyzed the maximum achievable score of different initial configurations of the deck and their effect on the outcome using several different strategies based on human behavior [Ber16]. While their strategies are based on typical human strategies, including a step where the agent will play a card if they are “certain enough” that it is playable, their main focus was on mathematically determining the threshold for which probabilities constitute “certain enough”. Regarding difficulty, Baffier et al. have shown that determining the optimal action in a generalized version of Hanabi is NP-complete even if players look at their own cards [Baf16]. Walton et al. compare several hand-crafted strategies based on expert knowledge, as well as Monte Carlo and tree search techniques with each other [WR17]. They, too, focus on the case where AI agents are playing with each other, though they provide different combinations of AI agents. This work was the basis for the Hanabi AI competition at CIG 2018 ², which consisted of a mirror mode, in which each AI agent played with copies of itself, as well as a mixed mode, in which all entries played with other entries. Canaan et al. [Can18] described their approach of how to learn a rule set to play Hanabi well in this competition in mixed mode.

Hirotaka Osawa [Osa15], finally, investigated the differences between several similar agents in the 2 player version of the game. This work is interesting for a variety of reasons, but for the purpose of my work the most interesting results was that having a model of the other player’s knowledge about their cards greatly increases the score that is obtained by the agents. He calls this version of his agent the Outer State Player, because it takes the knowledge state of the other player into account. He also presents an agent that, in addition to deciding which hints to give, tries to deduce the meaning of hints that it receives, which he calls the Self Recognition Player, because it tries to recognize in which situation it would have given the hint it received itself. Because these agents align closely with the model I am using, they serve as the main basis for my work on Hanabi that is presented in chapter 3.

2.1.2 One Night Ultimate Werewolf

One Night Ultimate Werewolf [Als14] is a social deduction game in which players are secretly assigned a role and then have to deduce the roles of other players. It derives its name and general game play idea from a previous party game called Werewolf or Mafia ³, but provides more streamlined rules by condensing the game into a single round consisting of – as the name implies – one night phase followed by one day phase and in addition to requiring discussion among players also has some roles that can physically change the game state, including faction assignment. These physical changes provide an additional challenge for the players, because while

²https://comp.fossgalaxy.com/competitions/t/11
players can look at their role cards at the beginning of the game, these cards can be switched around during game play. At the end of the game a player’s win-condition is tied to the role indicated on their card, even if that card changed during game play without their knowledge. On the other hand, this provides the players that have roles that can switch cards with information that other players do not have, making the deduction process easier for them.

In each game, there will be role cards equal to the number of players plus three, which means not all roles that exist will also be present in the game. Each player is dealt a role card that they can look at immediately, with the remaining three cards being put in the center, where some roles may interact with them. Figure 2.2 shows a possible initial configuration in a game with five players. The game starts with the night phase, during which all special actions are resolved in a predetermined order. Players play the night phase according to the role card that they were dealt initially. A partial list of roles that exist in the game in the order they can perform their action during the night phase is:

- **Werewolf**: Werewolves will be informed during the night who the other Werewolves are, which means that they have a knowledge advantage over most other roles. Each game will have a number of Werewolves equal to about one third of the number of players.

- **Masons**: Like the Werewolves, the Masons will be informed during the night who the other Masons are to gain a knowledge advantage. They have no other special function.

- **Seer**: The Seer may once during the night look at another player’s role card, or two of the extra cards from the center.

- **Robber**: The Robber may once during the night take another player’s role card and exchange it for their own and look at their new role card.

- **Troublemaker**: The Troublemaker may once during the night exchange the role cards of two players of their choice, but they may not look at these cards.

- **Rascal**: The Rascal is a restricted variant of the Troublemaker in that they can only exchange the cards of their two neighbors, and may not look at these cards.

- **Drunk**: During the night, the Drunk exchanges their card with a card from the center without looking at it.

- **Insomniac**: At the end of the night phase, the Insomniac looks at their own card again, to check whether it was changed during the night.

- **Villager**: The Villager has no special abilities.

- **Tanner**: The Tanner has no special abilities.
Figure 2.2 A possible starting configuration for One Night Ultimate Werewolf. The roles are revealed for demonstration purposes only, in an actual game they would be secret.

After every player has performed the action corresponding to their role, the night phase is over and a time-limited day phase (typically 5 minutes) begins during which players can discuss which roles they believe other players to have. Players do not have to be truthful about anything they claim, though, so this phase of the game is about building trust and each player trying to convince the others that what they say is true, while determining which statements made by the other players to believe. After this discussion phase, all players simultaneously vote for a player that they want to “kill”. All players that have a plurality of votes “die” and reveal their cards. The winner is then determined as follows:

- The Tanner wins if and only if he dies.
- If at least one Werewolf dies, all non-Werewolf, non-Tanners win.
- Otherwise the Werewolves win.

This basically means there are three factions: The Werewolves, the Tanner and the Citizens\(^4\). Strategically speaking, the Werewolves want to appear to be Citizens, while the Citizens want to

\(^4\)The Citizens are often also just called Villagers, but I will call them Citizens to avoid confusion with the Villager role
deduce the truth. However, since players do not necessarily know which team they are on (such as when a Werewolf’s role card was switched with a Citizen’s role card without either of them knowing), they need to take into account what other players may have done. The Tanner, finally, wants to be suspected to be a Werewolf, but not appear to be doing so too intentionally, or the other players will start to suspect that they are in fact the Tanner.

Because a major part of the game is the day phase during which players can make arbitrary claims, a feature that it shares with its ancestor Werewolf/Mafia, much of the existing research focuses on analyzing audio and video recordings of game play and use them for deductions. Chittaranjan and Hung, for example, use vocal features, such as speaking duration, speaking rate, or pitch to identify lies [Chi10]. There has also been research on using the textual content of the players’ utterances, though. Gillespie et al. describe how these utterances can be classified into categories such as claims, questions or hypotheses [Gil16]. The classes generated by their system are then used by Hancock et al. to detect deceptive behavior by generating explanations for each heard utterance type in the form of a plan [Han17].

Because of the complexity of communication, research into optimal strategies for the game and its ancestors has been limited. Braverman et al. use a probabilistic model to determine the optimal strategies in several simplified configurations of Mafia, as well as the maximum number of Mafia players (which are the equivalent to the Werewolf players in Werewolf) to ensure a fair game [Bra08]. Because of the simplifications of the game, which in particular remove most of the communication by players, the optimal strategies involve random choice on part of the Citizens who don’t know anything, and a protocol to follow for the roles that can gain information. Xiaoheng and Tetsuro improve on this work by handling the case in which the Werewolf players always behave as if they were normal Villagers [Bi16]. They describe a communication protocol to use for the players that shares a candidate list for players to eliminate that is resilient against deceptive Werewolf players. However, it is also limited because it assumes the Werewolf to behave exactly as Villagers, and therefore the player with the Seer role is able to reveal themselves and will be believed by everyone. Nakamura et al., finally, describe an agent that uses what they call a psychological others model, which is a representation of other agent’s beliefs, to determine probabilities for each player having a certain role using data obtained from domain experts for various combinations of role probabilities and actions [Nak16].

2.2 Intentionality and Theory of Mind

Dennett describes three stances with which humans view a complex system: The physical stance, in which they explain its behavior based on physical laws (consciously or unconsciously), the design stance, in which they explain its behavior in terms of rules of operation, akin to a program, and the intentional stance, in which they explain its behavior based on its beliefs and
That last stance is particularly prevalent when the rules of operation or physical laws are unknown or hard to deduce. Dennett names a chess-playing robot as an example, where no player will be able to actually reconstruct the code it is operating under from observing the game play, let alone reason on the level of electrical currents. In contrast, ascribing desires to the robot, such as that it wants to win the game, take the queen, put its pieces into a particular kind of position, etc. and explaining its behavior in terms of these goals is natural to humans. Indeed, the intentional stance is so common in humans that they sometimes even ascribe intentionality to agents that behave randomly, as long as it is within the confines of their domain. Gallagher et al. report that subjects playing rock-paper-scissors against a randomly playing computer still try to explain its behavior to be intentional [Gal02]. However, they also report a difference in brain activity between a randomly playing player and one following a strategy, which indicates that the subjects made a difference between them at least subconsciously. Intentional behavior is therefore important for AI agents in video games, as Mateas has also argued [Mat03], even if the intentionality is often just a natural result of their behavior rather than an explicit design goal.

To use the concept of intentionality in an intelligent agent, Cohen and Levesque developed a logic-based formalization [Coh90] based on a definition of intentionality by Bratman [Bra90]. This definition describes intentionality as having several properties:

- **Intents normally pose a problem for the agent** (Bratman), which means that the agent has to actively solve the problem of how to achieve its intentions.

- Existing intentions limit what other intentions an agent can adopt. Bratman calls this a *screen of admissibility*, which means that agents can’t adopt intentions that would be in conflict with their already existing ones.

- An agent keeps track of what they did to achieve intentions, both successful and unsuccessful attempts. This is needed to be able to discard plans that did not work or no longer work.

- Agents believe they can actually achieve their intentions, i.e. they don’t intend something that they believe to be impossible.

- Agents don’t believe that they won’t work towards their own intentions. This is a logical necessity, because otherwise agents might get “stuck” while trying to achieve their intentions, because they end up with a plan, but don’t actually execute it.

- The agent believes that they will actually achieve their intentions eventually, if the conditions are favorable. Note that the beliefs of the agent do not necessarily have to be correct, and indeed they may still achieve their intentions even if the beliefs under which they came up with the plan to achieve them were wrong in a way that caused the plan to fail. However,
the agent believes that, as long as there is a way to achieve the intention in a reasonable way, they will do so.

- Agents can plan to perform actions even if they do not intend all effects of that action, even if they know about these effects. Cohen and Levesque call these effects *unintended side-effects*.

Cohen and Levesque’s develop a logical definition of intentions over the course of their paper by integrating more and more of the desired properties, resulting in a modality \( \text{intend}_2 x \phi \psi \), where \( x \) is the agent, \( \phi \) is what they intend to achieve and \( \psi \) is a relativization condition. The meaning of this modality is that the agent currently believes \( \phi \) to not hold, but they have a persistent goal of it holding and performing some action to achieve it until it holds, or \( \psi \) holds. The persistent goal represents an agent’s *choice* and the belief to perform actions towards that goal the agent’s *commitment*. While this logical formulation provides the means to describe intentions logically, it does not provide any means for the actual adoption of intentions. The relativization condition is also not elaborated upon, and is just assumed to be given, raising the question of how agents should decide to reconsider their intentions. Because these issues are key to having intentional agents, they have been addressed in a number of ways by various researchers. Braubach et al. observed that goals are often only talked about in abstract terms which leaves a gap between the theory and the actual implementation [Bra04]. They developed a classification of goals into different types such as achievement goals, where the agent wants some condition to hold that currently does not hold, or query goals where they agent wants to collect information. While they mention the importance of an agent’s deliberation about which goals to adopt as intentions, their main contribution is to provide a formalism upon which such a deliberation mechanism could be built, and they do not provide one themselves in this paper. Pokahr et al. use this representation to describe a goal deliberation mechanism they call *Easy Deliberation Strategy*, which selects goals purely based on the information contained in the goals, without regard for the plans needed to achieve them [Pok05]. This has the advantage of resulting in a simple, efficient mechanism that can be used even when the number of potential goals is very large, but it has the drawback that certain constraints cannot be expressed, such as conflicts between plans to achieve goals, or simply plan-complexity. Another method of adopting intentions is *influencing*, i.e. adopting an intention based on another agent’s actions, including communicative acts. Mohanty et al. describe an agent model that uses social norms to determine which intentions to adopt based on communicative acts by other agents, such as a supervisor giving orders to subordinates [Moh97].

Georgeff and Rao discussed the intricacies of when to maintain and when to drop intentions when using a formal model [Rao95]. Consider, for example, an agent having an intention of \( \phi \). In many formal models, this would imply that the agent also has the intention \( \phi \lor \psi \), because \( \phi \)
implies $\phi \lor \psi$. If at some point in time the agent no longer considers it possible to achieve $\phi$, but they believe it is possible to achieve $\psi$ they would then be forced to adopt $\psi$ as an intention, or there has to be some mechanism to drop the intention containing the disjunction. There are several solutions to this, with the one proposed in the paper being the introduction of a new only intends modality that is not closed under implication. Another solution, and the one adopted by my work, is to have a model of intentionality that is external to the logic itself. Wooldridge describes an agent architecture that uses beliefs, desires and intentions, but also maintains them separately, as opposed to requiring the same logical formulation to be used for all three [Woo00]. The work presented in his book serves as one of the key inspirations for my agent architecture that is discussed in more detail in section 4.2.

So far, I have focused on describing intentionality without regard for other modalities, but Cohen and Levesque argued that intentionality is closely linked to beliefs. Moreover, to communicate with other players, the AI agent needs to have a model of what that other player already believes, and when they are told something by a human player, having a model of what the human player believes that the AI agent believes can help with deducing the intentions of the human player. This model of beliefs and beliefs about beliefs, etc., a theory of mind, has been identified as a key part of social intelligence in humans [BC97; Wim83]. Castelfranchi has also argued that it is essential for social agents to be able to deduce other agents’ goals [Cas98]. It can also be used to model trust between agents in an environment that features potential deception [Lia03], and has been shown to provide an advantage in competitive games [DW13].

Because beliefs play such an important role in the context of intentional behavior, prior work on intentional agent has already yielded some insights into how to integrate the two modalities. The computational generation of narratives, in particular, has been a fertile ground for this work, because characters that work towards a goal have been described as being more believable [Rie10]. Teutenberg and Porteous, for example, use character beliefs of the world to inform a planner about which actions characters can take towards their goals [Teu15]. Their model, however, does not include nested beliefs, and uses a predefined model for which effects of an action characters observe and which stay hidden from them. Shirvani et al. expand upon this idea with a model that incorporates nested beliefs [Shi17a]. However, to keep the computational complexity low, their model defines character beliefs as single worlds, which prevents them from reasoning about alternative possibilities. Incidentally, this feature is shared by Teutenberg and Porteous’ model, and prevents the expression of even simple logical puzzles like hat-guessing games. Thorne and Young provide a different model, that is also does not include nested beliefs, but rather than limiting beliefs to being basically another copy of the world representation, they incorporate unknowns into their model of beliefs. This allows their agents to reason about failure without leading to logical contradictions. Because the games I am looking at require reasoning about beliefs about beliefs, as well as a sound logical deduction procedure, I am using Dynamic
Epistemic Logic. The challenge with this approach is that the memory and CPU requirements quickly grow to infeasible levels, which provided an engineering challenge I will elaborate on in chapter 4. In the next section, I will describe Dynamic Epistemic Logic in more detail, and provide an overview over the particular flavor I am using.

2.3 Dynamic Epistemic Logic

As described in the previous section, modeling beliefs and beliefs about beliefs of agents is essential for agents to be able to form intentions, deduce other agents’ goals and communicate with them. The formal framework on which I am basing my work is Dynamic Epistemic Logic. It, in turn, is based on Epistemic Logic, the logic of knowledge and belief\(^5\), which consists of ordinary first order predicate logic with a single modal operator \(\square\) (pronounced “box” or “believes”), that is used to represent agents’ beliefs. Typically, the agent whose beliefs are represented is written as a subscript. For example, \(\square_x \phi\) would mean that the agent \(x\) believes \(\phi\). By nesting \(\square\) operators with different subscripts it is possible to describe beliefs about beliefs, as in \(\square_y \square_x \phi\), meaning that agent \(y\) believes that agent \(x\) believes \(\phi\).

The semantics of the \(\square\) operator are typically defined using Kripke structures [KRI63]. They describe beliefs as a set of worlds, where one world corresponds to what is factually true, called the actual world, and the others represent other possible worlds. These worlds are connected with an accessibility relation for each agent, describing which worlds an agent considers possible in each world. An agent is said to believe some fact \(\phi\) if \(\phi\) holds in all worlds accessible to them from the actual world. For example, consider a simple card game in which there are only spades and clubs, and agent \(x\) holds two cards that they can see, but agent \(y\) can not. Then there are four possible worlds, one for each combination of spades and clubs in \(x\)’s hand. The actual world corresponds to the ground truth of cards in agent \(x\)’s hand, but agent \(y\) also considers each other world possible, so for agent \(y\), all four possible worlds are accessible from the actual world, while for agent \(x\) only the actual world is. However, agent \(y\) also knows that agent \(x\) can see their cards, so regardless of which world is the true world, agent \(x\) will consider that world (and only that world) possible, and therefore for agent \(x\) each world is accessible from itself. Figure 2.3 shows this possible worlds model as a graph, where each node is a possible world and the edges between worlds represent the accessibility relation.

However, to actually play a game, agents not only need to be able to reason about what the players know, but also how actions change this knowledge. Dynamic Epistemic Logic (DEL) describes this process formally. While there are many flavors of DEL, van Ditmarsch et al. call

\(^5\)Some authors distinguish between doxastic logic as the logic of beliefs, and epistemic logic as the logic of knowledge. For the purpose of this thesis, I use epistemic logic to include both, but acknowledge the existence of the distinction with the title for doxastic games.
Figure 2.3 A possible worlds model for a possible epistemic state in a simple card game: Agent $x$ holds two cards, and knows that they are a club and a spade, while agent $y$ only knows that the cards could be clubs or spades, but not which is which. Agent $y$ also knows that agent $x$ knows which cards they have.

the dominant approach one in which actions are represented in a similar way as states [VD07]. In this model, an action actually consists of a set of possible actions, with one action representing the factual change in the world. Similar to states, these actions are also connected with an accessibility relation for each agent, representing what that agent believes is happening. Also, similar to states, nesting these accessibility relations allows the representation of actions such as “agent $x$ believes that agent $y$ believes that agent $x$ looked at the top card of the deck”. Alexandru Baltag developed the particular flavor of DEL that I will use in my thesis [Bal02]. It follows this approach and integrates purely epistemic actions with factual change operators in a way that also allows agents to merely suspect an action, even if that action does not actually happen. For the remainder of this document, whenever a formula in DEL is presented, it is to be understood to be Baltag’s variant of DEL, which I will also simply call Baltag. I will now describe Baltag’s logic in more detail, starting with the syntactical elements followed by its semantics.

2.3.1 Syntax

Baltag defines how to write formulas in DEL as well as actions. However, because formulas can contain actions, and actions can contain formulas, these definitions are mutually recursive. For formulas, Baltag allows:

- Atomic propositions, $p(u)$, with arbitrary arity
- Conjunctions of formulas $\phi \wedge \psi$
- Negations of formulas $\neg \phi$
- Formula evaluations after the application of an action, $[a]\phi$. This is to be read as: $\phi$ holds after $a$ is executed
Beliefs of agents about formulas: $\Box_x \phi$, where $x$ is an agent

Mutual belief of agents about a formula: $\Box^*_X \phi$, where $X$ is a set of agents

The set of all atomic propositions is referred to as $AtProp$, while the set of all valid formulas is called $L$.

Actions can consist of the following components:

- The flip operator: $flip p(u)$, where $p(u)$ is an atomic proposition
- Testing if a formula holds: $?\phi$
- Sequential composition of actions: $a \cdot b$
- Non-deterministic choice between actions: $a + b$
- The suspicion operator: $(a)^x$, where $x$ is an agent
- The truthful, mutual learning operator: $(a)^*_X$, where $X$ is a set of agents

Note that the only operation that actually changes anything in a world is the flip operator, and it only changes a single bit of information. However, as I will describe later, the test operator $? \phi$ can be used to change an agent’s beliefs by removing subsets from the set of worlds that they consider possible.

### 2.3.2 Formula Interpretation

An epistemic state $\mathcal{M}$ is used to interpret formulas, i.e. to determine which formulas hold. Formally, it is a 4-tuple $(S, \{x\} \cdot_x, \cdot_0, s')$, where $S$ is a set of (possible) worlds, each $\cdot_x$ is a function $\cdot_x : S \mapsto 2^S$, called the appearance map, $\cdot_0$ is a function $\cdot_0 : S \mapsto 2^{AtProp}$, called the content map and $s'$ is an element of $S$ called the actual world. Informally, the appearance map defines which worlds an agent considers possible in each world in $S$, i.e. what each world “looks like” to each agent, and the content map defines what holds in each world. For any world $s$, $s_0$ contains exactly the atoms that are true in that world. Note that this approach differs slightly from the more common definition of possible worlds, in which the accessibility relation is defined as an actual relation. The two definitions are equivalent, though, and an accessibility relation $\rightarrow_x$ between worlds can be defined as: $\rightarrow_x = \{(s_1, s_2)|s_1 \in S, s_2 \in (s_2)_x\}$, i.e. a world $s'$ is $\rightarrow_x$-accessible from a world $s$ iff it is in $s_x$, leading to the alternative, but equivalent, definition of $s \rightarrow_x s' \iff s' \in s_x$.

Defining the accessibility relation as a function simplifies several subsequent definitions, and also provides a more direct mapping to an implementation.

Baltag often uses systematic ambiguity to identify an epistemic state with its actual world, by lifting the content- and appearance maps from the level of worlds to that of a state. This
means, given a state $\mathcal{M} = (S, \{x\}_x, \cdot_0, s')$, writing $\mathcal{M}_x$ would be equivalent to writing $s'_x$, i.e. all worlds that are in agent $x$'s appearance map for the actual world. Likewise $\mathcal{M}_0$ refers to the content of the actual world. This becomes useful when one considers that each possible world could serve as the actual world of an epistemic state, so just as the appearance map can be applied to an epistemic state rather than a world, its result can be a set of states rather than worlds. This way, the set $S$ does not have to be explicitly defined anywhere, and it is possible to define an epistemic state by just giving the content of its actual world, and an appearance map that maps to other epistemic states defined the same way. I will explain in section 4.1.5.1 how this is actually beneficial when implementing an interpreter for Baltag formulas, but for this section the main benefit lies with the ability to talk about epistemic states more abstractly.

Given an epistemic state, $\mathcal{M} = (S, \{x\}_x, \cdot_0, s')$, formulas can be interpreted as follows:

- $\mathcal{M} \models P \iff P \in \mathcal{M}_0$, for atoms $P$, i.e. the atoms that hold in a state are exactly the atoms in the content of the actual world.
- $\mathcal{M} \models \phi \iff \mathcal{M} \not\models \phi$
- $\mathcal{M} \models \phi \land \psi \iff \mathcal{M} \models \phi$ and $\mathcal{M} \models \psi$
- $\mathcal{M} \models \Box_x \phi \iff \forall \mathcal{M}' \in \mathcal{M}_x : \mathcal{M}' \models \phi$, i.e. an agent believes $\phi$ iff $\phi$ holds in all worlds (here represented as epistemic states, as described above) they consider possible in the actual world.
- $\mathcal{M} \models \Box^* \phi \iff \forall \mathcal{M}' \in \mathcal{M}_X : \mathcal{M}' \models \phi$, where $\mathcal{M}_X^*$ is the iterated, mutual version of the appearance map, which consists of all worlds that are accessible via any combination of appearance maps of agents in $X$, i.e. the worlds in the reflexive, transitive closure of $\bigcup_{x \in X} \rightarrow_x$. This means a group of agents mutually believes $\phi$ iff $\phi$ holds in all worlds that any combination of agents can access in any number of steps.
- $\mathcal{M} \models [a] \phi \iff \mathcal{M}' \models \phi$ for every $\mathcal{M}'$ that satisfies $\mathcal{M} \Rightarrow^a \mathcal{M}'$, where the $\Rightarrow^a$-relation holds between two states $\mathcal{M}$ and $\mathcal{M}'$ iff $\mathcal{M}'$ is one possible result of applying $a$ to $\mathcal{M}$. These possibilities can arise because actions can be non-deterministic. Basically, $\phi$ has to hold regardless of which of the potential choices of $a$ is taken.

Consider, for example, the epistemic state $\mathcal{M}$ from figure 2.3. Let the atoms left($c$) and right($c$) stand for the left card and right card, respectively, where $c$ can be either clubs or spades. Then $\mathcal{M} = (\{s_{cc}, s_{cs}, s_{sc}, s_{ss}\}, \{x\}_x, \cdot_0, s_{cs})$, where $s_{cc}$ is the world with two clubs, $s_{sc}$ is the world where the left card is spades and the right card is clubs, etc. The appearance map $\cdot_x$ is

\footnote{Definitions by Baltag, reformatted to fit with the conventions used in this document, and with explanations added}
defined as \((s)_x = \{s\}\), because every world (only) appears as itself to \(x\), and \(\cdot_y\) is defined as \((s)_y = \{s_{cc}, s_{cs}, s_{sc}, s_{ss}\}\), because \(y\) considers any world possible in every world. The content map \(\cdot_0\), finally, is defined as \((s)_0 = \{\text{left(clubs)}, \text{right(clubs)}\}\). By following the definitions above, one can determine the truth of any statement. For example, \(\Box_x \Box_y (\text{left(clubs)} \rightarrow \text{right(clubs)})\) can be interpreted by:

1. \(\mathcal{M} \models \Box_x \phi\) where \(\phi = \Box_y (\text{left(clubs)} \rightarrow \text{right(clubs)})\), which means for every world \(\mathcal{M}'\) accessible by \(x\) from the actual world \(\mathcal{M}' \models \phi\) has to hold.

2. Because \(s_x = \{s\}\) for any \(s\), and therefore \(\mathcal{M}_x = \{\mathcal{M}\}\), the next step is to interpret \(\mathcal{M} \models \Box_y (\text{left(clubs)} \rightarrow \text{right(clubs)})\)

3. Like in step 1, the inner formula, \((\text{left(clubs)} \rightarrow \text{right(clubs)})\) has to be interpreted in every world in \(\mathcal{M}_y\). Because \(y\) does not know which world is the actual world, \(\mathcal{M}_y = \{\mathcal{M}_{cc}, \mathcal{M}_{sc}, \mathcal{M}_{cs}, \mathcal{M}_{ss}\}\).

4. One of these worlds is \(s_{cc}\), so it needs to be determined whether \(\mathcal{M}_{cc} \models (\text{left(clubs)} \rightarrow \text{right(clubs)})\).

5. \((\text{left(clubs)} \rightarrow \text{right(clubs)}) \equiv \neg (\text{left(clubs)} \land \neg \text{right(clubs)})\), which can be interpreted by determining whether \(\mathcal{M}_{cc} \models \text{left(clubs)} \land \neg \text{right(clubs)}\), which does not hold, and therefore its negation does (steps leading to the interpretation of the atoms omitted).

6. However, another world is \(s_{cs}\), for which \(\mathcal{M}_{cs} \models \text{left(clubs)} \land \neg \text{right(clubs)}\) holds, and therefore its negation, which is the implication from the original statement, does not.

7. Because \((\text{left(clubs)} \rightarrow \text{right(clubs)})\) does not hold in all worlds that are in \(\mathcal{M}_y\), \(\mathcal{M}_x \not\models \Box_y (\text{left(clubs)} \rightarrow \text{right(clubs)})\), and therefore \(\mathcal{M} \not\models \Box_x \Box_y (\text{left(clubs)} \rightarrow \text{right(clubs)})\)

Note how worlds and states are used interchangeably by identifying a state with its actual world. This why interpreting a formula in the world \(s_{cc}\), for example, is done by interpreting it in \(\mathcal{M}_{cc}\), which is the epistemic state obtained by setting \(s_{cc}\) as its actual world.

### 2.3.3 Actions and Action Application

Like states, actions are represented using a set of possibilities with an appearance map. Formally, an action is a set of quintuples \(A = \{(K, \{\cdot_x\}_x, \cdot_0, \text{pre}, k')\}\), where, similar to before \(K\) is a set of possible action tokens, \(\cdot_x\) defines an appearance map \(\cdot_x : K \rightarrow 2^K\), \(\cdot_0\) is the content map or change map, \(\cdot_0 : K \rightarrow 2^{\text{AtProp}}\), and \(k'\) is an element of \(K\) called the actual action (token). What is different compared to states is the addition of a precondition function, \(\text{pre} : K \rightarrow L\), where
$L$ is the set of all valid formulas. Additionally, an action is actually a set of such structures, because actions can be non-deterministic. Each element of the set corresponds to one possible deterministic choice, called a simple action. As with states, the functions $\cdot_x$, $\cdot_0$ and pre are often applied to simple actions rather than action tokens, which is to be interpreted as applying them to the actual action token of that simple action. Likewise, an action token obtained from an appearance map is often treated as being a simple action where that action token is the actual action token.

Before I describe how actions are applied to states, I want to briefly summarize the relationship between the syntax presented above and this model. To define how a syntactic element is mapped to an action, three items need to be defined: the content change it performs, its appearance to the agents and its preconditions. Like Baltag, I will use the symmetric difference operator $\Phi \triangle \Psi \equiv \Phi \setminus \Psi \cup \Psi \setminus \Phi$. Additionally, skip is an action with no effect, that looks like itself to all agents. Each of the syntactic elements is mapped to action tokens and simple actions as follows:

- flip $p(u)$ becomes a simple action with no precondition, an appearance of $\{\text{skip}\}$ to all agents, and $p(u)$ as its only content change.
- $?\phi$ becomes a simple action with $\phi$ as its precondition, an appearance of $\{\text{skip}\}$ to all agents, and no content change.
- $a \cdot b$ becomes a simple action with $\text{pre}(a) \land [a]\text{pre}(b)$ as its precondition, an appearance of $\{c \cdot d | c \in a_x, d \in b_x\}$ for all $x$, and a content change of $a_0 \triangle b_0$. Note that for each $c \cdot d$ in the appearance map the precondition, appearances and content change is defined recursively in the same way.
- $a + b$ becomes a set of two simple actions, with the two elements being the simple action corresponding to $a$ and the simple action corresponding to $b$.
- $(a)^x$ becomes a simple action with no precondition, an appearance that consists of the set of simple actions corresponding to $a$ for $x$ and an appearance of skip for all other agents, and no content change.
- $(a)^X$ becomes a set of actions, with one element for each simple action $a_i$ corresponding to $a$. For each of these simple actions $a_i$, the appearance map is then changed, such that $(a_i)_x = \{\text{skip}\}$ for all $x \not\in X$, and $a_i$ is added to $(a_i)_x$ for all $x \in X$, i.e. the action also appears as itself to the agents that mutually learn about it.

The first observation that can be made here is that syntactic elements may turn into sets of actions, because of the non-deterministic choice operator. This may appear undesirable, but is actually essential for epistemic effects. Consider an action $\text{flip } p \cdot (\text{flip } p + \text{flip } q)^x$, which can be
interpreted as "p is flipped, but to x it appears as if either p or q is flipped, but they don’t know which one." This action can be translated to a simple action by following the rules outlined by starting with the first sequence operator. This requires translating both of its operands first. The first, flip p results in a simple action that has a content change of \{p\}, no precondition and an appearance of skip to all agents. The second operand, \(\text{flip } p + \text{flip } q\)^x, results in an action that has no precondition or content change, but an appearance corresponding to the simple actions that correspond to flip p and flip q. However, because this action consists of a non-deterministic choice, it will result in two simple actions, one for flip p and one for flip q. Both of them have no precondition, an appearance of skip to all agents, and a content change of \{p\} or \{q\}, respectively.

Figure 2.4 shows a graphical representation of the resulting epistemic action. Note how the non-deterministic appearance resulted in two possible actions that appear to agent x as epistemic alternatives to the actual action, while agent y is not aware of anything happening at all.

Once an action is translated into simple actions of the form just described, each simple action \(a\) can be applied to an epistemic state \(s\) potentially producing a new epistemic state. This is done with what Baltag calls the restricted product \(s.a\), which is a partial function that is only defined if \(s \models \text{pre}(a)\):

\[
(s.a)_0 = s_0 \triangle a_0 \\
(s.a)_x = \{s'.a' | s' \in s_x, a' \in a_x\}
\]

Informally, this means that the content change is applied to the content of the state, and then each appearance of the action is applied to each appearance of the actual world, if the corresponding preconditions hold. It is this last part that allows agents to learn information. Consider the game state from figure 2.3 once more. If player y should learn that the left card is a clubs, the action...
(\(? \text{left}(\text{clubs})\)^*\text{y}) would result in that. As described above, this action would have a precondition of \(? \text{left}(\text{clubs})\) and an appearance of itself (for \text{y}). This action’s precondition, \text{left}(\text{clubs}) holds in the actual world, so applying it to the epistemic state in figure 2.3 will yield a new epistemic state. When it is applied to the epistemic state, its content change (which is empty) is applied to the actual world and then its appearance to each agent is applied to the appearance of the actual world to that same agent. Its appearance to agent \text{y} is the action itself, so it will be applied to each world that \text{y} considers possible, \text{if its precondition holds in that world.} This condition is crucial, because it means that the action does not apply to the worlds in which \text{left}(\text{spades}) holds, so these two worlds do not result in any worlds in the new epistemic state. The result is therefore that agent \text{y} no longer considers any worlds possible in which the left card is not a clubs, resulting in the epistemic state shown in figure 2.5.

\subsection{2.3.4 Practical Considerations}

As mentioned earlier, the only operator that directly changes the state of the world is the flip operator that can change a single bit of information at a time. Any non-trivial application of this logic therefore needs to encode all information into bits, encode the actions to change them accordingly and then convert back to the originally intended interpretation. For example, for a standard 52 card deck, an agent \text{x} looking at the top card of the deck would have to be encoded as a non-deterministic choice between all possibilities, and each choice wrapped in the truthful, mutual learning operator. Assuming that the atom \text{t}(\text{x}) has the meaning “The top card of the deck is \text{x}” the resulting formula would look like this:

\[
(\text{t}(\text{AceOfHearts}))^{*\text{x}} + (\text{t}(\text{TwoOfHearts}))^{*\text{x}} + \cdots + (\text{t}(\text{KingOfSpades}))^{*\text{x}}
\]
Because such long formulas come up in even simple scenarios like this, Baltag defines sum- and product-shortcuts for them, which would shorten the formula to $\sum_c (?t(c))^{xz}$. Of course, this abbreviation requires an implementation to have a concept of sets, or even a whole expression language to describe which set expressions the sum or product should range over. Other abbreviations Baltag suggests are on a purely syntactic level, though, such as this simple if statement:

$$\text{if } \phi \text{ then } a \text{ else } b \equiv (?\phi \cdot a) + (?-\phi \cdot b)$$

While these abbreviations make formulas slightly easier to write, many cases that appear in practice are still very cumbersome to write. Consider that the agent $x$ should exchange the top card of the deck with the single card they are holding in their hand (represented by the predicate $h$), while looking at both of them, and another agent, $y$, only knows that the exchange is happening, but not which cards are involved. To break down what this operation involves:

- Agent $x$ learns the identity of the top card, as above
- The top card of the deck becomes the card in agent $x$'s hand, and the card in agent $x$'s hand becomes the top card of the deck
- Agent $y$ sees that this is happening, but not which of the many possible exchanges (one for each possible pair of cards) takes place

I already described how agent $x$ learns the identity of the top card of the deck. For each of these possibilities, however, the agent now needs to consider each possible card they have in their hand, and make the exchange. All of these possible exchanges will be part of a non-deterministic choice, where agent $y$ only knows that the choice happens, but not which of the options was taken. As a formula, this can be written as:

$$\left( \sum_c \sum_d (?t(c) \wedge h(d)) \cdot \text{flip } t(c) \cdot \text{flip } h(c) \cdot \text{flip } h(d) \cdot \text{flip } t(d)^{xz} \right)^{xy}$$

Another challenge encountered when writing game actions in DEL is the difficulty of encoding the game domain. The preceding formula, for example, contained the predicate $t$ to express that a particular card is the top card of the deck. In any given state, only one card can be at the top of the deck, though, and it is up to the author of the actions to ensure that this property of the domain is kept as an invariant. In chapter 4 I will therefore present my approach to this problem, which consists of a macro language that can be compiled to Baltag's logic, and makes writing the actions that are necessary to encode complex domains much simpler.
As a first example for how intentional agents can improve the game experience for human players, I worked on the card game Hanabi, described in section 2.1.1. My work builds directly on Osawa’s work, specifically the Outer State Player [Osa15]. His Self Recognition Player, while increasing performance marginally, also requires enumerating all possible hands and performing some calculations on each possibility, which makes it unsuitable for use in a real-time scenario, such as when playing with human players. However, I also used the general idea of using the agent’s own hint-giving strategy for hint interpretation in a more computationally efficient way. Before I describe my own agents in more detail, I want to present an summary of Osawa’s. On each turn, the agent will perform an action according to a fixed, hand-crafted priority list:

1. If a card is certain to be playable, play that card
2. If a card is certain to be useless, discard that card
3. If a suitable hint can be given, give that hint
4. Discard a card

The two differences between my agent and Osawa’s are how they decide which card to discard, and the definition of suitable hints. Osawa’s agents always discard random cards if they are not certain that a card is useless. However, in many cases the agent has information about cards that can be used to calculate which cards are more likely to be useless, even if they are not certain.
The more important difference, though, is how to decide which hints to give. Osawa’s agents give hints in the following order of priorities:

- If there is a playable card, give a random hint about it
- If there is a useless card, give a random hint about it
- Give a random hint

The Outer State Player will also keep track of the knowledge state of the other player, and not give the same hint twice. However, when playing with a human player, this hint-giving strategy still conveys unwanted information. First, consider the fall-back case of giving random hints. A human player expects a hint to serve a purpose. However, they have no way of determining when a hint given by the AI agent is just the fall-back case of a random hint, or when it is about a playable card. So, for example, if the AI agent falls back to giving a random hint, and that hint consists of telling the human player that a single card in their hand is red, the human player might interpret that hint as meaning that the card is the next playable red card and play it on their turn. This is because humans assume communication to follow Grice’s maxims of communication [Gri75], and in this particular scenario the maxim of relation, which states that communication should be relevant, and not superfluous, is violated. The second problem with this approach is that even when the AI agent gives hints that are relevant (because they pertain to playable or useless cards), they still give a random hint about such cards. For example, if a player has the blue 3 and the blue 4 in their hand, and the blue stack currently has a 3 on top, which makes the 3 they hold useless and the 4 playable, and the AI agent tells them where their blue cards are, they have no way of knowing which card is which. They might think that both cards are 4s, or they use some meta-information, such as which card they drew more recently, to decide to play one of them. The problem in this case is a violation of the maxim of manner, which states that communication should be unambiguous.

A similar argument can be made for why discarding a random card as the default fall back action is not optimal. While it does not involve communicating any information to the human player, it does exhibit the same lack of a focus on a goal as the random hint giving strategy. And similar to the case with giving hints that might be misinterpreted, the agent does not check whether the action they are about to perform has any potentially harmful side effects. My agents address both of these issues using the idea of intentionality in a limited way. I will now describe how my agents differ from Osawa’s, and how the potential improvement achieved by this difference was tested.
3.1 An Intentional AI for Hanabi

As an improvement over Osawa’s work, I developed two AI agents that use the same general outline as his Outer State AI, but performs actions in an intentional way. For the case of giving hints this means that the agent will adopt a goal that it wants to achieve and takes the appropriate hint-action to achieve that goal. For discarding there is only one goal the agents follow, which is “minimize potentially lost points”, and they use the information available to them to calculate which card to discard to achieve this goal. Finally, because the agents may also receive hints, one of my agent also assumes intentionality on behalf of its cooperator and will interpret hints given to it using this assumption. Before I describe how the agents decide which hints to give, I will briefly explain how they store the knowledge players have about their cards.

3.1.1 Knowledge Representation

Because the 50 cards in Hanabi would result in too many possible worlds for an explicit Dynamic Epistemic Logic representation, I used a different way to represent the agents’ knowledge base. For every card in each player’s hand, the agent stores the possible cards in a two-dimensional array, with one column for each color and one row for each rank. Each cell in this array contains a natural number representing how many cards of the corresponding rank and color are still unaccounted for. Initially, each cell contains the number of copies of each card in the game, as shown in table 3.1. When a hint is given about a card, the table changes accordingly by setting all cells that belong to cards that do not correspond to the given hint to 0. For example, if a player is hinted about all red cards, the cards that are actually red will have all values in all other columns set to 0, while the cards that are not red will have all values in the column labeled “red” set to 0. Additionally, cards that are played or discarded decrease the value in the corresponding cell by 1. This means, after all 3 red 1s have been played or discarded, the cell corresponding to the red 1 will be 0 for all cards, regardless of which hints were given. Table 3.2 shows what a player knows after they are given a hint that a card is red, and one each of the red 1 and the red 2 have been played, but no other red cards have been played or discarded. Each agent keeps track of this knowledge for all players, and assumes this information to be common knowledge, i.e. every player tracks all hints that are given as well as cards that are played and discarded. Additionally, an agent can use its knowledge of the cards in the other players’ hands to update their own knowledge base, by subtracting 1 from every cell corresponding to each card they see. Using this approach, each agent can keep track of what they know about their cards as well as what the other players know about theirs using a negligible amount of memory of $5 \times 5 \times 5 = 125$ integers per player.
Table 3.1 Agents’ knowledge state for a card about which no information is known

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>yellow</th>
<th>green</th>
<th>blue</th>
<th>white</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2 Agents’ knowledge state for a card after they have been told that the card is red, and a red 1 and 2 have been played

<table>
<thead>
<tr>
<th></th>
<th>red</th>
<th>yellow</th>
<th>green</th>
<th>blue</th>
<th>white</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

3.1.2 Giving Hints

To decide which hint to give, the agents will first determine a goal for each card in their cooperator’s hand, then predict what the cooperator’s actions will be for each possible hint, and finally give the hint for which the predicted action on part of the cooperator best matches the agent goals. This is intentional behavior in the sense that the agent chooses a goal, forms a plan to achieve that goal and commits to this plan by executing its first step. However, it is a limited kind of intentionality, because the agent won’t commit to a goal over multiple turns. Figure 3.1 shows the algorithm used to determine which hint to give. The function `CalculateGoals` returns a potential goal for each card in the cooperator’s hand ($B$). These are actions that the AI would like the other player to perform, such as playing the card or discarding it. Then the agent iterates over all hints, first over all hints that tell the other player about all cards of a particular color, then over all hints that tell the other player about all cards of a particular rank, and predicts which action the player will do if they receive that particular hint given their current knowledge about their hand $M_B$. This action is then compared to the goals and scored, scoring lower if it contradicts the goals, and higher if it is a more desirable action such as playing a card. Finally, the highest scoring action will be returned if its score is greater than 0. This means that if all hints result in the other player performing an action that conflicts with the agent’s goals, no
Figure 3.1 Algorithm to determine which hint to give to the cooperator, if any

hint will be given.

The potential goals are assigned by `CalculateGoals` using a preference list:

- Cards that can be played are assigned the goal of being played with a score of 3.

- Cards that will never be playable, such as duplicates of cards on the board, are assigned the goal of being discarded with a score of 2.

- Cards for which duplicates are left in the game are assigned a goal of being discarded with a score of 1.

- Cards for which no duplicates are left in the game, but which are not playable yet are assigned a goal of being kept with a score of 0.

To predict what the other player will do with a hint that they receive, the agent uses a similar priority list, and the fact that humans will expect communication to follow Grice’s maxims of communication [Gri75]. In particular, the maxim of relation states that communication should be relevant, which in this case means that hints that are given have some impact on the game state,
i.e. the player is expected to do something with the cards they are hinted about. Additionally, the maxim of manner states that communication should be unambiguous, which means that the player will try to interpret the hint as having a direct and clear meaning. To perform the prediction, the agent calculates the new knowledge of the human player about the hinted-at cards and assigns an action in the following way:

- If the given hint is consistent with the card being playable, the player will play the card.
- If the given hint is consistent with the card never being playable again, the player will discard the card.
- If the given hint is consistent with the card having duplicates left in the game, the player will discard the card.
- Otherwise the player will keep the card.

A hint is consistent with a property if and only if, after receiving the hint, the knowledge the player has about the card still contains possibilities that have that property. For example, if a player knows nothing about a card and is told that the card is red, and the red stack currently shows the 2, the information the player has is consistent with the card being playable, but also with the card never being playable again, and - if no red 4 has been discarded yet - with the card having duplicates left in the game. The priority list means that the player will prefer playing cards over discarding them, though. In practice, this matches with how humans play the game, where such hints are naturally interpreted as indicating to play the card.

Finally, the prediction is compared to the potential goals, and scored. If the predicted action matches the goal action, the score of the goal action is added to the total score, otherwise, if the score of the predicted action is higher than the score of the goal action, the total score is set to $-1$. This ensures that the player does not think a hint is better than it actually is. For example, if the player would play a card that the agent would want it to discard, the score of the hint action is set to $-1$ and the action will not be performed.

### 3.1.3 Discarding Cards

As opposed to Osawa’s agent, when deciding which card to discard, my agent takes the information it has about its cards into account. From an intentional perspective, the agent has to goal of minimizing the expected value of lost points with the discard action. The information contained in the two-dimensional array holding the knowledge about each card in the players hand can be used to approximate the probability for a card being each of the possible color/rank combinations, by dividing each value by the sum of all values in the array. For example, if the entries for the red 1, red 2 and red 3 are 3, 2 and 1, respectively, and all other entries in the array are 0, the
probability that the card is the red 1 from the agent’s point of view can be approximated by $\frac{3}{3+2+1} = 0.5$, and likewise for the probability that it is the red 2 or red 3. For each color/rank combination a card can possibly be, the agent then calculates how many points they expect to lose by discarding it as follows:

- If the card will never be playable, the agent will lose 0 points by discarding the card.
- If the card will be playable at some point and no duplicates exist, the player will lose $5 - r + 1$ points, where $r$ is the card’s rank. For example, by discarding the only remaining 4 of a color, the players will lose 2 points in total, because neither the 4 nor the 5 can be played in that color anymore.
- If the card will be playable, but duplicates exist, the number of lost points is estimated to be $c \times (5 - r + 1)/d^2$, where $r$ is the card’s rank, $d$ is the difference between the top card of the deck and the card’s rank and $c$ is a constant less than 1. This ensures that cards that will be playable sooner are counted as more important than cards that will be playable later.
- Additionally, because 5s regenerate a hint token when successfully played, their value is increased by 0.5 to account for that fact.

The overall expected loss is the weighted sum of these expected lost point values, where the weights are the probability estimates. The agent will discard the card with the lowest value for this sum.

### 3.1.4 Interpreting Hints

As mentioned above, I created two versions of the agent. One, called the *Intentional AI*, only uses the hint-giving and discarding logic described above, while the other one, the so-called *Full AI*, also interprets hints it receives by assuming intentional behavior on part of the other player. This is done in the same way the agent predicts its cooperator will act, which means if the agent plays with itself, its predictions will match exactly with its actions. Concretely, this means:

- If a hint is given that is consistent with a card that is hinted about being playable, the agent will play that card.
- If a hint is given that is consistent with a card never being playable, the agent will discard that card.
- If a hint is given that is consistent with a card having duplicates left in the game, the agent will discard that card.
Table 3.3 Average scores (with sample standard deviation) after 10000 games for each AI pair

<table>
<thead>
<tr>
<th></th>
<th>Outer</th>
<th>Intentional</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer</td>
<td>12.8 (2.0)</td>
<td>13 (2.1)</td>
<td>6.9 (4.3)</td>
</tr>
<tr>
<td>Intentional</td>
<td>12.6 (2.6)</td>
<td>14.6 (2.7)</td>
<td></td>
</tr>
<tr>
<td>Full</td>
<td></td>
<td>17.1 (2.5)</td>
<td></td>
</tr>
</tbody>
</table>

For example, if the agent is given a hint that says that one of their cards is red, and there are playable red cards left in the game, the agent will assume that the hint was supposed to refer to a playable card, and play it.

3.2 Results

To test how my agents perform compared to Osawa’s, I first tested how all three agents perform when playing with other agents. As mentioned earlier, I used the agent Osawa called Outer State AI as a baseline, which follows the priority list described above and remembers what the other player knows to avoid giving duplicate hints. I compared this to my Intentional Agent and the Full AI. Table 3.3 shows the average scores after 10000 games for every combination of agents. What is noteworthy is that the Intentional AI performs basically indistinguishable from Osawa’s Outer State AI, while the Full AI performs significantly better when playing with another copy of itself. This is not surprising, because the agent interprets hints in exactly the way it gives them, which results in very efficient communication. However, when the Full AI plays with the Outer State agent, their average score is very low. This is because the Full AI is very aggressive with interpreting hints as serving a purpose, and frequently misinterprets the random hints it gets from the Outer State agent.

However, the main goal of this work was to show that my approach works better when playing with human players. To do that, I implemented a web front end for the game, shown in figure 3.2, which allows players to play the game in a standard web browser. I then ran an experiment in which participants played with one of the three agent types assigned at random, with one of 5 possible starting decks, and were then asked questions about their experience in a post-game survey. Players could also choose to play more games after the first one with a completely random deck, but would not be presented with any additional surveys. I recruited participants on several board gaming web forums, such as boardgamegeek.com and the board game sub-reddit. Overall, 224 participants completed at least one game, with 99 players playing more than one game. Among these, the median number of games was 3.5, with one particularly engaged participant playing 440 games. The total number of games played among all participants
was 1211. Participants were aged 18 – 64, but skewed heavily towards gamers, with 152 self-identifying as such, only 14 answering whether they identified as a gamer with “no” and 52 declining to answer. However, because teaching players this game or about gaming in general was not part of my objectives, I do not see this skewed sample as a problem.

The results of the experiment itself were encouraging. When only taking the first game of each player into account, the average score when playing with the Outer State AI was 11.09 (standard deviation: 4.59, n = 79), while the average score when playing with the Intentional AI was 14.99 (standard deviation 4.17, n = 73). The average score for participants that played with the Full AI was 12.88 (standard deviation: 5.98, n = 72). Figure 3.3 shows the score distribution for the three AI agents.

An ANOVA showed that the AI the participant played with was indeed the relevant factor in deciding the score, even when controlling for age, board game experience, experience with Hanabi, recency of play or which of the 5 decks they played with (p < 0.0001). A Tukey test
Figure 3.3 Score distribution after the first game played by each participant, grouped by which agent they played with.
accounting for multiple testing showed that the difference between the Outer State AI and the Intentional AI is statistically significant \( p < 0.0001 \), as is the difference between the Intentional AI and the Full AI \( p = 0.0295 \). However, the difference between the Full AI and the Outer State AI was only weakly statistically significant \( p = 0.0715 \). A particularly interesting result is that the average score participants achieved with the Intentional AI is actually higher than the score that AI achieves when playing with itself, while the average score when playing with the Outer State AI is slightly lower than in the pure AI scenario. I believe this best highlights how the Intentional AI gets closer to how humans expect the AI agent to behave, and also benefits from the human intelligence, while for the Outer State AI the human player is actually a hindrance.

When looking at all games played by the participants, this effect becomes even stronger. The average score for games played with the Outer State AI decreased significantly to 8.12 (standard deviation: 4.87, \( n = 417 \)) compared to only the first game for each participant, while the scores for the Intentional AI decreased only slightly to a mean of 13.83 (standard deviation: 5.18, \( n = 393 \)). The average score for the Full AI, however, increased slightly to 13.16 (standard deviation: 5.76, \( n = 401 \)). As a result, the difference between the Outer State AI and the Full AI is statistically significant \( p < 0.0001 \) when looking at all games. As above, figure 3.4 shows the score distribution over all games for the three agents. I believe that the decrease of the score for the Outer State AI is due to a learning effect, where players expected more intelligent behavior from the agents after they played with the Intentional or Full AI, and misinterpreted hints given by the Outer State AI. Furthermore, as players got more used to getting smarter hints, and realizing that they, in turn, could also rely on the AI to interpret hints intelligently, their scores when playing with the Full AI started to improve. While not statistically significant due to a lack of sufficiently many samples, it is nevertheless interesting to look at the top scoring games. The only game that resulted in 24 points was played with the Full AI, while of the 5 games that resulted in a score of 23, 4 were played with the Full AI, and only one with the Intentional AI. Of the 62 games with scores of more than 20 points, only 3 were played with the Outer State AI, 10 with the Intentional AI and 49 with the Full AI. This provides additional evidence that not only does the Full AI perform better when playing with human players, it is also better at enabling high scores.
Figure 3.4 Score distribution after all games played by the participants, grouped by which agent they played with.
To generalize the agent used for Hanabi to an intentional AI framework that can be used for a broader class of games and other domains, it is necessary to have a knowledge and action representation that can be used to reason about other agents’ beliefs, and their beliefs about beliefs, and how they change over time. Dynamic Epistemic Logic, which I described in section 2.3, provides these capabilities and is therefore suitable for this purpose. As already mentioned, Baltag’s variant of DEL has the capability to have agents suspect actions without the action actually happening, which is particularly helpful in domains in which some actions happen in secret, such as games. However, because of the limitations discussed in section 2.3.4, using Baltag directly is prohibitively cumbersome in non-trivial domains. For example, the action of giving a hint about all cards of a particular color in Hanabi requires a decision for each card, whether it is of the chosen color or not. Because there are 5 cards of a color chosen for the hint and 20 that are not, each such check requires at least 25 terms, for a total of 125 terms for all 5 cards. Worse, an action describing the game action of playing a card would first have to check whether the card that is being played is the next card in sequence for its corresponding stack. There are multiple ways to express this in DEL, with one consisting of the steps:

- Determine which color the played card has using a non-deterministic choice between the 5 color options.
- For each color, determine if the played card is the next in sequence or not, also using a
hintcolor(p: Players, c: Colors)
learn (p): Each i in HandIndex:
color(at(p, i)) == c

Listing 4.1 The “hint about a color” action for Hanabi

non-deterministic choice.

• If the card is the next in sequence, increase the top card of the stack by one.

• If the card is not the next in sequence, increase the mistakes made by one, by using a non-deterministic choice over the possible values of the number of mistakes.

The final action description therefore contains several nested non-deterministic choices, basically coming down to all possible combinations of played cards and top cards of the stack followed by what actually happens in each case. While it is possible to write this using Baltag’s abbreviations for sums and products, an implementation of this model would then have to provide the appropriate means to define and select sets to sum over, without a major gain in ease of authoring.

In this chapter I will describe Ostari, which is the system I developed to address these challenges. Ostari is a programming language for epistemic actions that maps to Baltag, but can also be executed directly. Listing 4.1 shows how the action of hinting about all cards of a particular color in Hanabi can be written using this new language. In the next section, I will describe the details of the language, its compilation and execution below, followed by an explanation of an agent design that is also integrated in the system.

4.1 The Ostari System

The system Ostari constitutes the basis on which all following parts of my work are built. It consists of a language, demonstrated briefly above and described in more detail in the next section, and a runtime system for this language that supports the definition of an initial state and various modes of execution and state inspection. Additionally, an agent framework, described below, is integrated in the system, which is used for game playing. Because of the focus on games, Ostari also has several convenience features to make defining games less cumbersome. While my main focus was the game One Night Ultimate Werewolf, I will also briefly describe other domains to which Ostari has been applied successfully. This includes some work on narrative generation by myself, as well as a detective game, implemented by an undergraduate student with me as a mentor.
4.1.1 Syntax

Actions in Ostari are written in the style of functions in an imperative C-style language, as seen in Listing 4.1 above. The grammar for action definitions can be found in figure 4.1. Each action can have one or more parameters, some or all of which can be secret, which means they are only known to a subset of the agents. When the action is executed, the action will appear to each other agent as a non-deterministic choice between all possible assignments of the secret parameters, while to the agent or agents aware of the values of the secret parameters it will appear as having the actual values of these parameters. Notably, the agent aware of the values of the secret parameters can also be a parameter to the action, which is useful to express actions where the agent performing it is aware of the choice of the parameters whereas other agents only know that an action occurred, but not with which parameters.

Actions act on objects, which are simply constants that are organized in finite sets called types. The statements that an action can contain operate on properties rather than on predicates. Properties are named attributes of an object or tuple of objects that may or may not be assigned a value. Formally, they represent partial functions mapping from the cross product of some types to another type. Actions can set the values of properties using standard assignment syntax such as at(x, 1) = AceOfSpades. In this example, the property at maps from the product of the type Players and the type HandIndex to the type Card. Properties need not be set to any value, which is represented syntactically by the special null value.

The possible statements in an action can be grouped into four types:

- As described above, assignment statements are written using an equals sign with a property expression on the left hand side, and any term (constant or property) on the right hand side. The property on the left hand side will be assigned the value of the right hand side.

- The control flow statements are a standard if/else construct, and a precondition statement. The latter defines in which worlds an action is applicable, which is particularly relevant for non-deterministic actions, or the appearance of actions to agents when secret parameters are present.

- Epistemic statements are used to convey information to agents. The learn and suspect statements are used to inform an agent about something in the world. The difference between them is that learn will tell them truthful information, i.e. it will only apply in worlds in which the condition that is conveyed holds, whereas suspect can be used to convey any information, true or not. public, finally, is a modifier for other statements. By default, statements are transparent, i.e. agents will not be aware of them happening. public makes an agent or set of agents aware of a statement happening. Note that a
public if statement will tell the agent which branch of the if statement was taken, which means they will (truthfully) learn whether or not the condition holds.

• Finally, it is possible to use in Baltag’s action syntax by enclosing it in angle brackets as a statement.

Note that the condition conveyed by the learn and suspect statements can contain the belief modality and quantifiers, but there are also two special forms using what I call epistemic quantifiers:

• learn Which acts like an existential quantifier in that it will inform the agent whether or not there is an object that satisfies the given condition, but additionally also tells them which object that is. For example, learn (x) Which c in Cards: top(deck) == c will inform agent x which card is at the top of the deck. If there is more than one object satisfying the given condition, the resulting action will be non-deterministic and result in multiple epistemic states, one for each such object.

• learn Each behaves similar, but instead of telling the agent which object satisfies the condition, it will tell them for each object of a type whether or not it satisfies the condition. For example, the Hanabi hint action in listing 4.1 contains the statement learn (p): Each i in HandIndex: color(at(p, i)) == c which will tell the player for each hand index whether or not the card at that index has the given color c.

The semantics of actions, the statement types and these two epistemic quantifiers will be provided in the next section in terms of how they are compiled to Baltag actions.

4.1.2 Compilation

Each Ostari action application can be compiled to a Baltag action as follows: Each public parameter is replaced with the passed value for the parameter throughout the action, while a copy of the action for each possible assignment of values to the secret parameters is made, with the parameters replaced by the corresponding values. These actions are joined by a non-deterministic choice and this entire sum is wrapped in the suspicion operator for all agents except the agents that are aware of the value of the secret parameters. The copy of the action corresponding to the actual parameter values is also wrapped in a mutual, truthful learning operator for the agents aware of their values and composed with the sum. This means, if a(o) is the action, with values o1, o2 and o3 as possible values for o, x is the agent that is aware of the actual value o1 and y is another agent that does not know the value of the parameter and F(o) is the compiled version of the action body, the resulting action will be \((F(o_1) + F(o_2) + F(o_3))^y \cdot (F(o_1))^x\). Informally
\( \langle \text{action} \rangle ::= \langle \text{identifier} \rangle (\langle \text{parameter} \rangle^*) \langle \text{statement} \rangle \)

\( \langle \text{parameter} \rangle ::= \langle \text{identifier} \rangle : \langle \text{type} \rangle \\
| \langle \text{identifier} \rangle (\langle \text{player} \rangle^+) : \langle \text{type} \rangle \)

\( \langle \text{statement} \rangle ::= \langle \text{Baltag} \rangle > \\
| \langle \text{property} \rangle = \langle \text{nullterm} \rangle \\
| \text{learn} (\langle \text{player} \rangle^+) \langle \text{fact} \rangle \\
| \text{suspect} (\langle \text{player} \rangle^+) \langle \text{fact} \rangle \\
| \text{public} (\langle \text{player} \rangle^+) \langle \text{statement} \rangle \\
| \text{if} (\langle \text{condition} \rangle) \langle \text{statement} \rangle \text{else} \langle \text{statement} \rangle \\
| \text{precondition} \langle \text{condition} \rangle \\
| \{ \langle \text{statement} \rangle^* \} \)

\( \langle \text{nullterm} \rangle ::= \text{Null} \\
| \langle \text{term} \rangle \langle \text{term} \rangle ::= \langle \text{property} \rangle \\
| \langle \text{constant} \rangle \)

\( \langle \text{property} \rangle ::= \langle \text{identifier} \rangle (\langle \text{term} \rangle^*) \)

\( \langle \text{fact} \rangle ::= \langle \text{condition} \rangle \\
| \text{Each} \langle \text{identifier} \rangle \text{ in} \langle \text{type} \rangle : \langle \text{condition} \rangle \\
| \text{Which} \langle \text{identifier} \rangle \text{ in} \langle \text{type} \rangle : \langle \text{condition} \rangle \)

\( \langle \text{condition} \rangle ::= \langle \text{property} \rangle \text{ == } \langle \text{nullterm} \rangle \\
| \langle \text{property} \rangle \text{ != } \langle \text{nullterm} \rangle \\
| \text{Forall} \langle \text{identifier} \rangle \text{ in} \langle \text{type} \rangle : \langle \text{condition} \rangle \\
| \text{Exists} \langle \text{identifier} \rangle \text{ in} \langle \text{type} \rangle : \langle \text{condition} \rangle \\
| \langle \text{player} \rangle : \langle \text{condition} \rangle \\
| \langle \text{condition} \rangle \text{ or} \langle \text{condition} \rangle \\
| \langle \text{condition} \rangle \text{ and} \langle \text{condition} \rangle \)

Figure 4.1 Grammar for action definitions in Ostari

this means that \( y \) sees the action as one of multiple options, one for each possible parameter assignment, but \( y \) sees the actual action, and that is the action that is also actually executed.

After the parameter replacement is performed, the statement that comprises the body of the action is compiled as shown in table 4.1. Since the control flow statements are composed of other statements, and some statements contain conditions, this compilation is done recursively. Conditions are compiled to the corresponding equivalent in Baltag, except for the quantifiers, for which no equivalent exists. However, because each quantifier names the type of the variable that
Table 4.1 Translation of Ostari macros to Baltag actions. Lower case letters are placeholders for properties, conditions or statements, with the upper case equivalents in the right column referring to the compiled version of that property, condition or statement.

<table>
<thead>
<tr>
<th>Ostari</th>
<th>Baltag</th>
</tr>
</thead>
<tbody>
<tr>
<td>f(o) = p</td>
<td>$(\sum_{p'} ?F(o,p') \cdot \text{flip } F(o,p') + \prod_{p'} ?\neg F(o,p')) \cdot \text{flip } F(o,p)$</td>
</tr>
<tr>
<td>learn (x) c</td>
<td>$(?C)^x$</td>
</tr>
<tr>
<td>suspect (x) c</td>
<td>$(?C)^x$</td>
</tr>
<tr>
<td>public (x) s</td>
<td>$(S)^x$</td>
</tr>
<tr>
<td>precondition c</td>
<td>$?C$</td>
</tr>
<tr>
<td>if (c) s else t</td>
<td>$(?C \cdot S) + (?\neg C \cdot T)$</td>
</tr>
<tr>
<td>{ s1; s2; ... }</td>
<td>$\prod_i S_i$</td>
</tr>
</tbody>
</table>

is quantified over, and these types are finite sets, the condition is compiled to a conjunction (for Forall) or disjunction (for Exists) over all possible values for that variable.

The special epistemic quantifiers Which and Each are compiled slightly differently from other conditions:

- **learn (x) Which v in Vs: c(v)** is compiled to a non-deterministic choice, with one option for each value in Vs. Each of these options contains the compiled condition, with v replaced with the corresponding value. Each of these options is then separately wrapped in the (mutual, truthful) learning operator. The result is a Baltag action of the form $\sum_v (?C(v))^x$. In other words, instead of wrapping the whole condition in the learning operator, each option of the sum is wrapped individually. This has the effect of informing the agent of which option was chosen.

- **learn (x) Each v in Vs: c(v)** is compiled to a sequence of actions, each of which is a non-deterministic choice between two options: $?C(v)$ and $?\neg C(v)$. As before, each of these choices is wrapped in the (truthful, mutual) learning operator individually, resulting in $\sum_v ((?C(v))^x + (?\neg C(v))^x)$. This has the effect of informing the agent for each value whether the condition holds or not, because only one of the two option will apply to the actual world.

Finally, at the core of all assignment statements and conditions are properties, and these also need to be compiled to Baltag. Generally, a property $f(o) = p$ is compiled to a predicate $F(o,p)$, and of all possible values for p the predicate will only hold for at most one value. As table 4.1 shows, to set a property, the first step is to find the value it is currently set to, and flip the truth value for that value, effectively unsetting the property. Additionally, because the property may already be unset, there is an additional check to see if the predicate does not hold for any value.
of \( p' \). After this, the truth value for the newly set value can simply be flipped. However, this
glosses over the fact that properties may be nested, and the right hand side of the assignment
may also be a property expression. First, to resolve nested properties, temporary variables are
introduced that take on the value of the inner property. For example, \( f(g(o)) = p \) may also be
written as \( v = g(o) \) and \( f(v) = p \), i.e. \( v \) represents the value that \( g(o) \) currently has. Because
the value of \( v \) is not known a priori, this requires a non-deterministic choice between all possible
values in any place where \( F \) is referenced in the compiled action. Of course, there may be even
more nested properties as parameters to \( g \), which are processed analogously, and also require
non-deterministic choices between all possible values of their corresponding variables. In the
general case, if the nesting is \( f_n(f_{n-1}(\cdots f_2(f_1(o))\cdots)) \) the resulting formula contains nested
sums for all temporary variables resulting from resolving these nestings, resulting in a formula of
the form

\[
\sum_{\vec{v}} F_1(o, \vec{v}_1) \cdot \prod_i F_i(\vec{v}_i, \vec{v}_{i+1}) \cdot F_n(\vec{v}_n, p')
\]

to check the current value of the property, where the sum over \( \vec{v} \) ranges over all possible
combinations of assignments to the variables in the vector \( \vec{v} \). To set the new value of the property,
the last term is replaced with a flip operation. Properties on the right hand side can be viewed
as nesting of properties starting with a constant symbol, and are thus resolved analogously by
the introduction of temporary variables, with each assignment of values to these variables as
one choice, and only that choice will be applicable in the end. It should be noted that this
approach results in a number of terms that is exponential in the nesting depths. When I describe
the implementation details below, I will also detail how my implementation avoids actually
evaluating exponentially many terms. First, though, I will describe how actions can actually be
executed and their results observed.

4.1.3 Execution Environment

In addition to the macros used to define actions, Ostari also provides a runtime environment to
execute these actions. Each Ostari file consists of multiple sections:

- The `types` section is used to define all constants and which types they have. Each type
definition consists of the type name and a comma-separated list of constants that have
that type. Constants can be listed in arbitrarily many such lists, i.e. each constant can
have multiple types, and there is no explicit relationship between the types.

- The `properties` section contains the definitions of all properties that are used, including the
types of their domains and values. Because properties are (partial) functions, the syntax
used for this is the same as the one used in Haskell for function definitions: The name of the function is followed by two colons, and the types of the parameters and value are separated by the symbols \( \rightarrow \). Additional limitations placed on the property can be provided in parenthesis. This is important for some optimizations that are described in the next section.

- The **actions** section contains the action definitions for the domain, such as game actions used in Hanabi.

- The **initial** section defines the initial epistemic state to which actions will be applied. This state is defined as a set of worlds, with property assignments for each world, and which other worlds the world looks like to each agent.

- The **execute** section constitutes the actual execution sequence of the program. It is executed line by line, with each line either printing information about the current epistemic state, or taking that epistemic state and transforming it into a new one, usually by the application of actions. The supported operations for this section are described below.

- The **game** section is used to provide additional information to the game playing capabilities of the system, such as which game actions are allowable in which phase of the game. All constraints defined in this section could also be expressed in the actions themselves, but by providing them externally the system can use them for optimizations.

- The **goal** section contains definitions for goal conditions the agents should work towards in a game. Each goal condition consists of optional bindings, an activation condition and a goal. Agents are presented with all goals for which the activation condition holds. In practice, the activation condition is used to limit goals to be adopted by particular players, or in particular phases of the game.

When the system reads an input file, it will create the initial state from the specification in the initial section and then start executing the commands in the execute section line by line. The valid operations that can be used in the execute section are:

- Executing an action, by using C-style function call syntax, e.g. `hintcolor(b, red)`. This will compile the action as described in the previous section and apply it to the current state, producing a new state or set of states if the action is non-deterministic. The system can be configured at compile time to either continue with all of the resulting states, applying all future operations to each of them, or to discard all but one of them.
• An agent suspecting an action, by prefixing an execution with $x$ suspects, where $x$ can be any agent. For example, Sherlock suspects kill(Moriarty, victim) can be used to have Sherlock believe that the kill action is happening with the given parameters without actually performing the action. This is executed by compiling the action and wrapping it in the suspicion operator.

• Printing the Baltag formula corresponding to a compiled action. This is done by prefixing the action call with the directive print:. Since these formulas often contain several hundred terms, this output is mostly useful for debugging purposes by reducing actions to potentially problematic parts and inspecting the compiled version.

• Printing the actual world or parts of the state. These operations are also prefixed by the print: directive, followed by facts, model, or stats. The facts mode prints everything that holds in the actual world, which are all facts that are considered to be objectively true. model also prints everything that holds in the actual world, and additionally, for each agent, all worlds that they consider possible by listing everything that holds in these worlds. Basically, facts prints the actual world, and model prints the actual world and all worlds that are one “hop” away for each agent in the graph representation of the epistemic state. stats, finally, only prints how many worlds each agent considers possible, which is easier to read than the full list of worlds, and useful for debugging purposes.

• Querying the epistemic state is supported with the query: directive, which can be followed by any valid condition. This condition will be evaluated in the current state and True or False will be printed depending on whether the condition holds or not. Additionally, the epistemic quantifiers Which and Each, which will cause the query statement to print for which element of a set the condition holds (if any), or print for each element of a set if the condition holds with the quantified variable substituted for that element.

• The goal: directive can be used to have the system search for a sequence of actions that lead to a condition being satisfied. Once such a sequence is found, the actions in it are executed in the current state and the resulting state will become the new current state. Section 4.1.6 discusses this operation in more detail.

• Playing a game is also a supported operation. It uses the play: directive followed by player names and types and will lead the players through the game according to the definitions in the game section. The system takes care of passing the turn between the players, lets them choose which of the currently valid actions to perform and applies their chosen action to the current state. After game play has concluded, the resulting state will become the new current state so it can be inspected by the user. This operation provides the environment in which the intentional agents in section 4.2 play games.
An example input file, corresponding to the game One Night Ultimate Werewolf can be found in appendix A.1.

4.1.4 Belief Quality

As described in the previous section, Ostari supports querying the epistemic state by testing whether an arbitrary condition holds, or enumerating all worlds agents consider possible. However, binary yes/no answers are often not detailed enough to determine anything interesting about an epistemic state, while a list of possible worlds is potentially long and unmanageable. Instead, some questions are better answered on a scale. For example, rather than determining whether an agent believes a certain condition, i.e. whether it holds in all worlds they consider possible, it may be more interesting in how many worlds or what percentage of these worlds it holds. I call this measure the quality of a belief, and it ranges from 0, meaning that the condition holds in no worlds, to 1, meaning the condition holds in all worlds, i.e. the agent believes the condition. On a basic level, Ostari allows querying the quality of a belief by using the quality: directive in the same way as the query: directive described above. However, because this measure is also useful in any place where conditions are allowed, such as in actions, I also added a new quantifier Q. It is used to determine whether the quality of some belief is above or below a certain level. Syntactically this is written as \( Q[\text{op value}] (x) \phi \), where \( \text{op} \) is \( > \) or \( < \), \( \text{value} \) is any floating point constant between 0 and 1, and \( x \) is any agent. This new quantifier is evaluated as follows:

\[
\mathcal{M} \models Q_{\geq v,x} \phi \iff \frac{\{M' \in \mathcal{M}_x, M' \models \phi\}}{|\mathcal{M}_x|} \geq v
\]

\[
\mathcal{M} \models Q_{\leq v,x} \phi \iff \frac{\{M' \in \mathcal{M}_x, M' \models \phi\}}{|\mathcal{M}_x|} \leq v
\]

In the degenerate case where \( \mathcal{M}_x \) is empty, i.e. when the agent considers no worlds to be possible, the fraction is considered to evaluate to 0.

The quality measure, and the corresponding quantifier, can be interpreted similar to probabilities. However, they only constitute true probabilities, if each world is considered equally probable. In practice, the agents may gain access to knowledge that causes them to become more skeptical of certain scenarios/worlds. In Ostari, worlds can be marked as being deemed less likely by an agents by assigning them weights, which are real numbers, initialized with zero, where higher values mean that the world is considered less likely. As an extension of the quality modality, I introduced a weighted quality modality, \( W[\text{op value}] (x) \phi \). To evaluate this quantifier, there needs to be a formal way to associate a world with a weight, which I denote as \( |w| \), or, using the same abbreviations as for other properties of the actual world of a state, \( |\mathcal{M}| \) for the weight of the actual world of \( \mathcal{M} \). This weight can then be used to evaluate the weighted
quality quantifier as follows:

$$\mathcal{M} \models W_{>v,x}\phi \iff \frac{\sum_{M' \in \mathcal{M}_x, M' \models \phi} 1}{1 + |M|} \geq v$$

$$\mathcal{M} \models W_{<v,x}\phi \iff \frac{\sum_{M' \in \mathcal{M}_x, M' \models \phi} 1}{1 + |M|} \leq v$$

What this definition basically does is to calculate the ratio of worlds in which the condition holds to that of all worlds, where each world $w$ is weighed with $\frac{1}{1 + |w|}$. This means that a world with weight 0 (the default) is considered twice as likely as a world with weight 1. For example, if an agent considers 2 worlds possible, one in which $p$ holds, and one in which it does not, the quality $Q$ of a belief in $p$ would be 0.5, but the weighted quality would depend on the weights of the world. If the world in which $p$ holds has weight 0, while something made the agent suspicious of the world in which it does not, and they therefore assign it a weight of 1, the weighted quality of the belief in $p$ would be:

$$\frac{1}{1 + 0} + \frac{1}{1 + 1} = \frac{1}{2} + \frac{1}{2} = \frac{2}{4} = \frac{1}{2}$$

To be able to actually use this weighted quality, agents need to be able to assign weights to worlds. Ostari supports an `addWeight(f)` statement, where $f$ can be any floating point constant. A typical use case for this statement is inside an if statement, in order to mark a subset of worlds as less likely. As an example, listing 4.2 contains an Ostari encoding of an action from the One Night Ultimate Werewolf domain described in detail in chapter 5. It encodes the effect a player claiming to have a particular role has on the beliefs of the other players. Note that the action has three parameters, one corresponding to the player making the statement, another to the role they claim to have, and the third to the role they actually have, which is enforced by the precondition. This third parameter is secret and only known to the player performing the action, though, and all other players will consider all possible alternative actions, and apply them to all worlds they consider possible. In each world in which the role of the acting player matches the role they claim to have, nothing happens, while the weight of each world in which the two do not match is increased by 1. For more details on how to interpret this behavior, and how this information is actually used by the agents refer to chapter 5.

Finally, I need to address how this new syntax is compiled to Baltag. The quality and weighted quality quantifiers, as they are defined above, already constitute an extension to Baltag, and their compilation works in the same way as the compilation of the $\square$ operator to the $\square$ quantifier, where the onus is on the compilation of the properties contained in the condition. The `addWeight` statement, though, does not have a straightforward mapping to Baltag. In principle,
it is possible to use a special property called weight, and compile addWeight to an emulation of an addition operation, but doing so is highly inefficient and cumbersome. Instead, I opted to extend the definition of pointed models by adding the \cdot operator described above, which means that the implementation stores the weight of a world explicitly. Analogous to the addWeight statement in Ostari I then also added an addWeight operation to the Baltag action language. The semantics of this operation require the addition of a weight change field to the simple actions described in section 2.3.3, which is not used by any other action type. addWeight(v) then results in an action model that has its weight change set to v, with no precondition or appearance, and when applied to a world results in a new world with its weight increased by v. This optimization allows weights to be updated efficiently, and the semantics of the quality and weighted quality quantifiers enable querying these values in Ostari. In the next section I will describe several additional optimizations I performed to make action execution actually tractable.

4.1.5 Optimizations

During the implementation of the language and execution environment described above, I discovered several performance bottlenecks that were prohibitive for using the system to model non-trivial domains. While some complexity is inherent to the logic, such as a potentially large number of possible worlds, there are several opportunities to reduce the CPU and memory requirements. Some of these techniques modify how the compilation is done to exploit properties of the chosen implementation language, Haskell, while others are focused on the internal representation of states and optimize for common cases. Because of the correspondence between actions in Ostari and Baltag is formalized, it is also possible to execute actions without compiling them first. In this section I will describe all of these implementation details, starting with a compilation level optimization of nested properties, followed by several representation level optimization and a description of uncompiled execution.
Listing 4.3 Haskell data type for an epistemic state, which Baltag also calls a pointed model. It consists of three fields: The appearance function, a set of atomic propositions that holds in the actual world, and the list of all valid agents.

4.1.5.1 Nested Properties

Since properties form basic unit of abstraction in Ostari, and are used in many places, their compilation to Baltag is particularly impactful. However, recall that the number of terms in formulas resulting from nested properties grows exponentially in the number of nesting levels. Since nesting properties several levels is not uncommon, this would result in potentially intractable compilation times, before the action could even be executed. However, Ostari is implemented in Haskell, which evaluates expressions lazily and this feature is used in several places. First, the way epistemic states are represented uses the same systematic ambiguity that Baltag uses in his paper. A state consists of a world, which defines what holds in that world, and an appearance function that defines which other worlds the actual world looks like to each actor. However, each of these worlds, in turn, has an appearance to each actor, and so on. Instead of explicitly enumerating all worlds, the internal representation of epistemic states only contains the actual world and an appearance function. This way, each other world can also be viewed as if it was a state in which that world is the actual world. This means, an epistemic state consists of a set of atomic propositions that hold in its actual world, and a function that maps actors to other epistemic states. Listing 4.3 shows the Haskell data type used to represent this. Because of how the lazy evaluation works the appearance function will only ever be called, if the appearance of a world needs to be obtained, e.g. when the user wants to inspect it in some way. This results in a representation that can describe potentially very large epistemic states without having to evaluate them unless necessary.

Nested properties and the resulting exponential number of terms are addressed in two ways. One of these is reordering of terms to reduce the number of terms that need to be evaluated. As described above, the formula for checking the value of such a nested property is \( \sum_{\vec{v}} F_1(o, \vec{v}_1) \cdot \prod_i \sum_{\vec{v}_i} F_i(v_i, v_{i+1}) \cdot \sum_{\vec{p}} F_n(\vec{v}_n, \vec{p}'). \) However, the order in which the sum is evaluated is irrelevant, and can therefore be chosen as needed. Because the sequence operation distributes over the non-deterministic choice operation, the check for any of the variables can also be pulled outside the sum. Using this, my system places the variable that has the largest number of possible values first, resulting in \( \sum_{v_1} F_1(o, v_1) \cdot \sum_{\vec{v}} \prod_i \sum_{\vec{v}_i} F_i(\vec{v}_i, \vec{v}_{i+1}) \cdot \sum_{\vec{p}} F_n(\vec{v}_n, \vec{p}'). \) Because at most one of the \( F_1(o, v_1) \) can hold at any time, this means only one copy of the inner sum.
needs to be evaluated. This inner sum can be resolved in the same way, though, and this way,
instead of requiring the evaluation of an exponential number of terms, only linearly many need
to be evaluated. Also, because Haskell evaluates terms lazily, the other terms never need to be
generated in the first place, resulting in linear run time as well, plus the overhead the Haskell
run time requires to keep the unevaluated chunks around. The second way the number of terms
is reduced is by the observation that some properties never change, or only have a limited set of
possible values. For example, a property color of a card will not change in most card games.
Similarly, in Hanabi, a property top that states which card is on top of a stack only has a
limited range of valid values. For example, only the red cards can be on top of the red stack,
but the signature of the property is \textbf{Stacks} -> \textbf{Cards}. When defining a property, it is possible
to list which values are valid assignments for which arguments. For properties such as color
that means that every card will be assigned its color statically, while the top property will list
the subset of cards that can be on top of each stack. I call such properties static or restricted,
respectively. When a property is compiled, terms corresponding to invalid assignments to static
or restricted properties can be skipped. For static properties even the check for its value can be
omitted, since the property can only have one value. This means, in the formulas of the form
\[ \sum_{v_1} F_1(o, v_1) \cdot \sum_{v'} \prod_i F_i(v'_i, v'_i+1) \cdot F_n(v'_n, p'), \]
any \( F_i \) that correspond to restricted properties
will be eliminated at compile time if their arguments are not listed in the list of valid values.

4.1.5.2 State Representation: Integers and String

The two main operations that are performed during action execution are checking whether a
particular term is set, and flipping a term. Both of these cases access the set of atomic propositions
in some possible world. While the set operations themselves are fairly efficient, a large percentage
of the time they need is actually spent on string comparisons, because they are linear in the
length of the strings involved. However, each Ostari program has a known and finite set of
valid strings that can show up, consisting of all set names, constant names, property names and
variable names. As a preprocessing step, Ostari therefore collects all of these strings and assigns
a unique integer ID to each one. The internal representation of states and actions then uses these
integers instead of strings. While this technically limits the number of allowed names in a single
Ostari program to the range of integers, this is not a limitation that is likely to be relevant in
practice. An actual limitation lies with debugging of the system, since internal states are harder
to inspect when they use integers instead of meaningful names. However, the execution code was
written in a generic way, so that the ID replacement can be turned off for debugging purposes.
Additionally, there are conversion functions between the string- and integer-versions available for
all internal data structures.
4.1.5.3 Property Indices

As mentioned above, checking whether a term is set or not, and flipping the truth value of terms are the two most common operations, and therefore benefit most from optimizations. However, for every flip there are typically many checks. Additionally, to manipulate agent beliefs, often only checks are used to eliminate worlds that the agent should no longer consider possible. With the current representation of worlds as sets of propositions, checking a single proposition takes time that is logarithmic in the size of the world, i.e. the number of propositions that hold in it. Similarly, adding or removing elements from the set, which corresponds to flipping terms, takes time that is logarithmic in the size of the set. For many properties, this equal cost of changing and checking constitutes an ideal balance, but in actual games there may be properties that are checked substantially more often than they are changed, especially in the beliefs of agents. As a concrete example, consider the role card a player is dealt during a game of One Night Ultimate Werewolf. While there are a few roles that can exchange cards between players, the majority of the game consists of the discussion phase, during which players only reason about which roles players have or could have, without actually changing this assignment. To make this reasoning process more efficient it would therefore be beneficial to make checks faster, even if that happens at the expense of manipulation actions.

To improve the performance of accessing certain properties Ostari stores them in arrays. Rather than performing a lookup in a set, checking such a proposition then only consists of a single array access, which is a constant time operation. However, because data structures in Haskell are immutable, flipping a proposition that is stored in an array requires a copy of the entire array to be made, which is an \( O(n) \) operation\(^1\). The way propositions are indexed is to store them in a two-dimensional array, where one index corresponds to the property and the other to the parameter, and the content of the cell consists of the property value, or a special Null value. Note that this limits indexing to propositions that are derived from properties with exactly one parameter. However, since such properties correspond to attributes of objects, they are the most commonly used properties, and thus this optimization results in large performance gains in practice. The current implementation has the benefit of simplicity, but it could in principle be extended to properties with more than one parameter if that becomes necessary in the future.

The decision which properties to index this way depends on the application, and has therefore be made by the domain author. Because of the semantic correspondence between attributes and properties with one parameter, defining the index is done by defining a set of objects, which then will constitute the indices along one dimension of the array. The system will then index all

---

\(^1\)While this seems like a drawback of Haskell as an implementation language, note that this is necessary to keep access to a copy of the state as it was before an action is executed. Since the majority of action applications occur during planning, which is basically speculative execution, copies of the entire state would have to be made in any case.
properties that are defined only on a subset of these objects. As a practical example, in One Night Ultimate Werewolf I defined all players to be indexed. This causes exactly the properties that are defined on players or a subset thereof to be indexed. The resulting array would then consist of one column per player, and one row for each of these properties. In other words, each column corresponds to what might be considered a “struct” representation of a player in other programming languages.

4.1.5.4 Common Knowledge

A key feature of Dynamic Epistemic Logic and Ostari is the capability to reason about differences in knowledge between different characters. However, when modeling actual application domains, there will be several properties for which such reasoning is not necessary. For example, in a game like One Night Ultimate Werewolf, some properties are secret, but others, such as which topic is currently being discussed, are known to all players. Or, in other words, when any such property changes, it also changes in every possible world of all agents. Rather than keeping redundant copies of such properties and actually performing this update in every possible world, they are collected in their own “world”, and all other worlds contain a reference to this single copy. When checking whether a proposition holds, the system then has to check the referenced world containing the common knowledge in addition to the set of facts or index array of the world.

The declaration of a property as common knowledge is done when a value is assigned to it, by using the prefix modifier common. For example, \texttt{common topic(p) = role} will set the role property of \texttt{p} to \texttt{role} for all players. This is done by updating the shared world once, which creates a new copy of it, and updating the references to the new shared world in all worlds.

4.1.5.5 Direct Execution of Ostari Actions

While the optimizations presented so far allow moderately complex domains to be encoded and executed in reasonable time in Ostari, the compilation of properties to predicates still greatly limits the size and complexity of domains that are tractable. For example, while the handling of nested properties described in section 4.1.5.1 reduces the number of necessary lookups to be linear in the number of nested levels, each individual lookup requires a number of checks that is linear in the number of possible values for the property, and each of these checks may require time that is logarithmic in the size of the world, if the property in question is not indexed. If properties were stored in a proper data structure directly, instead of being represented as a set of possible atomic propositions, only one lookup in that data structure would have to be performed, such as a single lookup in a dictionary/map. Indexed properties, described in section 4.1.5.3, are actually designed to store the value of a property, and can be accessed in $O(1)$. This means,
data PointedModelA a b = PMA {
    appearanceA :: Map.Map a [PointedModelA a b],
    factsA :: Map.Map a (Map.Map [a] a),
    commonA :: Map.Map a (Map.Map [a] a),
    indexedA :: b,
    agentsA :: [a],
    weightA :: Float,
    contextA :: Context
}

Listing 4.4 Haskell data type for an abstract epistemic state, which Baltag also calls a pointed model, consisting of an appearance map, content of the actual world, a reference to common knowledge, an index, the available agents, the weight, and a reference to domain information needed to convert between integer and string representations.

where the compiled version of e.g. an equality test \( f(x) == g(y) \) would perform a number of index accesses equal to the number of possible values for \( f \) and \( g \), evaluating this entire equality directly could be done in \( \mathcal{O}(1) \).

As a major performance improvement, I implemented a mode of executing Ostari actions directly, without compiling them to Baltag first. In this mode, the epistemic state is stored in a data structure that is more closely aligned with the Ostari way of representing properties, as seen in listing 4.4. Note that the data type is parameterized with two types, \( a \) and \( b \), where \( a \) is the primitive type for constants, which are either integers or strings, and \( b \) is the type of the index to use. This second parameter is necessary because integers can actually be stored in special unboxed arrays, making accesses to them even faster by removing another level of indirection. The content of the actual world of the state is stored in \( \text{factsA} \), which is a nested map with the first layer using the property name as the key, and the second the parameters of the property. This was done to improve efficiency at the expense of complexity and some additional memory overhead. Because maps, as other data structures, are immutable, all inner maps corresponding to properties that remain unchanged by an operation can be reused, while only the map corresponding to the property for which a value changes, as well as the outer map have to be updated. Common knowledge properties are stored in the field \( \text{commonA} \). The field \( \text{appearanceA} \) contains one entry for each agent, corresponding to which worlds that actor considers possible. Like Baltag, my implementation represents worlds as states, i.e. each of the worlds an agents considers possible is represented by the \( \text{PointedModelA} \) type, and contains its own appearance map, facts, etc.

As mentioned earlier, the type of the index array is a parameter to the \( \text{PointedModelA} \) type, and is usually dependent on the type used for \( a \). When \( a \) is the \text{String} type, ordinary arrays are used for the index, but when \( a \) is \text{Int} it is possible to use a special unboxed array type, that stores integer arrays as-is in memory, resulting in very fast access times. This parameterization
also makes it possible to use other data types, such as \texttt{DiffArray}\textsuperscript{2}. The weight of the actual world of the state is stored directly as a float to make weight updates faster.

States of type \texttt{PointedModelA} can be converted to Baltag epistemic states by converting the properties contained in \texttt{factsA}, \texttt{commonA} and \texttt{indexedA} to predicates and populating the corresponding fields in a \texttt{PointedModel} data structure, and performing the same operation recursively for all worlds in the appearance map. Converting an epistemic state from Baltag to Ostari can be done if and only if each predicate satisfies the property invariant, meaning that for an \(n\)-ary predicate at most one proposition for every assignment of the first \(n-1\) parameters holds. For example, if \(at(x,\text{library})\) and \(at(x,\text{theater})\) are both elements of the content of any single world of an epistemic state, the conversion can not be done, because the value of the property \(at(x)\) would be ambiguous. Actions that are compiled from Ostari to Baltag maintain this invariant, but since it is possible to use arbitrary Baltag actions as statements in Ostari actions, this can not be guaranteed. Currently it is therefore necessary to choose between execution in Ostari directly, which does not allow the use of arbitrary Baltag actions, or compiled execution for the entire program. In the future it would be possible to check this condition and allow switching back and forth between the two modes as needed.

When an Ostari action is called, the parameters passed to it are used to transform it to an intermediate representation, consisting of the actual action, and the action options induced by the secret parameters, if any, similar to how the action would be compiled to Baltag. Then statements in the body of the actual action are executed, followed by the options considered possible by the agents applied to the worlds they consider possible. When executing a statement, it is applied to a list of states, and produces a new list of states from that, as follows:

\begin{itemize}
  \item \(f(x) = y\) produces a new state for each state where the \texttt{factsA}, \texttt{indexedA}, or, for the actual world only, \texttt{commonA} data structures are updated to reflect the assignment.
  \item \texttt{learn} \(x\) \(c\) produces a new state for each state in which \(c\) holds, where the \texttt{appearanceA} map is updated such that only states in which \(c\) holds are retained for agent \(x\).
  \item \texttt{suspect} \(x\) \(c\) produces a new state for each state where the \texttt{appearanceA} map is updated such that only states in which \(c\) holds are retained for agent \(x\).
  \item \texttt{public} \(x\) \(s\) produces a new state for each state where \(s\) is applied to the state, as well as to each state in the appearance of the actual state for \(x\).
\end{itemize}

\textsuperscript{2}\texttt{DiffArray} is an experimental variant of arrays that supports constant-time updates, at the expense of making accesses to previous versions of the array slower. Unfortunately, the current implementation in the Haskell package repository is still experimental, does not perform well in practice, and appears to not be actively developed at the moment.
• **precondition** \( c \) filters the list of states, removing those in which \( c \) does not hold, and produces this filtered list as the result.

• **if** \( (c) \) **s** else **t** applies either **s** or **t** to each state, depending on whether \( c \) holds.

• \{ **s1**; **s2**; ... \} applies **s1** to each state, then applies \{ **s2**; ... \} to each resulting state.

As can be seen, the execution of statements maps directly onto the data structure used to represent the state, resulting in a very efficient execution of actions. The main limitation of this approach is that the number of worlds can grow very large in certain domains. While this implementation is suitable for One Night Ultimate Werewolf, and several other domains, there are more opportunities for optimization in future work that I will discuss in chapter 7. I will now describe how the system supports searching for action sequences to achieve goals, using this reasonably efficient action execution mechanism.

### 4.1.6 Planning

One key functionality provided by the Ostari system is automatic reasoning about which actions should be taken to achieve some condition. This is very similar to classical planning, where, given an initial state and a list of possible actions, the task is to find a sequence of actions such that a given goal is reached. However, while classical planning typically uses first order logic as the underlying formalism, Ostari provides the means to define epistemic actions, and planning capabilities using agents' beliefs as part of the initial state, action and goal definitions. Because the planning process involves agents' beliefs, actions may also only be suspected by agents without actually happening, and the system also needs to take this option into account. If the problem is viewed as a tree, where the current epistemic state is the root node, edges represent possible actions, and further nodes represent the state after applying an action, a naive approach to the problem would be to perform a breadth-first search starting at the root until a node is found at which the goal condition is satisfied. Suspected actions can be taken into account by introducing additional branches in the tree corresponding to them. However, an agent may suspect that another agent suspects that an action is happening, and these doubly nested suspected actions would also have to be added. Because there is no limit on the number of nested suspicions in theory, this would result in infinitely many branches, which makes breadth-first search impossible. Even when nested suspicions are not taken into account, the branching factor is \( n + n \cdot m \), where \( n \) is the number of (ground) actions and \( m \) the number of agents, which is prohibitively large for many problems. For example, the Hanabi hint action actually represents \( 5 \cdot (m - 1) \) different actions, because the agent can choose any of the 5 colors and any of the \( m - 1 \) other players. For 3 players, this one action alone would result in a branching factor of 40,
because it can be resolved to 10 different actions, and each of the 3 agents could suspect any of these to happen, in addition to the action actually happening.

Because the problem at hand is basically just a variation of the classical planning problem, there are known, better approaches. Instead of applying every possible action at every node, it is possible to use heuristic search by extracting a heuristic from the problem, as described by Bonet and Geffner [Bon01]. To calculate their heuristic they use a cost function $g_s$ that estimates a cost for achieving an atom $p$ in a state $s$:

$$g_s(p) = \begin{cases} 0, & \text{if } p \in s \\ \min_{o \in O(p)}(1 + g_s(\text{Prec}(o))), & \text{otherwise} \end{cases}$$

Note that this cost is based on the STRIPS-formulation of planning, in which actions have Add and Delete lists containing the atoms that they set or unset, respectively. In this case $O(p)$ is the set of actions that contain $p$ in their Add lists, and $\text{Prec}(o)$ is the set of preconditions of an action $o$. Baltag, however, uses the flip operator, which changes the value of an atom instead of stating which atoms are added or deleted. It is still possible to use the same cost function, though: Either $p$ already holds in $s$, in which case no actions need to be considered at all, or it does not, and any action flipping $p$ would set it. However, beliefs present additional problems. Consider the case in which an agent considers two worlds possible: One in which $p$ and $q$ hold, which is the actual one, and one in which neither $p$ nor $q$ hold. If the goal is for the agent to believe $p$, an action that tells that agent the truth value of $q$ would achieve that goal, since it will remove the only world in which $p$ does not hold from what they consider possible. The way Baltag actions actually work present another problem: It might seem reasonable to say that an action requires one of the tests that are made in it, but which one depends on the future state the action is applied in, making it unsuitable for use in the heuristic. Additionally, there is a great number of tests, one for each possible value of a property. To address these issues, rather than using the test operations as they are present in the actions, I only use the predicate names (which correspond to property names) they refer to. Additionally, each action will have three classes of preconditions and effects: The first class corresponds to the actual content change of the action, the second class consists of the changes made to the beliefs of agents, such as suspected assignment statements, and the third class consists simply of which agents’ beliefs are changed. To make the distinction clearer, I will refer to what an action requires and what it references. These terms are written as tuples $((x_i), p, n)$, where $(x_i)$ is a (potentially empty) sequence of agents, $p$ is an optional predicate name, and $n$ is an integer. I will mark empty spaces

---

3This has the consequence that STRIPS actions are idempotent, if they don’t change their own preconditions, while Baltag actions are not necessarily.
in the tuple with $\emptyset$.

An action’s requirements $\text{req}(a)$ are defined as:

\[
\text{req}(\text{flip } p(u)) = \{
\text{req}(?p(u)) = \{(\emptyset, p, 1)\}
\text{req}(a \cdot b) = \text{req}(a) \cup \text{req}(b)
\text{req}(a + b) = \text{req}(a) \cup \text{req}(b)
\text{req}((a)^x) = \{(x; y), p, 2\} | (y, p, n) \leftarrow \text{req}(a)\} \cup
\{(x; y), \emptyset, 3\} | (y, p, n) \leftarrow \text{req}(a)\}
\text{req}((a)^*X) = \text{req}(a) \cup \left( \bigcup_{x \in X} \{(x; y), p, 2\} | (y, p, n) \leftarrow \text{req}((a)^*X)\} \cup
\{(x; y), \emptyset, 3\} | (y, p, n) \leftarrow \text{req}((a)^*X)\} \right)
\]

An action’s references $\text{ref}(a)$ are defined as:

\[
\text{ref}(\text{flip } p(u)) = \{
\text{ref}(?p(u)) = \{(\emptyset, p, 1)\}
\text{ref}(a \cdot b) = \text{ref}(a) \cup \text{ref}(b)
\text{ref}(a + b) = \text{ref}(a) \cup \text{ref}(b)
\text{ref}((a)^x) = \{(x; y), p, 2\} | (y, p, n) \leftarrow \text{ref}(a) \cup \text{ref}(a)\} \cup
\{(x; y), \emptyset, 3\} | (y, p, n) \leftarrow \text{ref}(a) \cup \text{ref}(a)\}
\text{ref}((a)^*X) = \text{ref}(a) \cup \left( \bigcup_{x \in X} \{(x; y), p, 2\} | (y, p, n) \leftarrow \text{ref}((a)^*X) \cup \text{ref}((a)^*X)\} \cup
\{(x; y), \emptyset, 3\} | (y, p, n) \leftarrow \text{ref}((a)^*X) \cup \text{ref}((a)^*X)\} \right)
\]

Of note in these definitions is that an action references its requirements in the epistemic world. This is done because a check of e.g. $p$ in the appearance of an action can actually change the agent’s belief about $p^4$. Additionally, requirements and references of the form $(x, \emptyset, 3)$ are added, that refer to an agent’s belief being changed somehow, to address the case where an agent believes about $p$ can change even when $p$ is not referenced at all. The integers associated with each of the requirements and references represent how direct the change is, with 1 representing a change to the actual world, 2 a change to the epistemic state that directly pertains to a predicate.

---

4 This is also the reason for the term references rather than changes. The operations may not explicitly change anything, but a reference in the form of a test operation may still affect an agent’s beliefs by eliminating possible worlds.
symbol, and 3 to arbitrary belief updates. With this information, the heuristic value for actions relative to a goal can be defined as follows:

\[
h_{(x,p,n)}(a) = \begin{cases} 
  n, & \text{if } (x, p, n) \sim \text{ref}(a) \\
  \min_{(x,p,n) \in \text{ref}(a), a' \in \text{Actions}, (x,p,n) \sim \text{req}(a')} (n + h(x,p)(a')) & \text{otherwise}
\end{cases}
\]

Where the expression \((x, p, n) \sim \text{ref}(a)\) refers to a relaxed set containment operator, specific to the encoding of requirements and references, that accounts for the fact that the requirements or references may contain tuples without a predicate expression. It is defined as \((x, p, n) \sim X \iff (x, p, n) \in X \lor (x \neq \emptyset \land (x, \emptyset, n) \in X)\). The way this heuristic is calculated is by starting with the goal and determining for each predicate in the goal which actions contain a reference to that predicate, and setting the heuristic value for these actions to the appropriate \(n\). For each action that does not reference any of the predicates in the goal, the heuristic value is determined by which other actions require its references. For each reference \((x, p, n)\), any other action \(a'\) that satisfies \((x, p, n) \sim \text{req}(a')\), i.e. that requires that reference, is considered, and the minimum heuristic value among these actions is added to \(n\). Another way to look at this is as a graph, where each node represents one action, and there is a directed, weighted edge from an action \(a\) to an action \(a'\) if and only if there is a reference \((x, p, n)\) of \(a\) that is required by \(a'\), i.e. \((x, p, n) \sim \text{req}(a')\), and the weight assigned to that edge is \(n\). The goal is added to this graph as a dummy action that only has requirements, corresponding to the predicates that occur in the goal, and no references, and connected to all actions that reference these requirements in the same way. The heuristic value of an action is then the length of the shortest path to the node corresponding to the goal. I will describe how this heuristic works in practice in section 4.3.2 below.

Using this heuristic is not the only approach, though. Other heuristic search planners, such as FastForward [Hof01] or FastDownward [Hel06] use other heuristics, but still mostly perform forward search through state space. Alternatively, the system could also reason backwards from the goal, and limit its search to actions that actually achieve the goal. Yet another approach would be to use a least commitment approach searching through plan space as described by Weld [Wel94]. This would have the added benefit that actions need not necessarily be ground before being inserted into the plan, as binding their parameters could be delayed until needed. However, since exploring the intricacies of planning in the presence of epistemic actions is not the main focus of this thesis, I will leave a more thorough investigation of this topic for future work. Some of the domains I examined as part of this thesis, though, have other properties that make planning in the way described in this section infeasible. In the next section I will discuss a different approach that addresses the case in which a goal might not actually be reachable.
4.1.7 Unreachable Goals

A key challenge in the domains that are of particular interest for this thesis is that the goals that the agents should pursue might actually not be achievable. For example, in the One Night Ultimate Werewolf domain, one of the goals an agent should pursue would ideally be “Change the beliefs of the other players so that they believe that you are a Villager”. Because the domain features lying, though, no communicative action may be assumed to completely convince any other player of anything, because they may always opt not to believe the statement being made. However, this is where the weight assignments for worlds, described in section 4.1.4, can serve as a useful tool. Rather than assuming that other players believe what is said, the agent assumes that they will consider worlds that contradict their statements as being less likely. The weighted quality measure also provides another tool that can be used when determining success. Instead of requiring the agent to compute a plan that reaches the goal, I view the problem as one of optimization, where the agent should find a sequence of actions that maximizes the weighted quality of a goal condition.

In addition to handling the case in which agents’ actual beliefs are not changed by the epistemic actions, using the weighted quality of a goal can also serve as heuristic information. One problem with the heuristic described in the previous section is that it treats all actions that have any effect on an agent’s beliefs as basically the same. This means that at any point, there are many different actions with the same heuristic value, resulting in potentially many states that need to be visited before a goal can be reached. Unfortunately, this also implies that if the search process was stopped at any point, perhaps because the agent ran out of time allotted to it, it is unknown how close to the goal the search process got. Instead of using the heuristic for actions described above, my agents therefore use the weighted quality of the goal as the heuristic value of a particular state. The search process itself could then be performed by an implementation of A* [Har68]: The cost to reach a state is equal to the number of steps taken to reach it, and the heuristic value associated with it is equal to the 1 minus the weighted quality of the goal at that state. Note that for non-epistemic goals this means that the heuristic value is always 0, which causes the algorithm to perform a breadth-first search [Moo59].

This approach still has some issues in practice, because the heuristic and the cost function use fundamentally different units of measurement. Where one measures a cost in steps, the other measures a difference in weighted quality, which will always be between 0 and 1. Instead of using the state in which the sum of the cost and the heuristic, \(c(s) + (1 - W_g(s))\), is minimal, my search algorithm instead uses cost and heuristics as trade-offs between exploration and greedy search. For each state a value \(\frac{W_g(x)}{(1+\beta \log(1+c(s)))}\) is computed, and search continues from the state on the search frontier for which this value is maximal. Table 4.2 shows some values the heuristic will take for different weighted qualities and plan lengths, when \(\beta = 1\). Increasing \(\beta\) can force
Table 4.2 Values the epistemic planning heuristic takes for different weighted qualities and plan lengths, with $\beta = 1$

<table>
<thead>
<tr>
<th>$W_g(x)$</th>
<th>$c(s)$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.1</td>
<td>0.06</td>
<td>0.05</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.12</td>
<td>0.10</td>
<td>0.08</td>
<td>0.08</td>
<td>0.07</td>
<td>0.07</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.18</td>
<td>0.14</td>
<td>0.13</td>
<td>0.11</td>
<td>0.11</td>
<td>0.10</td>
<td>0.10</td>
<td>0.09</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.24</td>
<td>0.19</td>
<td>0.17</td>
<td>0.15</td>
<td>0.14</td>
<td>0.14</td>
<td>0.13</td>
<td>0.13</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.30</td>
<td>0.24</td>
<td>0.21</td>
<td>0.19</td>
<td>0.18</td>
<td>0.17</td>
<td>0.16</td>
<td>0.16</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.35</td>
<td>0.29</td>
<td>0.25</td>
<td>0.23</td>
<td>0.21</td>
<td>0.20</td>
<td>0.19</td>
<td>0.19</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>0.41</td>
<td>0.33</td>
<td>0.29</td>
<td>0.27</td>
<td>0.25</td>
<td>0.24</td>
<td>0.23</td>
<td>0.22</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.47</td>
<td>0.38</td>
<td>0.34</td>
<td>0.31</td>
<td>0.29</td>
<td>0.27</td>
<td>0.26</td>
<td>0.25</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>0.53</td>
<td>0.43</td>
<td>0.38</td>
<td>0.34</td>
<td>0.32</td>
<td>0.31</td>
<td>0.29</td>
<td>0.28</td>
<td>0.27</td>
<td></td>
</tr>
</tbody>
</table>

more exploration, by increasing the impact the plan length has on keeping the heuristic value low, while lower values for $\beta$ cause the search process to be more greedy. The logarithm of the plan length is taken to increase exploration for short plans without punishing longer plans overly much. For example, as shown in the table, a (partial) plan of length 1 that achieves a weighted quality of 0.5 will be expanded before a plan of length 2, even if the latter achieves a weighted quality of 0.6. However, a plan of length 5 that achieves a weighted quality of 0.5 will be expanded after plans of up to length 9 that achieve a weighted quality of 0.6.

The planning process itself keeps expanding state nodes until either the goal is actually reached, i.e. has weighted quality 1, or up to a predefined number of nodes have been expanded. At that point, the plan that reaches the state in which the goal condition has the highest weighted quality is returned. Ostari also supports an extended version of this process, in which multiple separate goals are passed to the planning process. The heuristic value described above is calculated separately for each of these goals, and the maximum among these values is used as the value of a state. When the process terminates, it will return the plan that achieves the highest weighted quality among all plan for all passed goals, i.e. while multiple goals are passed in, the process will calculate which of these goals the agent can reach most closely. In the next section I will describe how this process is used by the agents to form and reconsider intentions.

### 4.2 Intentional Agents for Doxastic Games

The main focus of my work was the development of agents that behave intentionally, and use their intentions to communicate effectively, or even deceive other agents or humans. In this
section I will just briefly outline how Cohen and Levesque’s model of intentionality [Coh90] can be used in scenarios involving communication, and practical considerations when applying it to develop actual agents. As described in section 2.2, intentional agents choose what to do according to their preferences and then commit to achieving these intentions as long as is practical. The condition of being practical is not well-defined, and I will describe in section 4.2.5 how my agents decide when to reconsider their intentions. As a reminder, in Cohen and Levesque’s work, this is done by adding a condition to the intention, i.e. their intention modality has the form \( \text{intend}_2 x \phi \psi \), where \( x \) is the agent whose intention it is, \( \phi \) is the intention and \( \psi \) is the relativization condition. This approach is well suited to express how agents’ intentions evolved over time, like in the case when there is a complete time-line of what happened and could have happened, and a system needs to reason about agents’ intentions after the fact. However, my use-case is the actual formation of intentions, and Cohen and Levesque’s formalism does not address how agents should reason about future intentions before they adopt them. What they do describe, though, are the properties of intentions as enumerated in section 2.2. These properties serve as the basis for how my intentional agents behave.

Another challenge not addressed by existing work is the communication between AI agents and humans, including communication in an environment with willful deception. My work therefore expands upon previous work by incorporating not only beliefs about the state of the world, but also beliefs about the beliefs of other agents in the world, and allowing agents to have intentions that involve such beliefs. Because agents cannot influence what other agents will actually do or accept as true, or even necessarily observe what other agents did in the past, plans to achieve such intentions, or agents’ beliefs about how the current world state came about, can contain actions that an agent only suspects will happen or have happened, without knowing so for a fact. For example, a player in One Night Ultimate Werewolf may want to convince other players that they have a particular role, but they suspect that another player is the Seer and has seen someone’s card. The player then has to take into account these suspected actions when trying to make up their story. Over the course of the game, it may also happen that other players state things that the player may or may not accept as being true, and that new information can have an influence on what the player thinks they can convince other people of. In this example, if another player does end up claiming to be the Seer and that they saw that a third player is the Rascal, the first agent may not want to claim to be the Rascal anymore.

While the results in the Hanabi study were encouraging, the goal of my work is to show that my approach is general enough to be used in multiple games and other domains. To do that, I built an agent framework based on how the Hanabi agent works, and encoded several different domains for use with this framework. One restriction of this generalized approach is the complexity of the games it can be used for, as measured by how many possible worlds agents need to consider. For example, Hanabi, with its 50 cards, would require the agents to
consider $50! > 3.04 \cdot 10^{64}$ different worlds, which is beyond the storage or processing capacity of a contemporary PC. There are, of course, several ways to address this limitation, from simply moving to more powerful hardware to improving the state encoding, but these are out of scope for this thesis. I will discuss how this limitation could be addressed in future work in section 7.2.3. Other games, such as One Night Ultimate Werewolf, have much more manageable numbers of possible worlds ($8! = 40320$ for a 5 player game of One Night Ultimate Werewolf, for example), though, and thus will be the test cases for the generalized approach. I will now describe the general outline of the agent, as well as how it can be applied to several test domains. The main application domain, One Night Ultimate Werewolf, will be discussed in more detail in chapters 5 and 6.

4.2.1 Agent Design

At the core of the agent design lies a BDI architecture adapted from the work of Wooldridge [Woo00]. While Wooldridge’s agent performs a loop where it continuously executes actions, in my case the game is played in turns, and when it is an agent’s turn it needs to determine its action for that turn. Therefore, my agent actually needs to perform two tasks: Interpret actions performed by other agents on their turns, and deliberate about which action to take on its own turn. The first task is addressed by the epistemic state of the game, which contains a model for which worlds the agent considers possible, including their weights. To determine its next move, the agent keeps track of its current intention $I$ and the plan $\pi$ to achieve it. On each of its turns, the agent performs the following steps:

1. Check if the current plan $\pi$ is not executable anymore, or has been finished.
2. Check if the current intention $I$ is still desirable, or if there is a better option.
3. If either of these is true:
   (a) Find a new intention $I$.
   (b) Find a new plan $\pi$ to achieve $I$.
4. Execute the first step of $\pi$.

Appendix A.2 shows the Haskell implementation of these steps. It comes in three variants:

- The fanatical version never performs step 2, i.e. it never reconsiders once it has adopted an intention, as long as it can execute the plan corresponding to that intention.
- The capricious version always performs steps 3a and 3b whether or not the conditions in steps 1 and 2 hold, i.e. it does not commit to its intentions over multiple turns.
The balanced version performs all steps as listed, and therefore reconsiders its intentions if it deems that desirable.

The main contribution of this part of my dissertation consists of the details of how the deliberation to reconsider an intention is performed. Before I describe how intentions and plans are represented in my agents, and what it means to form and reconsider intentions, I will briefly describe how statements made by other players are interpreted as world weights.

4.2.2 Interpreting Received Information

A challenge in doxastic games, especially those involving deception, is the interpretation of received information. Obviously, if another players says something that conflicts with the agent’s existing (certain) knowledge, this new information can be discarded. However, it might also mean that the agent should trust the speaker less in the future, or reconsider whether to believe their previous statements. From the perspective of the speaker the effect of a statement is also not entirely clear. Consider an agent saying “I am a Villager”. A listener may trust the agent, and change their belief accordingly, or they don’t and either don’t change their beliefs, or change their beliefs to the negation of the statement and no longer believe it even possible that the agent is a Villager. If they have neither a reason to believe the agent outright, nor to distrust them completely, though, the effect of the statement on their beliefs is less clearly defined.

With the definition of the weighted quality of a belief as described in section 4.1.4, the effect of a statement on a listener’s beliefs can be represented by changing weights of worlds. The basic assumption underlying this reasoning is that the majority of players, and agents in general, usually has an interest in telling the truth, and deception is the exception rather than the norm, which means worlds are less likely the more lies are told. The assumption, that the majority of agents has an interest in divulging the truth holds, is typical in social deduction games, where the faction that constitutes the majority of players has an interest in finding the factual truth. In such a scenario, instead of explicitly believing or disbelieving a speaker’s statements, an agent can compare the statement to every world they consider possible, and mark those worlds which the statement contradicts as ones in which the speaker has lied. By accumulating lies and truths in the different worlds in this manner, worlds which can only be true if many players simultaneously lied will be considered less likely than those in which fewer players lied. For example, if player $a$ says “I am a Villager”, the agent will mark every world they consider possible in which player $a$ is not a Villager as less likely. When player $b$ then says “I am the Seer”, the agent will mark worlds in which player $b$ is not the Seer as less likely as well. After these two statements, worlds in which player $a$ is a Villager, and player $b$ is the Seer are considered the most likely, worlds in which either of these two role assignments is not the case are less likely, and worlds in which neither of these two role assignments is true are considered the least likely to be the actual
Table 4.3 World weights only considering 3 players after player a claimed to be the Seer, player b claimed to be a Villager and player c also claimed to be the Seer.

<table>
<thead>
<tr>
<th>Player a</th>
<th>Player b</th>
<th>Player c</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seer</td>
<td>Villager</td>
<td>Werewolf</td>
<td>1</td>
</tr>
<tr>
<td>Seer</td>
<td>Werewolf</td>
<td>Villager</td>
<td>2</td>
</tr>
<tr>
<td>Villager</td>
<td>Seer</td>
<td>Werewolf</td>
<td>3</td>
</tr>
<tr>
<td>Seer</td>
<td>Werewolf</td>
<td>Villager</td>
<td>2</td>
</tr>
<tr>
<td>Werewolf</td>
<td>Villager</td>
<td>Seer</td>
<td>1</td>
</tr>
<tr>
<td>Werewolf</td>
<td>Seer</td>
<td>Villager</td>
<td>3</td>
</tr>
</tbody>
</table>

world. If player c now states that they are the Seer, this would contradict all worlds in which player a is assigned the Seer, and reduces how likely the agent considers them, however player a’s previous statement makes all worlds in which player c is the Seer less likely, too. Table 4.3 shows the effect of these three statements on the world weights, limited to the roles of these 3 players. Note that how likely a world is considered to be decreases with increased world weight. In this scenario, the most likely worlds would be considered the ones in which either player a is the Werewolf, or player c is the Werewolf, with player b being a Villager, just as they claimed.

To form intentions, the agent considers the weights they assign to worlds, and also what weights they believe other agents assign to the worlds. This allows the agent to tell lies that build on what other agents said, or contradict them in a targeted way. For example, a Werewolf player can determine the worlds they believe other players consider most likely, and claim roles that are consistent with most of these worlds.

4.2.3 Intentions

Syntactically, an intention is simply a valid DEL formula $\phi$. This formula may, and usually will, contain epistemic operators, such as $\Box$ or even $Q$ to indicate that another agent’s beliefs, or the quality of their belief, are the goal to be achieved. An agent can hold an intention $\phi$, i.e. they intend for $\phi$ to be true, or as true as possible. To hold an intention, the agent needs to have a plan to achieve it. A plan $\pi$ in this case is simply a totally ordered list $\pi = (a_i)_i$ of actions that can be applied to the current epistemic state $S_0$. Because intentions in my system typically consist of goals that can not be reached, there is no formal requirement for the result of the plan. In theory this means that an agent could hold an intention $\phi$, and have a plan $(a_i)_i$ that would result in the weighted quality of $\phi$ being 0. In practice, this is averted by how the agent forms

\footnote{The restriction to totally ordered plans is only for notational convenience and simplicity of the argument for the remainder of this document. If a plan is partially ordered, any total ordering of the steps can be chosen at any point a plan is required.}
and reconsiders its intentions, which is what I will describe next.

4.2.4 Forming Intentions

To form an intention, an agent needs to have some metric of what is desirable and then choose from these desirable states. In the case of games, desirable states usually involve winning or scoring well, but these are not the only things to optimize for. For example, dynamic difficulty adjustment is often used to make games more enjoyable for players by reducing the difficulty when players perform poorly, or increase it when they do well. The exact optimization criterion, and difficulty adjustment, are outside the scope of my work, but my system provides a mechanism for an expert to provide desirable states that are then used by the agent to form intentions. For example, for One Night Ultimate Werewolf it is generally the goal of a Werewolf player to deceive the other players into thinking that they are not a Werewolf. Possible goals for an agent in such a situation can include that they convince another player that they are a Villager or the Seer or any other role. The agent, when presented with the choice between these desirable states, will then have to select which one they want to pursue. To do so, they determine which scenario they can convincingly convey to the other players. For example, if another player already claims to be the Seer and is believed by other players, it will be harder for the agent to convince the other players that they are the Seer. In my system, these potential goals are provided by a domain expert, but they can be numerous and on a very high level. For the One Night Ultimate Werewolf domain, for example, the goals for the Werewolf player contain “Convince the other players that you are a Villager”, “Convince the other players that you are the Seer”, etc. The agent will then determine which of these goals to adopt based on how well it can reach them.

To actually form an intention, the agent considers all of the candidate goals provided by the domain expert, and uses the epistemic planning process described in section 4.1.7 to determine which of the goals it can reach most closely, i.e. with the highest weighted quality. It then adopts that goal as its intention, with the plan produced by the planning process, and also stores the expected quality of the goal. This expected quality serves as the basis for the determination of when to reconsider an intention, as described in the next section.

4.2.5 Reconsidering Intentions

The key contribution of my work is how agents reconsider their intentions. In prior work, one of the limitations was that planning to achieve a new intention is computationally expensive and therefore should not be done unnecessarily. In my system, the planning process can be as short, but potentially less accurate, as needed, and therefore this exact same limitation does not apply. Instead, the planning process in my system provides a measure for how well a goal can be reached, and will return the goal that can be reached most closely among all available
goals. An agent may opt to perform this process every single turn, and always adopt whatever new goal is returned as being the best by this process. I call this approach the *capricious* agent. However, the hypothesis set forth by this thesis is that human players expect and prefer agents to be committed to their goals to some extent. The extreme case of this behavior is what I call the *fanatical* agent, which only performs a new planning operation when it finished executing its existing plan, or determines it can no longer execute that plan. The third option is to reconsider intentions "sometimes", where the exact definition of sometimes determines how committed an agent is to a goal.

Consider an agent that has an intention $\phi$, that they can reach to weighted quality $w$ with a plan $(a_i)_i$. On their next turn, they compute a new intention $\psi$, with a plan $(b_i)_i$, that they can reach to a weighted quality $w'$. A *balanced* agent will adopt the new intention $\psi$ if and only if:

$$w' \geq \alpha \cdot w$$

The value of $\alpha$ controls how committed the agent should be to its goal. When $\alpha = 0$, the inequality always holds, and the agent will always adopt the new best intention, which corresponds to the *capricious* agent. On the other hand, when $\alpha = \infty$, the agent will never adopt the new intention, leading to a *fanatical* agent. It might seem that values of $\alpha \leq 1$ are not different from the capricious case, but note that the quality of the old plan is *not* newly calculated. Instead, the comparison is between how good the existing plan was estimated to be previously, versus how good the new plan is estimated to be in the current state. Also note that this procedure does not actually distinguish between goals and plans to achieve them. This means, that a new plan to adopt the intention that is already held is only adopted if its quality fulfills the inequality, i.e. it is an improvement of factor $\alpha$ over the existing plan.

Determining which value of $\alpha$ to actually choose depends on any given domain, and the circumstances in which the agents are to be used, and is another major part of this thesis. Chapter 5 describes the effect of different values of $\alpha$ on agents playing One Night Ultimate Werewolf against other agents, while chapter 6 contains how human players perceive different agent types. Before I describe the One Night Ultimate Werewolf in detail, though, I will briefly outline some other application domains for Ostari.

---

6For many domains that consists exclusively of communicative actions this is never actually the case, since agents can always say whatever they want, and therefore no action becomes unexecutable.

7Note that this is actually desirable, because actions performed in one plan may have *unintended side-effects*, and committing to a *plan* rather than a *goal* accounts for these side-effects as they may be observed by other agents.
4.3 Application Domains

To demonstrate that my agent design is not limited to One Night Ultimate Werewolf, or even just social deduction games in general, I will briefly discuss some other domains. While none of them were treated in as much detail as One Night Ultimate Werewolf, they present starting points for more in-depth future work, which I will describe in chapter 7. The domains describe in this section contain logical puzzles that involve reasoning about character beliefs, and story generation for narratives involving beliefs and suspicion. I will also briefly discuss how Ostari enabled an undergraduate student to build a detective game as a summer project, as an example for how Ostari eases authorial burden for epistemic actions.

4.3.1 A Logic Joke

The first domain I present is a class of logical puzzles, best represented by the following joke, which is part of mathematical folk lore:

Three logicians walk into a bar. The bar keeper asks them: “Do all of you want a beer?”

The first logician answers: “I don’t know”.

The second logician answers: “I don’t know”,

The third logician answers: “Yes”.

The idea behind this joke is that every logician only knows if they themselves want a beer, but not about the other logicians’ thirst, and that the negation of “all logicians want a beer” is “there is a logician that does not want a beer”. Every logician therefore has three possible answers: “Yes”, if they know that all logicians want a beer, “No”, if they know that there is at least one logician that does not want a beer, and “I don’t know” if neither of the two previous cases applies. In the instance presented in the joke all three logicians indeed want a beer. When the first logician answers, they only know that they themselves want a beer, but they do not know about the other two logicians, and so they will answer “I don’t know”. At this point reasoning about other agents’ beliefs becomes important for the other two logicians: The second logician does not know whether the other two logicians want a beer, but they know that the other logicians know that fact about themselves. The second logician can therefore reason that if the first logician did not want a beer, they would know that they did not want a beer, and would therefore have known that at least one logician does not want a beer and answered “No”. Because they did not answer with “No”, the second logician can conclude that the first logician indeed wants a beer. Additionally, they want a beer themselves, but do not know about the third logician, so they will answer with “I don’t know” as well. The third logician then performs the same reasoning.
about the first logicians, and concludes that both of them want a beer, and they know that they
themselves want a beer, and therefore can answer with “Yes”.

In Dynamic Epistemic Logic the situation in this joke is represented by 8 possible worlds,
one for each possible assignment of “wants a beer” and “does not want a beer” to the three
logicians/agents, with the actual world consisting of the one in which “wants a beer” is assigned
to all three logicians. Let the three logician be named \(a\), \(b\), and \(c\), and \(\text{wants}(x, \text{Beer})\) mean that
logician \(x\) wants a beer. Figure 4.2 shows the possible worlds and how they appear to each agent,
where \(\text{wants}(x, \text{Beer})\) is abbreviated with \(x\) in each world. Note that in each world each agent
considers three other worlds also possible, which together are exactly the four worlds in which
their desire for beer, or lack thereof, matches. For example, the red edges connect all four worlds
in which \(a\) wants a beer, and the blue edges connect all four worlds in which \(a\) does not want a
beer. This means, that in any world in which a logician wants a beer, they will also believe that
they want a beer, but also that each logician believes that in each world they consider possible
that the other logicians know whether they want a beer or not.

To perform the necessary reasoning in Ostari, the action shown in listing 4.5 is used. It
represents one logician answering the question of the bar keeper. If the logician believes that all
logicians want a beer, they will answer with “Yes”. Otherwise, if they believe that there is at
least one logician that does not want a beer, they will answer with “No”, otherwise they answer
with “I don’t know”. The only thing of note in this action is that the if statements are marked
as public to the logicians, which means the other logicians will learn which branch was taken,
and therefore whether or not the condition holds. When logician \(a\) performs this action, they
neither believe that all logicians want a beer, nor that there exists a logician that does not want
a beer. Because this second if statement is public, logicians \(b\) and \(c\) can eliminate all worlds they
consider possible in which logician \(a\) would believe that there exists a logician that does not want
a beer. In particular, these are all worlds in which logician \(a\) does not want a beer, leaving only
the worlds which are connected by red edges in figure 4.2. In Ostari this is done by executing
the public action in all possible worlds for the agents to which it is public, and only keeping the
ones in which the same branch is taken as in the actual world.

After this action, logician \(b\) performs the same action, and, like logician \(a\), they neither believe
that all logicians want a beer, nor that there is a logician that does not want a beer, so they
end up saying “I don’t know”. Logician \(c\) then eliminates all worlds in which either of these two
beliefs would hold for logician \(b\), leaving only the world in which \(\text{wants}(a, \text{Beer}), \text{wants}(b, \text{Beer})\),
and \(\text{wants}(c, \text{Beer})\) hold. When logician \(c\) finally performs the same action, they will believe that
all logicians want a beer, and answer with “Yes”.

Similar to this joke, there are several logic puzzles that can be encoded in Ostari analogously.
The muddy children puzzle, which Baltag [Bal02] uses as a demonstration for his variant of
Dynamic Epistemic Logic, consists of \(n\) children, a subset of which has mud on their faces. In
Figure 4.2 A representation of the initial beliefs of the logicians in the joke. Each node represents a world in which the logicians listed in the label of the node want a beer. Note that each world is additionally also considered to connected to itself for all three logicians in this example.
contrast to the logician joke, the children do not know if their own face has mud on it, but they can see all other children’s faces. In this scenario, an outside actor, canonically said to be one of the children’s parents, states that at least one of the children’s faces has mud on it, and then asks which of the children knows whether their face has mud on it or not. Similar to the logicians joke, the children have to reason about what the other children would do for each world they consider possible, which leads to them deducing whether they have mud on their face or not. Another formulation of this kind of puzzles are hat-guessing games, in which a group of agents wear hats of different colors, and can only see a subset of other hats [But09]. These puzzles constitute basic demonstrations of the capabilities of Dynamic Epistemic Logic and Ostari, but in the next section I will describe an application that is more involved, the generation of stories that involve beliefs.

4.3.2 Narrative Generation

While the focus of my work is on agent design for doxastic games, I also want to show that my work has other applications. As described in section 4.1.6, Ostari provides the means to find an action sequence such that a goal condition is satisfied, similar to planning. A domain planning has been successfully applied to is that of the computational generation of stories [Deh81]. For this task, the initial state is viewed as the setting for the story, and the goal is provided by an author. The planner will then find a sequence of actions that the characters in the story perform.
to achieve the author’s goal. It has been noted that the standard formulation of this problem often results in stories that lack in terms of believability, because the characters act towards the author’s goal instead of their own, i.e. they lack intentionality. Riedl and Young describe a solution to this problem, in which each character only performs actions if they serve one of their own goals [Rie10]. Ware and Young further expanded upon this work by allowing conflict to arise which can result in characters’ intentions failing [War11]. As the basis for my work is Cohen and Levesque’s work on intentionality in which they argue that there is a tight link between intentions and beliefs [Coh90], I would argue that story planning could also benefit from a representation of character beliefs. Indeed, Shirvani et al. have argued similarly and describe their story planning system that supports character beliefs in addition to intentions [Shi17b]. However, their focus is on producing a fast story planning system, which limits its generality.

Ostari, on the other hand, exposes the full expressive range of Dynamic Epistemic Logic and can, with a proper domain, be used to generate short stories. As an example, consider a whodunit detective story in which the detective and the murderer necessarily have different beliefs about the state of the world. In particular, the murderer will know who killed the victim, while the whole point of the story is for the detective to collect clues and deduce the murderer’s identity. Part of the murderer’s agenda is it to prevent the detective from doing so. They could, for example, try to incriminate someone else. More concretely, consider the actions in listing 4.6 that can be used for the first part of a simple detective story. The take, put and kill actions all have secret actors, i.e. only the person performing the action will know who performed it. The property gunowner describes who owns a gun, which - for simplicity - also means that they could have it in their possession.

Ostari can be used to generate a story using these actions by providing an initial state and an execution trace, as shown in Listing 4.7. Initially, the only relevant information is that Moriarty owns the Manor and Watson owns the Shed. However, in contrast to a classical planning system, it is also possible to explicitly execute some actions first, and then plan towards a goal. This goal would not even have to constitute the end of the story, as further actions could be executed after, as well as further goal directives. In a way, the goal: directives serve as landmarks, as they are sometimes used in narrative planning [Por09]. In this case, first the action **take(Moriarty,pistol)** is executed, and then the system is directed to satisfy a goal of ⊨ (Moriarty): ⊨ (Sherlock): murderer(Victim) == Watson, which means that Moriarty believes that Sherlock believes that the murderer of the victim is Watson, i.e. Moriarty is trying to incriminate Watson. The action sequence that the system finds in this case is **kill(Moriarty, Victim, pistol)**, **put(Moriarty, pistol, Shed)** and **Moriarty suspects find(Sherlock, pistol, Shed)**, which corresponds to Moriarty killing the victim, planting the pistol in the Shed, where Moriarty suspects that Sherlock will find it. Because finding the gun will cause Sherlock to believe that it belongs to Watson, and because the murder required the gun, this
take(m(m): Suspects, g: Guns)
{
    gunowner(g) = m
}
put(m(m): Suspects, g: Guns, l: Locations)
{
    precondition gunowner(g) == m;
    gunowner(g) = Null;
    at(g) = l
}
kill(m(m): Suspects, v: Victims, g: Guns)
{
    precondition gunowner(g) == m;
    murderer(v) = m
}
find(d: Detectives, g: Guns, l: Locations)
{
    precondition at(g) == l;
    suspect (d): gunowner(g) == owner(l)
}

Listing 4.6 A simple detective story domain

would cause Sherlock to believe that Watson is guilty. However, note that the action of Sherlock finding the gun did not actually happen, it was only suspected by Moriarty.

This domain demonstrates Ostari’s planning capabilities applied to a domain that traditionally requires disparate knowledge between different agents. However, since story generation was not the focus of my thesis, the stories produced by this domain are very simple. The results were promising enough, though, to build a detective game, in which the player has to uncover a murderer, on the basis of this work, which I will describe in the next section.

4.3.3 Detective Game

To demonstrate how Ostari can be used in a game where keeping track of a player’s beliefs, and updating them according to what they encounter in the game world, is essential, I came up with the idea of a detective game, as an extension of the detective story generation described above. The actual implementation of this game was performed by an undergraduate student as a summer project [Moh18], where I served as a mentor. The game consists of a python module that generates characters with several different properties and outputs an Ostari script that contains actions for the characters to commit a murder, and for the detective to gather clues. These clues consist of physical evidence left behind by the actions performed by the murderer, as well as actions to question the suspects. I built a simple web server that could communicate with Ostari, based on the architecture used for the study described in chapter 6, and the student worked on
making the presentation more usable. The result is a detective game that can be played in a web browser, where the player has a limited number of actions available to identify the culprit before the murderer gets away and they lose. Ostari is used to keep track of what the detective learns during game play and whether they have gathered enough evidence to have a case against the criminal by eliminating all other suspects. While the game was primarily implemented by the student, this constitutes another demonstration of the applicability of Ostari to other domains, and how it facilitates the development of applications which rely on belief manipulation.

### 4.3.4 Social Deduction Games

Before I describe the main application dissected in this document, the game One Night Ultimate Werewolf, I want to briefly discuss the broader class of social deduction games. In these games, players are typically assigned to one of two factions, where one faction has an information advantage, and the other faction a numerical advantage. Examples include One Night Ultimate Werewolf and its ancestors, Werewolf and Mafia, The Resistance, but also games that include more traditional board game elements such as Battlestar Galactica: The Board Game. Most of these games involve voting, with a majority or plurality vote driving the game to a conclusion. The smaller faction therefore has an incentive to lie, because if they told the truth about their own faction affiliation, the members of the other, larger faction would not vote in their favor. Likewise, players on the larger faction are more likely to tell the truth in order not to be caught in contradictions and be suspected of being in the smaller faction. This insight is why Ostari’s weighted quality can be used for game play decisions by the agents. The next two chapters contain a more detailed description of how the game One Night Ultimate Werewolf was implemented in Ostari and the experiments I performed, but I will describe how my work could be extended to a
few select other games in chapter 7.

As the name social deduction games implies, players do not just make deductive inferences, there is also social reasoning involved. For the purpose of this thesis I will focus on the logical properties of the game, and how to convey information, and deceive players in a logically consistent manner, but there are also other important aspects of the game. For example, to determine if a player is lying, players often use “tells” or “cues”, whether they are real or imagined, such as facial expression, vocal features, or hesitation. The way such features could be integrated into my agents in future work will also be briefly discussed in chapter 7.
The main application presented in this document is the domain of One Night Ultimate Werewolf. However, because the focus of my work is not the real time analysis of arbitrary statements in natural language, I will be using a modified version of the game, in which the statements players can make are predefined. Of course, the range of statements players are able to make should allow them to describe the game state as they believe it to be, in order to be able to convince the other players of their version of reality. Therefore, there are fixed-syntax statements available that cover all situations in which players state which role they have or don’t have (e.g. “I’m not a Werewolf!”) and what they did as that role (e.g. “I am the Troublemaker and swapped Lisa’s and Günther’s cards”), as well as accusations they have for other players (e.g. “I have seen Xena’s card. She is a Werewolf”). Additionally, players will take actions in a turn-based fashion to avoid the necessity for synchronization. However, players may skip any turn, so they are not forced to reveal any information if they do not want to (except for the fact that they don’t want to perform an action). The time limit present in the original game translates into a turn-limit in this version.

In this chapter I will describe the details of how the game is encoded in this turn based fashion in Ostari, the actions that are available to the players during the discussion phase, and how the night phase and voting phase are encoded. I will then describe how my agents can play
this game and present some typical scenarios and actions taken by the agents. Finally, I will discuss an experiment I performed in which I had different agent types play against each other, and the effect agent commitment has on their performance in the game.

5.1 One Night Ultimate Werewolf in Ostari

The game One Night Ultimate Werewolf consists of several parts that need to be encoded in Ostari, each with its own unique requirements. For computational resource reasons, I will focus on games with 5 players, and the roles Villager, Werewolf, Seer, Rascal, Robber, and Insomniac, but the same approach can be used to extend the game to more players and roles with more powerful hardware or further optimizations described in chapter 7. In this section I will first describe how the game is represented in Ostari, and then describe the encoding of the actions for the night phase, the discussion phase and the voting phase. I will also discuss how the game in this encoding differs from the game as it is typically played by human players. In the following section I will then describe how agents actually play the game based on this encoding.

5.1.1 Game State Representation

At the beginning of the game, players are secretly assigned different roles. In the actual game this is done by dealing cards that have the roles depicted on them to the players, where some roles are depicted on multiple cards. Since objects in Ostari are unique, this is encoded with card objects which are what is dealt to each player, and followed by a role assignment phase, in which players are assigned roles depending on the cards they were dealt. For example, the cards Werewolf1 and Werewolf2 would both result in the player being assigned the role Werewolf. In Ostari, these operations are performed as two separate actions: Dealing the card is done by an action assignCard, shown in listing 5.1, which takes a player and a card as parameters, where the card is secret, and only known to that player. This means that every other player will see the action as an option between any possible card being dealt to that player. The precondition in the action that ensures that no other player was already dealt that card actually serves two purposes. Apart from preventing any actual card from being dealt twice it also makes sure that any world any other player considers possible does not contain duplicate cards. For example, in the 5 player game with 8 cards, if player a is dealt the card Seer, every other player will create a new possible world for each possible card, resulting in 8 worlds, including one in which player a actually has the card Seer. When player b is then dealt the card Villager1, each other player will again create 8 possible worlds for each world they already consider possible, one for each possible value of the card parameter, and try to apply the action to it. However, for the assignment of Seer to the card parameter, and each world in which player a was already
assignCard(p: Players, c(p): Cards)
{
    precondition Forall x in Players: (card(x) != c);
    card(p) = c
}

Listing 5.1 Ostari action representing dealing a card.

assignRole(p: Players)
{
    public (Players) role(p) = roleassignment(card(p));
    public (Players) initrole(p) = roleassignment(card(p));
    learn (p): Which r in Roles: role(p) == r
}

Listing 5.2 Ostari action representing assigning roles from the card that was dealt.

assigned the Seer card, the precondition will not hold, and the action will fail to apply. The result is that there will only be 7 new worlds created for each world the agents consider possible. The same will happen for the next player, with 2 worlds being eliminated for each possibility, resulting in 6 new possible worlds, and so on, which would lead to $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4$ worlds. However, each player also knows which card they got themselves, and the action will not apply in worlds in which other players already have that card, eliminating them, which means every agent will consider $7 \cdot 6 \cdot 5 \cdot 4 = 840$ worlds possible after the cards are dealt. Note that the remaining three cards, which are dealt in the center, are identifiable by each agent in each of these worlds, but their order is not.

For the roles assigned to the players, it is actually important to distinguish between two types of assignments: The roles players initially have, which is what determines which actions they can perform during the night, and the roles players actually have. For this, there are two properties, initrole and role, where the former is used in preconditions of actions in the night phase, as described below, and the latter is used to determine the winning team. The action shown in listing 5.2 also contains statements that make the player learn which role they have. Note that the learn statement works by eliminating all worlds in which the given condition does not reflect the actual world. Since the role and initrole properties have the same value assigned to them in all possible worlds, it is unnecessary for players to also learn their initrole.

To initialize the game, these two actions are executed for each player in the game, after which each player considers the 840 worlds possible that correspond to each assignment of roles to other players, and in each of these worlds each player considers 840 worlds possible, and so on. These worlds contain the information which cards each player was dealt, which role they started with and which role they now have (which are identical). Note that the three center cards are
identifiable by each agent but they do not reason about their ordering. This means that agents can refer to cards in the center, which are denoted with center1, center2, and center3, but they do not actually know which is which, even when they see them. While this constitutes a difference from the actual game, this information is rarely useful with the roles included in the game, because none of them can interact with cards with the center, and reasoning about the order would increase the number of possible worlds by a factor of 3! = 6.

After the initial state is set up, game play can begin in earnest, using the play: directive available in Ostari. The game is defined as consisting of multiple phases, with one subphase for each role during the night phase, followed by the discussion phase, a voting phase, and finally a tallying phase during which the winning team is determined. In the next section I will give an overview of how the actions in the night phase work, followed by the discussion phase, which is the main focus of the game, and finally briefly describe how voting and tallying the votes is implemented.

5.1.2 Night Phase

During the night phase, certain roles perform their special action in a predefined order. For the roles present in my game these are, in order:

- The Werewolves learn who the other Werewolves are, if any.
- The Seer can look at any other player’s role card.
- The Rascal may exchange their two neighbors’ cards, without looking at them.
- The Robber may exchange their own card for any other player’s card, and look at their new card.
- The Insomniac looks at their own card.

While these actions, as written, always refer to players’ cards, the relevant information actually consists of their roles, as different cards with the same role are indistinguishable. In Ostari, the actions of each of these roles happens during that role’s game phase, where each game phase only permits the players to use the action in question. Since game play proceeds in turns, on each players turn they may use the action if they satisfy its preconditions, and therefore each action needs a precondition to ensure that the player started with the corresponding role. For example, listing 5.3 shows the action the Robber can perform during the night. The action has two secret parameters that are only known to the player performing the action, corresponding.

\[\text{The distinction between a player’s starting role and their current role is particularly important for roles that are exchanged. For example, if the Robber exchanges their own card for the Insomniac’s, they will not perform the Insomniac’s action during the night phase.} \]
rob(p(p): Players, p1(p): Players)
{
    precondition initrole(p) == Robber;
    if (eq(p) != p1)
    {
        role(tmp) = role(p);
        role(p) = role(p1);
        role(p1) = role(tmp);
        learn(p): Which r in Roles: role(p) == r
        robbed(p) = p1;
    } else
    {
        robbed(p) = p
    }
}

Listing 5.3 The encoding of the Robber’s action during the night phase.

to the fact that no one else knows who the Robber is and who they steal from. As mentioned, the precondition ensures that only the player that started as the Robber performs the action, but it also ensures that each variant of the action that the other players consider possible is only applied if the variant is applied to a world in which the player performing is started as the Robber. In other words, player \(a\) considers it possible that the Robber is (among other options) players \(b\), but they will only apply a \( \text{rob} \) action performed by player \(b\) in worlds in which they assume that player \(b\) started as the Robber. Since each player knows which role they started with, there are no worlds in which they are the Robber themselves if they did not actually start with the role, and they will not imagine to have performed the \( \text{rob} \) action in that case.

In addition to performing the actual action of exchanging another player’s card for their own and learning the new role, the \( \text{rob} \) action also stores who the player robbed from. Notably, this is also applied to each world the other players consider possible for each combination of robber and victim. The actions for the other roles are implemented analogously, with a detailed listing available in appendix A.1, with each action setting a property corresponding to the action that was performed. During the discussion phase, when a player claims to have performed some action during the night, such as having robbed from someone else, the players can then compare this claim to each possible world, and identify in which worlds the statement was the truth and in which it was a lie. I will describe the details of how this discussion is implemented in the next section.
5.1.3 Discussion Phase

The main deviation of my implementation from the actual game is how the discussion phase works. Because the focus of my work was not real-time processing and understanding of natural speech, or even timing of statements, but rather the underlying communicative logic of the game, the discussion phase is implemented in a very structured way. Rather than allowing agents to say things at arbitrary times, they take turns and can make one statement on each of their turns. The time limit from the original game is replaced with a limit on the number of turns each players gets. Additionally, players can not make arbitrary statements, but are instead limited to the following statement types:

- Claiming to have started with a particular role.
- Claiming to currently having a particular role.
- Claiming to have performed a particular night phase action and with which targets.
- Claiming which role they have seen that a player has.

These statements were chosen such that players could, in principle, retell everything they did in the game, or lie about having done anything they could have done. For example, player $a$ can state that they started as the Robber, performed the rob action with player $b$ as the target, and that they are now a Villager. Player $b$, meanwhile, could state that they actually started as the Seer, saw player $a$’s card, and that that card was a Werewolf card.

To actually make the statements in the game, the actions are encoded as Ostari actions, in a way that enables agents to reason about likely possible worlds using the weighted quality measure. The goal is for each action to increase the weight of each world in which the claim is contrary to the facts in that world, thus making such worlds be considered less likely by the agent. Listing 5.4 shows the action of an agent claiming to have started with a particular role. The action has three parameters, where the first one is the player performing the action, the second one corresponds to the role they claim to have started with and the third one is the role they actually started with. However, since this third parameter is secret, and only known to the player performing the action, each other player will consider the action to be one of several options, one for each possible role. Because the action also has a precondition to ensure that the acting player started with that role, for each world any other player considers possible, there is only one of these options that can be applied to that world, corresponding to whichever role they consider the acting player to have started with in that world. The effect of this action is that each player will apply exactly one version of this action to each world they consider possible, and add weight to that world if the role they consider the player to have started with in that world is different from the one the acting player claims. Statements about actions the players
claimInitRole(p: Players, r: Roles, r1(p): Roles)
{
    precondition initrole(p) == r1;
    if (initrole(p) != r)
    {
        addWeight(1)
    }
    else
    {
    }
}

Listing 5.4 The claimInitRole action from One Night Ultimate Werewolf encoded in Ostari

claimRobbed(p: Players, p1: Players, t(p): Players)
{
    precondition robbed(p) == t;
    if (robbed(p) == p1)
    {
        addWeight(1)
    }
    else
    {
    }
}

Listing 5.5 The claimRobbed action from One Night Ultimate Werewolf encoded in Ostari

performed, or which roles they saw, work analogously, as shown in listing 5.5 for the action of claiming to have robbed from another player.

The nominal goal of the actions in the discussion phase is for the players to be able to convince the other players of a certain scenario being true. Their implementation, as presented in this chapter, achieves this by marking worlds that would conflict with the information being conveyed as less likely, with the idea that, on average, players are more likely to tell the truth than to lie. However, the way the players' beliefs are actually tested are by asking them to vote for a player to eliminate, which is what I will describe next.

5.1.4 Voting Phase

After the players had a chance to convince other players of their story, the game proceeds to the voting phase. Just as in the original game, each player gets one vote. While the original game has the votes being cast simultaneously, the Ostari version works in turn order, but since the votes are not public this has no effect on the players' votes. For my implementation of One Night
vote0(p: Players, v: Players)
{
    precondition Forall p1 in Players: (eq(p1) == p) or (W [<0.9] (p): role(p1) == Werewolf);
    precondition W [>0.8] (p): role(v) == Werewolf;
    precondition eq(p) != v;
    tmpv(game) = votes(v);
    votes(v) = succ(tmpv(game))
}

Listing 5.6 One of the vote actions in Ostari, corresponding to an interval of the weighted quality for the target to be a Werewolf between 0.8 and 0.9

Ultimate Werewolf, the players do not actually have a real choice of who to vote for, they will always vote for the player other than themselves that they believe most likely to be a Werewolf, as measured by the weighted quality. Since Dynamic Epistemic Logic, and Ostari, do not have a maximum operator, or even a way to directly compare the weighted quality of two conditions at this point, the implementation of this vote was done by subdividing the possible range of the weighted quality into discrete intervals, and having one vote action for each interval. I will call this weighted quality of the belief that someone is a Werewolf the agents confidence in their vote. Listing 5.6 shows one of the resulting vote actions, corresponding to a confidence between 0.8 and 0.9. The action can only be used if the voting player believes that other players are Werewolves with a weighted quality of 0.9 or less for all players, and believes that the targeted player is a Werewolf with a weighted quality of 0.8 or more. If there were multiple players for which this is true, the voting player can choose any of them.

The advantage of this implementation of voting is that it is reasonable from any single agent’s perspective to vote for the (other) player most likely to be the Werewolf. If a player is the Werewolf themselves, this vote accounts for the possibility of having been swapped with someone else, and if a player is not a Werewolf the goal for their faction is to vote for the Werewolf anyway. However, the drawback is that this voting mechanism does not account for other players. For the Werewolf players that means that they may end up voting for each other, if one of them became suspicious. For the remainder of my work I addressed this by limiting the game to only having one player actually be assigned the Werewolf. I will address this limitation in chapter 7. For the Citizens there are cases in which players might want to split the vote between the top two suspicious players in order to kill both of them. However, in practice the Werewolf player may always foil such plans, so that they are rarely successful anyway. The main advantage of this simple voting mechanism is that it establishes a baseline that can be used to compare how different agents perform during the discussion phase, because it largely eliminates their decisions from the voting phase. In the next section I will describe how my agent framework was applied
to this game, and what typical and anomalous games look like.

5.2 Intentional Agents for One Night Ultimate Werewolf

Using the encoding of the game in Ostari described above, it is already possible to have agents play the game. Built into Ostari is an example agent that simply chooses actions randomly. However, the goal of this thesis was the investigation of how to utilize intentionality in the game. In order to be able to use the agent design described in section 4.2 to play One Night Ultimate Werewolf the agents need to be provided with goals. The agents will then form plans to achieve these goals, compare the weighted quality of the expected outcome, and decide how long to stay committed to their plans. In this section I will describe how the agents actually decide what to do, which goals they are given to achieve, how they plan to achieve them, and which plans they typically come up with.

5.2.1 Agent Goals

As described in section 4.2.4 for agents to form an intention, they use a list of candidate goals, and choose the one they believe that can achieve the best, as measured by the weighted quality. However, in an actual game not all goals are always applicable. To achieve this, the goals consist of an applicability constraint in addition to the actual goal condition. Each goal in the goal: section of an Ostari input file consists of three parts: Bindings, the applicability constraint, and the goal condition itself. The bindings are a way to make the goals easier to author, but also to be able to refer to property values outside of other agent’s beliefs. For example, a binding of \( r = \text{initrole}(\text{self}) \) can be used in a goal definition with the goal condition \( w \ [>0.7] (a): \ \text{initrole}(\text{self}) \ == \ r \). If the value of \( r \) was not bound outside the weighted quality it would always refer to whichever value agent \( a \) considers it to have in any particular world. The applicability constraint is used to determine which goals agents should even pursue, which also helps to include or exclude goals based on which phase the game is in. For example, the goal of convincing other players of having started with a particular role is only useful during the discussion phase, and thus will have an applicability constraint of \( \text{phase}(\text{game}) \ == \ \text{DiscussionPhase} \). In the rest of this section I will discuss the goals the agents are provided with in more detail, starting with all goals that are applicable during the night phase, when the roles perform their special actions, and followed by the goals for the discussion phase.

5.2.1.1 Night Phase

When performing their roles’ actions during the night phase, the players do not actually have any information other than what their own role is. Therefore, any decision they make about
what to do is arbitrary. In real life games, players often exchange cards of two particular people based on their personalities, or physical proximity if it is hard to reach across the table, but for the simulated games I am describing here such information does not exist. To achieve some variability and plausible deniability for the agents, they are therefore given goals with random targets. For example, one such goal is \( \text{robbed}(\text{self}) = x \), where \( x \) is bound with \( x = \text{random}(\text{Players}) \), and the goal has an applicability constraint of \( (\text{initrole}(\text{self}) = \text{Robber}) \) and \( (\text{phase}(\text{game}) = \text{RPhase}) \) to ensure that it is only used by the Robber and only during the Robber subphase.

By defining the goal this way, the agents can use the same mechanism for performing their actions during the night as they use for the discussion phase. When the goal is applicable, \( x \) will be bound to a random element of the set \( \text{Players} \), and the agent will try to find a plan to achieve the goal with this binding. Solving this planning problem is trivial, because it requires at most one action to achieve the goal. To cover the case in which the player randomly chooses themselves as the target, the \textit{robbed} property is initialized to the player themself for every player. The actions for the other roles that can choose what to do, the Seer and the Rascal, work analogously. The Werewolf is always told about the other Werewolves, and the Insomniac is always told about their own role, since neither of them have any choice and their respective actions are mandatory, they do not need any goals.

5.2.1.2 Discussion Phase

In contrast to the Night Phase, players have access to information during the Discussion Phase that can help them decide what to do. Some of this information was gathered during the Night Phase, but the vast majority of it comes from the other players’ statements. These statements update the agents’ mental models as described above. To determine which goals are applicable, the agents use not only factual information about the world, but also what they believe to be likely. For example, a Werewolf player has an interest in directing suspicion away from themself, but players can, generally speaking, not be certain that they are in fact Werewolves, since their cards might have been switched. To address any eventual switching of cards, the goals are thus defined as follows:

- If an agent is \textit{reasonably certain} that they are a Werewolf, they have a goal for each non-Werewolf role to convince the other players that they have that role.

- If an agent is \textit{reasonably certain} that they are not a Werewolf and that they know that someone else is a Werewolf, they have a goal to convince the other players of the Werewolf they know about.

- If an agent is \textit{reasonably certain} that they are not a Werewolf, they have goals to convince
the other players of which role they started with, and which role they think they have now.

The idea behind these goals is to provide the agents with guidance for what they should strive to achieve in the game, without dictating their behavior. Consider a player that is reasonably certain that they are a Werewolf. The only information contained in these goals is that the agent should convince the other players that they are not a Werewolf, but not how they do so, or which particular role they should convince them that they are. It is up to the agent to decide which of the different roles they will pretend to be, and which statements to use to make that plausible to the other players.

Of particular note is the use of the phrase *reasonably certain* in the goals. Since players never know for sure which role they have, they will use the weighted quality of their belief that they are or are not a Werewolf. The threshold value for how high this weighted quality should be was part of my experiments, described below. Listing 5.7 shows how these goals are encoded in Ostari. Each goal starts with the bindings, where a binding of *any* expands to create one goal for each element of the given set, and the set GoodRole contains all non-Werewolf roles, followed by the keyword *in*, and the applicability constraint, a colon and finally the actual goal. The token $\text{cert}$ is replaced with different values to determine how certain agents should be of their role before attempting to follow a particular goal. In the next section I will briefly describe how the agents use these goals to come up with their plans.

### 5.2.2 Planning

Recall the agent design from section 4.2.1 that describes what each agent does on their turn: First they determine if they want to keep their current plan (if any), and if not, they then come up with new plans for their potential goals and choose the plan that reaches the respective goal most closely. As mentioned above, during the night phase any plan the agents come up with consist of at most one step, corresponding to one of the options they have for their role’s special action. The interesting case is therefore the discussion phase. Player $a$ starts the game, and determines which of the goals are applicable. At the start of the game, most of their information is derived from what they learned during the night, and what they suspect could have happened during the night. A Werewolf player, for example, believes that they are still the Werewolf, unless one of their neighbors is the Rascal and swapped card, or if the Robber stole from them. In terms of possible worlds for a Werewolf player this means:

- The agent considers $7 \cdot 6 \cdot 5 \cdot 4 = 840$ worlds possible after the cards were dealt (ignoring the order of the center cards as described above).

- After the Rascal performed their action, the agent creates two new worlds for each world in which there is a Rascal in the game, one in which they swapped their neighbors’ cards
\( x = \text{any}(\text{GoodRole}) \)
\begin{align*}
\text{in} \quad (W \ [\geq \text{cert}] (\text{self}): \ \text{role}(\text{self}) = \text{Werewolf}) \text{ and } \\
(\text{phase}(\text{game}) = \text{DiscussionPhase}): \\
\quad \text{Forall } p \text{ in } \text{Players}: \ ((\text{eq}(p) = \text{self}) \text{ or } \\
\quad \quad (\text{seen}(p) = \text{self}) \text{ or } \\
\quad \quad W \ [\geq 0.7] (p): (\text{role}(\text{self}) = x));
\end{align*}

\( x = \text{any}(\text{Players}), \ p = \text{any}(\text{Players}) \)
\begin{align*}
\text{in} \quad (W \ [\geq \text{cert}] (\text{self}): (\text{role}(\text{self}) \neq \text{Werewolf})) \text{ and } \\
(W \ [\geq \text{cert}] (\text{self}): (\text{role}(x) = \text{Werewolf})) \text{ and } \\
(\text{phase}(\text{game}) = \text{DiscussionPhase}) \text{ and } \\
(\text{eq}(\text{self}) \neq p): \\
\quad W \ [\geq 0.7] (p): (\text{role}(x) = \text{Werewolf});
\end{align*}

\( r = \text{role}(\text{self}), \ p = \text{any}(\text{Players}) \)
\begin{align*}
\text{in} \quad (W \ [\geq \text{cert}] (\text{self}): (\text{role}(\text{self}) \neq \text{Werewolf})) \text{ and } \\
(\text{phase}(\text{game}) = \text{DiscussionPhase}) \text{ and } \\
(\text{eq}(\text{self}) \neq p): \\
\quad W \ [\geq 0.7] (p): (\text{role}(\text{self}) = r);
\end{align*}

\( r = \text{initrole}(\text{self}), \ p = \text{any}(\text{Players}) \)
\begin{align*}
\text{in} \quad (W \ [\geq \text{cert}] (\text{self}): (\text{role}(\text{self}) \neq \text{Werewolf})) \text{ and } \\
(\text{phase}(\text{game}) = \text{DiscussionPhase}) \text{ and } \\
(\text{eq}(\text{self}) \neq p): \\
\quad W \ [\geq 0.7] (p): (\text{initrole}(\text{self}) = r);
\end{align*}

**Listing 5.7** Goals for agents playing One Night Ultimate Werewolf.
and one in which they did not. Of the 840 possible worlds, the Rascal is present in $480^2$, so the Werewolf agent will consider $480 \cdot 2 + 360 = 1320$ worlds possible.

- Similarly, for each possible option the Robber could have stolen from, the agent will create a new possible world. Since there are 5 players, this means the Robber can steal from 4 of them, or not at all. Of the 1320 possible worlds, the Robber is considered to be present in the game in 720 of them. This means that the agent now considers $720 \cdot 5 + 600 = 4200$ worlds possible.

- Among all of these possible worlds, the player’s card was exchanged only if either the Rascal was sitting next to them and swapped a card or if the Robber stole from them. There are 720 worlds in which the Robber stole from the Werewolf player, and 720 in which the player’s card was exchanged by the Rascal. Note that 240 of these worlds are actually the same, because they correspond to the scenario in which a player’s card was exchanged and stolen. Additionally there are 60 worlds in which the Rascal and the Robber are seated next to each other, meaning the Rascal first exchanged the Werewolf player’s card with the Robber’s, and the Robber then stole their initial Robber card back and returned the Werewolf card to the player that originally had it. Overall, there are therefore $720 + 720 - 240 - 60 = 1140$ worlds in which the Werewolf player considers that their card has been exchanged.

Note that this calculation actually ignores the possibility of a second Werewolf player, which results in fewer worlds the Werewolf player considers possible, because they are told who the other Werewolf is, as well as fewer worlds in which their card was taken, because they could have been exchanged with the other Werewolf player. With 1140 of 4200 worlds in which the Werewolf player considers that they no longer have the Werewolf role, all of which are initially equally weighted, the (weighted) quality of the belief of the agent that they are still the Werewolf is $1 - \frac{1140}{4200} = 0.73$, which provides a guideline for which value to use for the definition of reasonably certain. If the required certainty is higher than this value, even a player that started as the Werewolf will no longer consider that they are still the Werewolf at the beginning of most games.

---

2 There are 4 possible players for the Rascal, and for each of these there are $6 \cdot 5 \cdot 4 = 120$ ways to assign cards to the other 3 players.

3 Similar to above, there are 480 of the 840 starting worlds in which the Robber is present in the game, but only in 240 are both the Robber and the Rascal present. In the previous step, the agent doubled the number of possible worlds for each world in which they considered the Rascal to be present, meaning the Robber and Rascal are considered to both be present in 480 worlds now. Additionally, there are another 240 worlds in which the Robber is present but the Rascal is not, resulting in a total of 720 in which the Robber is considered to be present.

4 Among the 480 starting worlds in which the Rascal is present, they are seated on either side of the Werewolf player in 240. In 120 of these worlds, the player considered the Robber to also be present, and thus created 5 worlds from each of them, while in the other 120 the Robber is not considered to be present. Finally, note that while the agent creates two possible worlds for each of these scenarios in a previous step, at this point only the worlds in which the exchange actually happened is relevant.
However, if the required certainty is lower than 0.5 players can simultaneously be reasonably certain that they are a Werewolf and that they are not a Werewolf, resulting in potentially non-sensical behavior.

Once an agent has determined which goals are applicable based on their estimation of the game state, they use the planning process described in section 4.1.7 to use the communicative actions described above to find a plan that achieves these goals as closely as possible. The result of this process is a plan, and an associated expected weighted quality $w$, which the agent will store. On subsequent turns the agent will follow this same process but only adopt the new plan if that new plan’s expected weighted quality of achieving its goal $w'$ is significantly greater than the old weighted quality $w$, which is the case if

$$w' \geq \alpha \cdot w$$

Note that the expected weighted quality is not updated once a plan has been formed, and the applicability of goals is also not re-checked. In other words, if an agent is reasonably certain that they are a Werewolf, and forms a plan to convince the other players that they are e.g. a Villager, they will stay committed to that plan, even if they subsequently become less certain about what role they have. This is done to make the agents more committed to their goals, even in the face of uncertainty. In the next section I will discuss several sample games to show the kinds of plans agents actually come up in practice, and how they change their commitment to a plan over time.

### 5.2.3 Example Scenarios

To better demonstrate the operation of the agents, I want to present several scenarios as they occur during game play, which plans the agents come up with in these scenarios, and when they choose to switch to a new plan. Note that these are just single, hand-picked examples, chosen for different aspects they demonstrate. I will present a more general investigation of the agents’ behavior and performance in the following section. First, I will discuss a typical game in detail, to show how the agents can be expected to behave in the majority of cases. After that I will show some hand-crafted scenarios which are less typical, and discuss how the agents perform in these. Finally, I will briefly discuss some quirks and unusual behavior the agents exhibit at times.

#### 5.2.3.1 A Typical Game

Due to the random role assignment, there is always the possibility that one faction obtains such an advantage that it becomes almost impossible for the other faction to win. Such a scenario presents the exception rather than the rule, though, or players would likely not be willing to play the game. In real life games this balance is achieved by carefully selecting which roles
are available in the game in a way that makes lopsided games more unlikely. By default, the Werewolves start with an advantage, because there are very few of them, and they know exactly who started on their team. To improve the chances of the Citizens it is therefore advisable to add roles that gain information, such as the Seer, or exchange cards, such as the Rascal, because these actions can also be used to gain knowledge. For my work I went a step further and even assigned the cards to the individual players statically, in order to craft interesting scenarios. The first game I want to present constitutes a fairly typical scenario. The starting role assignments are:

- **Anna** starts as the Werewolf.
- **Brian** starts as the Rascal.
- **Carol** starts as a Villager.
- **Dave** starts as the Seer.
- **Eowyn** starts as the Insomniac.

There are two main ways this game can play out: Either Brian, as the only player that can manipulate the players’ roles, exchanges his neighbors’ cards, or he does not. The challenge for Anna lies in determining that these are the two possible outcomes, and which of the two cases occurred. If Brian, for example, does not swap the cards, but then lies and says that he did, Anna needs to decide whether to trust him or not.

First, I want to present the actions from a game in which Brian indeed swapped his neighbors’ cards. Dave, as the Seer, looked at Eowyn’s card and saw that she is the Insomniac. On Anna’s first turn, she forms a plan to convince the other players that she is the Seer, which results in a plan that achieves that goal up to a weighted quality of 0.4 by repeatedly stating that she is the Seer\(^5\). Brian then decides to pursue the goal of convincing Anna that he is the Rascal. Interestingly, the plan he finds to achieve this goal up to a quality of 0.71 consists of stating that he swapped his two neighbors’ cards, because the Rascal is the only role in the game that can perform the swap action. Carol, who has no information just wants to convince others that she is the Villager, so she states that she started as the Villager, from which she expects a quality of 0.73. Note that Anna said that she is the Seer now, while Carol says that she started as the Villager, because, as far as the two players are concerned, those are the most believable statements they can make. Anna knows that she did not start as the Seer, and if the actual Seer has seen her card that would go against their beliefs, but the actual Seer does not know for sure that their card was not exchanged with Anna’s. Likewise, Carol knows for a fact that she started

\(^{5}\)This may seem like a low quality to achieve, but keep in mind that Anna is going first, and is basically guessing what she can convince the others most easily of.
as the Villager, but if someone exchanged their card, it will be harder to convince them that she is still the Villager.

The next player, Dave, adopts a goal of convincing others that he is the Seer, which he does by stating that he saw Anna’s card, expecting a weighted quality of 0.7 for his goal. Finally, Eowyn wants to convince the other players that she is the Insomniac, and forms a plan consisting of saying three times that she is the Insomniac now and three times that she started as the Insomniac. This is behavior often exhibited by players in real life that are the Insomniac, albeit they usually only repeat their claim if asked.

The next turn is Anna’s. In this example game, the certainty threshold for players to believe in their goals was set to 0.7, which is only slightly below the base quality they attribute to keeping their initial card. This means, as soon as someone casts doubt on a player still having their own card, they will believe that. In this case, Anna now believes that she no longer is the Werewolf because Brian stated that he swapped his neighbor’s cards. Anna considers a new goal of convincing Dave that she started as the Werewolf. She finds a plan which reaches this goal with a quality of 0.92 by stating that she started as the Werewolf. Note that other candidate goals for Anna would have included convincing e.g. Brian that she is a Werewolf, but since Dave already stated that he saw Anna’s card, the quality Anna expected to get was higher for directing her statement at Dave. Brian, on his turn, still wants to convince other players that he is the Troublemaker, and keeps following his plan which involves stating that he did swapped his neighbors’ cards. Carol now has an interesting dilemma on her hands, because she could believe Brian’s claims that her card was exchanged for Anna’s, and Anna just stated that she started as the Werewolf. Since she believes that there is a high chance that she is no longer the Villager, the goals that become available to her involve lying about her own role. When trying to find a plan for that new goal, Carol finds a plan to convince the other player’s that she is the Robber, because no one else said anything about the Robber yet, and the weighted quality of her still being the Robber is higher than for the other roles, because there are worlds in which she stole a card from Anna, and got back her Robber card. Her plan involves stating that she started as the Robber and that she is the Robber now, and she can reach the goal of convincing the other players of this fact with a weighted quality of 0.41. Recall that the plan she was pursuing so far had an expected weighted quality for its goal of 0.73. At this point, the level of commitment becomes important for Carol to decide how to proceed. If her level of commitment at this point is 0.5, she would adopt the new plan she found, because $0.41 \geq 0.5 \cdot 0.73$, but for a commitment level of 0.56 or more, she would stay committed to her old plan. As I will show later, commitment levels that are reasonable in practice are actually close to 1, which would mean that Carol would

\footnote{In practice, this is only a minor distinction, since the actual goal conditions are very similar. The main difference is that a Werewolf wants to convince all other players (that have not seen their card) of their innocence, while the Citizens just want to form clusters of trust.}
stick to her plan in this scenario and continue to state that she is a Villager.

For the remaining turns, the agents just reinforce the statements they already made, since none of their statements significantly change how certain anyone is about being a Werewolf anymore. At the end of the game, though, Anna votes for Carol with confidence of more than 0.9, but Brian is actually torn between believing Anna and believing Carol, and ends up voting for Anna with a confidence between 0.3 and 0.5. Because Carol actually believes that she is the Werewolf at this point, the quality of her belief for anyone else being the Werewolf is less than 0.3, and she ends up voting for Anna randomly. Dave, on the other hand, has seen Anna’s card, which is his strongest lead, and is also torn between Anna’s and Carol’s testimony and therefore ends up voting for Anna, with a confidence between 0.5 and 0.7. Eowyn, finally, does not have any information, and votes randomly for Brian. The result is that Anna is eliminated, but since Carol was the Werewolf, the Werewolves win. Note that the random votes from Carol and Eowyn could only have affected which player was eliminated if they had both gone to Brian, but since he is not the Werewolf either, Carol would still have won. In contrast, in another run of the game where Carol’s commitment level was set 0.5, and she actually switched plans and started to claim that she was the Robber, Brian’s confidence in his vote decreased even more and he ended up voting randomly. Dave, on the other hand, actually voted for Carol, because in the scenario in which she tells the truth about stealing from Anna, but Brian lied about swapping cards, as well as in the scenario in which Brian told the truth about swapping cards, but Carol lied about stealing, she would end up with the Werewolf card that Brian saw. In this particular run, three of the random votes still went to Anna, and another one to Brian, which means Carol would still have won, but by introducing more uncertainty her win became more dependent on luck.

Finally, consider what would happen if agents became less suspicious of their role assignment, meaning that the certainty threshold required for an agent to believe that they have a particular role is set to a lower value, such as 0.5. In this case, Anna would not immediately assume that she is no longer the Werewolf and therefore not volunteer the information that she started as the Werewolf immediately. Instead, it takes 2 turns from Dave stating that he saw Anna’s card and that it was a Werewolf card until the game takes the same course as above. While this may seem like a minor change from before, keep in mind that this was just one possible course of the game. Below I will describe the effect the certainty threshold has on the outcome of the game more generally.

5.2.3.2 Unusual Scenarios

While the game described above demonstrates how the agents play typical games, it is also interesting to observe how they behave in corner cases. As a first example, consider a game in which both Werewolf cards are in the center of the table, and no player is a Werewolf. The game
rules state that the Citizens only win in this case if every player gets exactly one vote. As an example for such a game, I assigned Anna the Robber, Brian the Rascal, Carol a Villager, Dave the Seer, and Eowyn the Insomniac. It so happened that the Robber and the Troublemaker both declined to perform their special action, while the Seer, Dave, looked at Carol’s Villager card. During the discussion phase, all players try to convince the others of their own roles. Anna actually does this by claiming to have stolen Eowyn’s role, while Dave claims to have seen a Robber card. Both of these statements are lies, but they are actions the characters could have performed, so they do not affect their believability. The other three players simply state what roles they started with. The result of this is that none of the players have a lot of confidence in anyone being the Werewolf at the end of the game, so they end up voting randomly. The one exception to this is Brian, who had a confidence between 0.3 and 0.5 that Carol was the Werewolf. In this particular run, one of the random votes went to Brian, and three to Anna. On one hand, this shows that if there is no Werewolf, the players do not end up wrongly suspecting someone strongly. On the other hand, it also demonstrates weaknesses with the static voting system, because the players just end up voting randomly instead of coordinating their efforts to not kill anyone. Another case not handled gracefully by the voting system are games with more than one Werewolf. For example, if Anna and Brian are Werewolves, Carol and Dave are Villagers, and Eowyn is the Insomniac, none of the Citizens can gather enough information to actually know who to vote for for certain, but Anna and Brian will vote for each other, since each of them believes that the other is a Werewolf with the highest weighted quality among all players.

Finally, I want to briefly discuss a scenario in which the game heavily disadvantages the player that starts as the Werewolf. For instance, if Anna starts as the Rascal, Brian as the Werewolf, Carol as Robber, Dave as the Seer and Eowyn as the Insomniac, every single non-Werewolf player has some information at their disposal. Furthermore, there is a high chance that someone takes Brian’s card, and knows that they end up as the Werewolf. Additionally, since Brian is not the first player, Anna’s initial statement that she is the Rascal adds enough uncertainty that Brian is no longer reasonably certain that he is the Werewolf, but he is also not reasonably certain that he is a non-Werewolf, so he says nothing. In the game, Dave actually looked at Brian’s card, and knows that he started as the Werewolf. Additionally, Carol stole Anna’s Rascal card, so she also believes her. The most interesting role, however, is Eowyn’s, since she knows for a fact that she is the Werewolf, and after all other players have made their statements she also has a high weighted quality for the belief that the other players have a high weighted quality for her being a the Werewolf. This is one of the rare instances where the planning process basically breaks down, because no action Eowyn can perform improves the other players beliefs, and so she does nothing. However, Brian changes his assessment of the situation enough to believe that he is no longer a Werewolf, and actually adopts a goal to convince Dave that he started as a
Werewolf, which is now in Eowyn’s hands. Since Dave has seen Brian’s card, and has stated so, Brian already believes with a high quality that Dave believes with a high quality that Brian started as the Werewolf, so he just states that he did not swap his neighbors’ cards. The interesting part happens at the end of the game. While most players had high quality beliefs that the other players had high quality beliefs in their roles, most of them were wrong. Anna, for example, believed Carol’s statements that she was the Robber just as much as Brian’s, but since Brian never outright stated that he started as the Werewolf, Anna voted for Carol with confidence between 0.3 and 0.5. Brian, likewise, believed Carol that she was the Robber, but also Anna that she was the Rascal. As such, she was not convinced of anyone and ended up voting for Carol randomly. Carol was actually confused by Anna’s statements that included that she is the Rascal, which Carol knew to be factually false, and hence she voted for Anna with confidence greater than 0.9. Dave saw Brian’s card as being a Werewolf, and thus voted for Brian, while Eowyn knew that she was the Werewolf, and had no information to suspect anyone else of being a Werewolf too, so she also voted randomly, choosing Anna.

5.2.3.3 Behavioral Quirks

From the description of the games and scenarios above it may already have become apparent that the agents exhibit certain behavior patterns that might be considered unusual. Here, I briefly want to discuss some of these quirks, address why they occur, and how they could be addressed in future work, if necessary. First, the nature of how the game actions are encoded, together with the definition of the weighted quality, cause plans to often contain repetitions of the same action. For example, stating that an agent is a Villager increases the weight for all worlds in which other agents consider that the agent is not a Villager, and performing the action multiple times simply increases the weight further. This is not necessarily unwanted behavior, though, because it corresponds to players reinforcing their own statements which is often done in actual games. Regardless, as I will discuss in more detail in the next chapter, the exact repetition of the same statement is something that seems odd to human observers.

Another behavior the agents often exhibit, which can also be observed in the games described above, is that they skip stating their role if the action they claim to have performed also reveals their role. For example, rather than stating that they are the Robber and that they stole from a player, the agents only make the second statement, because it logically implies the first anyway. This is actually an efficient way to convey the same information, and humans also pick up on it. Alternatively, if players want to state that they did not start as a certain role, they might state that they did not perform a certain action. If this behavior is undesired, the actions corresponding to the statements could be equipped with preconditions that control the flow of conversation.

A challenge with epistemic logic is the situation in which an agent’s worlds disappear, sc.
by being eliminated, and the agent ends up believing everything. In order to prevent this from happening, worlds are never eliminated by actions during the discussion phase, but statements that go against what the agent knows to be factual will devalue every world in that agent’s mental model, meaning that the agents considers all their worlds to be lies. This is usually not noticeable, because other statements by other agents will cause only a subset of worlds to be considered less likely, but in edge cases, such as the game above, it causes the agent’s certainty to be distorted.

Lastly, the agents are not very interactive between each other. When one agent claims that they are the Seer, another may simply do the same. This also makes sense, because it undermines the first agent’s believability, but human observers would prefer the agents to address each other directly rather than making generic statements to everyone. Currently, this is a limitation of the actions available to the agents, but I believe a future model would benefit from increasing the focus on dialog between agents, rather than broadcasting information.

Overall, none of these behavior patterns affect game play directly, but as I will discuss in the next chapter, they are noticed by human players. In this chapter, however, I will next describe how the agents perform when playing in games against each other, describe the effect different parameters have on agent performance, and compare the performance of different agent types.

### 5.3 Simulation Experiments

In the previous section, I have shown how the agents behave in several individual games that were hand-crafted and -picked for demonstration purposes. In this section, I want to present a more quantitative analysis of the agents’ performance. To do this, I ran several experiments with different starting configurations, agent types and parameter settings. I will first describe the general setup of the experiments, the relevant parameter variations, and agent types, as well as what was measured. I will then present select results, with more detailed results contained in section B.1 in the appendices, and finish with a short discussion of the interpretation of these results.

#### 5.3.1 Design

The main focus of this chapter is how different variations of the agents discussed above perform in terms of win rates in the game. The most important parameter to change, which represents the underlying idea of my thesis, is the commitment an agent has to their plan, called $\alpha$ above. Additionally, the implementation of the agents provides some tuning parameters. Because the planning process can never actually reach most of its goals, in particular the goals that involve convincing other agents of facts in the world, it is interrupted after a certain number of states
have been expanded. To verify that this cut off was not done too soon, I performed runs for different cutoff values, to show that the agent performance does not change significantly after a certain number of states are expanded. Additionally, the threshold value for a weighted quality to be considered reasonably certain has an effect on agent behavior as discussed above, and therefore I ran simulations with different values to determine what that effect is. Furthermore, the starting configuration of how roles are assigned to different players affects how easy or hard games are to win for each of the two factions.

While running simulations that cover all variations of these parameters against each other would provide the most information, the computational requirements of these simulations are non-trivial. Each individual game runs in $1 - 2$ minutes, but to get results within $\pm 1.5\%$ in the 95% confidence interval, 4000 games are required [Yam73]. Even with 10 games running in parallel, the expected time to complete one setting is therefore about 8 to 10 hours. To keep the computational requirement within reasonable limits, I therefore performed some exploratory experiments to determine interesting settings for the parameters. This resulted in an experimental setup consisting of four major experiments:

- One experiment comparing different values for the number of states that are expanded before the planning process is terminated.
- One experiment in which the players are assigned roles where the Werewolf player's card can not be exchanged with any other player, comparing different levels of commitment.
- One experiment in which the Werewolf player's card can be exchanged with another player's card, also comparing different levels of commitment.
- One experiment in which the Citizens have a strong information advantage over the Werewolf player, tipping the game heavily in their favor, comparing different values for the quality threshold that is considered reasonably certain.

The second and third experiment, which involve games that can be considered typical, constitute the main results of this chapter. As I will show below, in the case where the Werewolf player's card can be swapped with another player, the level of commitment affects the outcome of the game. For each of the four experiments I performed at least 500 runs per setup, leading to results that are accurate to at least $\pm 4.4\%$ in the 95% confidence interval. However, for the two main experiments, I performed 4000 runs per setup, with results accurate to $\pm 1.5\%$ in the 95% confidence interval. Results were collected within about 6 weeks of running experiments on a standard desktop machine dedicated to running 10 games in parallel.

For each experiment, the agents were assigned one of several different agent types. As a baseline, there was an agent that performed actions randomly during the discussion phase, but
still voted according to the scheme presented above. The other agent types used different values for \( \alpha \), including 0, representing capricious agents that always change to whichever the best plan is on any turn, and \( \infty \), which results in fanatical agents that never change their plans. Other values for \( \alpha \) were chosen between 0.1 and 5, with a focus on values close to 1 since these represent the case in which a new plan is directly compared to how the existing plan was evaluated on a prior turn. In each game, all agents starting as Citizens were assigned the same agent type, and compared to a different agent type that was assigned to the player starting as the Werewolf. After the predefined number of games, I measured in which percentage of games the player that started as the Werewolf won, even if that player was now on the Citizen’s team. This was done to measure the performance of each agent type. For example, if Anna started as the Werewolf, and was assigned the capricious agent type, and the Citizens played as balanced agents with \( \alpha = 1.25 \), I report the percentage of games that Anna was on the winning team, which gives a measure for how well capricious agents perform against balanced agents with \( \alpha = 1.25 \). Notationally, the baselines of random, capricious, and fanatical agents are called by these names, while agents that actually determine when to switch plans are called balanced, with the value \( \alpha \) used for the agent in parenthesis. For space reasons, these names are abbreviated \( r \), \( c \), \( f \), and \( b \) where appropriate.

In the tables in section B.1 in the appendices the rows correspond to the agent type assigned to the Citizens and the columns to the agent type of the Werewolf player. Therefore, the result for this game would be found in the row labeled capricious, and the column labeled \( b(1.25) \). By convention, Anna will always be the Werewolf player, which means the tables can be read as containing the win rate for Anna. I will now describe the most interesting results obtained from each of these experiments.

### 5.3.2 Results

In this section, I will discuss the most important results from each of the four experiments. Recall that the margin of error in the 95% confidence interval is at most \( \pm 4.4\% \) for experiments in which 500 games were played, and \( \pm 1.5\% \) for the experiments in which 4000 games were played. However, as the results skew away from a win rate of 50%, the accuracy increases, and should be interpreted accordingly. For example, if the true win rate was 5%, 500 games would result in an estimate of the win rate within \( \pm 1.96 \cdot \sqrt{\frac{0.05-0.05}{500}} = \pm 1.9\% \) in the 95% confidence interval. This is particularly relevant for the last experiment, in which the games were particularly disadvantaging the Werewolf player.

#### 5.3.2.1 Number of Expanded States

The first experiment I present investigates how quickly the planning process finds an adequate solution. If the planning process can only expand very few states, it is less likely to find a
reasonable solution. As an extreme example, if the number of states that can be expanded is 1, then the planning process will expand that state with all possible actions resulting in one new state for each action, and return a plan consisting of the single action that produces the state with the highest weighted quality for any available goal. As more states can be expanded, the planning process can actually start producing plans that consist of more than a single action. On the other hand, once the planning process could expand a certain number of states, expanding more states may not improve the quality of the resulting plan significantly anymore.

Ideally, the hypothesis that was tested would be that expanding more than some number of states will not change the outcome of the game anymore. However, in practice, computational and especially memory requirements of the planning process grow quickly. Expanding more than 200 states at a time results in run times of several minutes, and a memory requirement of more than 4GiB per game, making such configurations impractical. The experiment I performed therefore focused on finding a lower limit for how many states have to be expanded to result in reasonable game play. Figure 5.1 shows the win rate of the player starting as the Werewolf in several different agent setups for different limits $n$, defining many states the planning process could expand before being terminated in a typical game. The game consists of Anna as the Werewolf, Brian a Villager, Carol the Troublemaker, Dave the Seer and Eowyn the Insomniac. The expected win rate of the Werewolf player is about 50% in this setup for intentional agents, as discussed in more detail in section 5.3.2.2. Note that the performance of the agents for very small $n$ is degenerate in favor of the Werewolf player, because short plans for players that believe they might have been swapped often include stating that they started as a Werewolf. With $n \geq 25$, though, the players perform less erratically, and the game is more balanced, as expected. Importantly, the performance of the agents does not change significantly as $n$ increases beyond that. Nevertheless, to account for eventual differences in behavior in other scenarios, I used $n = 100$ for the remainder of the experiments.

5.3.2.2 No Swapping of the Werewolf

The second experiment I want to present mostly constitutes a verification that the agents behave reasonably in a typical game, and that there are no differences between agent types when there should not be. Consider a game in which Anna starts as the Werewolf, Brian as a Villager, Carol as the Rascal, Dave the Seer and Eowyn the Insomniac. Anna will always stay the Werewolf, and the exchange of cards between Brian and Dave have no effect on their team affiliation. For this experiment, I had random, capricious, and fanatical agents play against balanced agents with values for $\alpha$ set to 0.5, 0.66, 0.8, 1, 1.25, 1.66, 2.5, and 5. Unless the agent type for one of the teams was random, Anna’s win rate was within 1.5% of 50.25%, indicating that the level of commitment had no effect on her performance. When Anna played as random, her win rate
Figure 5.1 Win rate in a typical game for various agent types for different numbers of states that are expanded before the planning process is terminated.
was within 1.5% of 15% against all levels of commitment, and when the Citizens played as random, Anna’s win rate was within 1.5% of 89% for all levels of commitment. This consistent performance across all combinations of agent types provides evidence that the decision to change one’s story only becomes meaningful if the game situation warrants it, as will be described in the next section.

5.3.2.3 Swapping the Werewolf

For the third experiment, I considered a variation of the previous game, in which the Werewolf’s card could be switched with another player’s. In this setup, Brian is the Rascal and Carol a Villager, which means that Anna’s card may be switched with Carol’s, turning her into an ordinary Villager. Like before, I tested several different commitment levels against random, capricious and fanatical agents, for this experiment with values for $\alpha$ set to 0.1, 0.2, 0.5, 0.66, 1, 1.33, and 4. These values were chosen to extend the experiment to less committed agents.

The first noticeable result was that the performance of random agents actually improved, because their statements make it harder for the rational agents to determine the truth. However, when Anna was playing as random, she still only won between 25% and 30% of the games against the other agent types, whereas when the Citizens played as random, Anna won between 53% and 58% of the time. Note that for both of these cases, different levels of commitment performed differently. When Anna played as random, she achieved her lowest win rate when the Citizens played as balanced(0.66), where Anna only won in 24.8% of the games. Anna’s highest win rate of 30.1%, on the other hand, was achieved against Citizens playing as balanced(0.1). More interestingly, Anna’s performance also depended on her level of commitment against Citizens playing as random, and against Citizens playing as capricious, as shown in figure 5.2. Note that in both cases the extreme value can be found at $\alpha = 1$, but against random agents this represents Anna’s best performance, with a win rate of 57.2%, whereas against capricious agents it was her worst performance, with a win rate of 46.9. This means, not only does an agent’s performance depend on their own level of commitment, this dependency also changes with the other player’s strategy. Finally, note that fanatical agents were hardly affected by the difference in setup, still resulting in a win rate for Anna close to 50% when Anna played as fanatical, as well as when the Citizens played as fanatical.

5.3.2.4 Quality Threshold

The last experiment in this section was performed to investigate the effect of how certain players should be of their roles before committing to a plan, especially in atypical scenarios. Consider a lopsided game, in which Anna starts as the Werewolf, Brian as the Rascal, Carol as Insomniac,  

\footnote{This figure has been adapted from a previous publication \cite{Ege18}}
Figure 5.2 Anna’s win rate based on her level of commitment $\alpha$ against random and capricious Citizens
Dave as the Robber and Eowyn as the Seer. Anna’s win rate is below 5 percent for most assignments of agent types, but when playing against highly committed ($\alpha = 2$) or even fanatical Citizens, her win rate drops to 0%. This is in part because the Citizens do not change their strategy, and basically behave as if nothing happened to their cards. Anna, on the other hand, because of the certainty threshold of 0.7, is quickly suspicious of which team she is actually on now, and will try to argue on behalf on the Citizens, who completely ignore her. To test the effect of this certainty threshold on the agent’s performance, I ran experiments in two variants: One in which Anna was a fanatical agent, with the Citizens being balanced with $\alpha = 2$, and one in which the agent types were assigned the opposite way. Figure 5.3 shows Anna’s win rate for several different certainty thresholds for both of these scenarios. As can be seen, if the certainty threshold is set to 0.5 or less, Anna actually wins a small, but non-zero percentage of games, whereas the default value of 0.7, or an even higher value of 0.9 results in Anna winning 0% of the games in either case. While this may indicate that certainty thresholds of 0.5 or less are therefore preferable, the games themselves reveal a problem with them: When Anna does not need a high certainty to consider herself either a Citizen, or a Werewolf, she may consider herself to be both, and flat out state that she is a Werewolf, because she knows the identity of a Werewolf with reasonable confidence, and she believes that she is on the side of the Citizens. In the next section I will discuss the main insights gained from these experiments, and how I applied them to making the agents play better with human players.

5.3.3 Discussion

While the ultimate goal of my work is for my agents to play with human players, the experiments and scenarios described above nevertheless provide interesting insights into how the agents make their decisions, and limitations they have. For example, while most of the experiments were performed with a certainty threshold of 0.7, there are several instances in which the Werewolf player becomes too suspicious about their own role. On the other hand, setting the certainty threshold to 0.5 may result in aberrant behavior where the agent believes they are on both teams simultaneously. Further testing, particularly with the implementation presented in the next chapter, revealed that a threshold of 0.6 provides a good trade-off between suspicion and gullibility, so to speak.

On a technical note, the time needed to run experiments presented a challenge, which will only be exacerbated when the agents should play with human players, where timely responses become more important. To address this, I performed some optimizations in Ostari that are geared towards the particulars of how the One Night Ultimate Werewolf actions are implemented. Consider the actions in listings 5.4 and 5.5. In both of these actions, the value of a property is first used in a precondition, and then again in the condition of an if, but compared to a different
value. Because these comparisons are not only performed in the actual world, but in every world agents consider possible that other agents consider possible, a large number of these lookups are performed. The only other operation these actions perform is increasing the weight of certain worlds, which is involves updating a single floating point number that is kept as a separate field. As a result, the time to look up the property value of `initrole` or `rob` dominates the time needed to execute these actions. The optimization I performed is a limited case of common subexpression elimination, an optimization procedure commonly performed by many compilers [Coc70]. Rather than identifying all cases of reoccurring subexpressions, this optimization targets only the case in which a property is referenced by a `precondition` followed by an `if` statement. Using this simple optimization reduces the time needed to execute any single action by almost half, reducing game run time to typically less than 1 minute for a complete game.

Finally, the key insight from the experiments in this chapter is that while the level of commitment does have an effect on agent performance, that effect is relatively small. While the agents perform reasonably well against random agents, when comparing the performance of different commitment levels with each other the win rate for Anna never fluctuates by more than 5% for different commitment levels. The question I want to answer in the next chapter is whether these rather small changes in behavior are noticeable enough for human players that
they would prefer one behavior over another. Additionally, there are several adaptations that had to be made to make the game playable and enjoyable for human players, that I will also describe in the next chapter.
In the previous chapter I have discussed how AI agents can play the game of One Night Ultimate Werewolf in principle, how commitment to a plan is defined, and the effect of different levels of commitment on the agents’ performance in the game. While AI agents that play against other AI agents and exhibit reasonable behavior are interesting, an important aspect of my work is the interaction between these AI agents and human players. In this chapter I will discuss some additional requirements for the agents in order to have them play with human players, how to address these requirements and present the technical execution of this approach. As before, I am particularly interested in the effect different levels of commitment have on game play. However, rather than simply comparing raw win rates, human players can be asked how believable the AI agents were, or how much sense their actions made. I will present the results from an experiment with human subjects, including an interpretation of the effect of different commitment levels, and also discuss other lessons learned from the experiment, and what future challenges they imply.

6.1 Approach

The basic approach for playing One Night Ultimate Werewolf with humans does not differ greatly from how the agents I presented in chapter 5 work. They are still provided with a set of high
level goals that they should reach, actions they can perform to achieve these goals, and have to find plans to reach them. As before, the aspect I am most interested in is a comparison of different levels of commitment to a plan that an agent can have. However, whereas before the agents implicitly understood that the actions performed by other agents served a goal to convince the other players of something, and they themselves volunteered information in order to further their own agenda, humans expect more interaction. I will now outline what this means for my agents, and then proceed to describe how the agents were modified to be more interactive.

6.1.1 Requirements for Playing with Humans

As I just mentioned, human players expect to be able to interact with AI agents in a game in a way that the AI agents themselves do not technically need. Say agent A, who happens to be a Werewolf, adopts a plan to convince the other players that they actually are the Seer. Agent A might go about this plan by simply repeatedly stating that they are the Seer, which, as I described above, has the effect of reinforcing the statement in order to convince the other players that this is really the case. The Seer has the ability to look at another player’s card during the night phase, and if agent A really was the Seer, they could share that piece of information, but because the other agents are busy convincing other players of the truth behind their own story, no one will inquire as to what agent A might have seen. Humans, on the other hand, are naturally inquisitive, and to a claim of I am the Seer a human is likely to want to responded with Ok, so what did you see?. However, the encoding of the game as described above contained no actions that would correspond to questions. In order to make the game more aligned with human expectations, I therefore added additional actions that allow players to ask:

- What role a player started with.
- What role a player believes they have now.
- What card a player claims to have seen.
- What card a player claims to have stolen.
- Which cards a player claims to have swapped.

The goals used by the AI agents never lead to plans that would include any of these query actions, and therefore the AI agents will never ask anything. I will discuss the limitations imposed by this effect later. On the other hand, human players can use these actions to inquire about information, that the AI agents may choose to answer, but they are not required to do so. This is necessary because the human player should not be able to dictate what the AI players are going to say, but it is still desirable for the AI agents to react to the human player. I will now discuss how this balance was achieved.
askRole(p: Players, p1: Players)
{
    precondition isHuman(p) == True;
    common topic(p1) = role
}

Listing 6.1 The “ask which role another player believes they have” action for One Night Ultimate Werewolf, with question answering.

6.1.2 How to Answer Questions (or not)

When playing One Night Ultimate Werewolf my agents already have to decide which goal to pursue, such as convincing other players that they belong to a certain faction. Questions that are asked by other players can then simply serve as additional goals for the agents. When an agent then decides which goal they want to achieve, they can choose between convincing other players of a certain fact, or answering a question they were asked, and will pursue whichever plan they determine to be better. In order to do this, there are three necessary components:

- A question has to create/enable a new goal for the player that was asked.
- There need to be actions that can be used to satisfy the goal.
- Agents need to be able to choose these new goals.

To encode question answering in Ostari, I added an additional (common knowledge) property called \textit{topic} that represents what an agent was asked about\textsuperscript{1}. This property assigns a value from a set \textit{Topics} to each player, which contains values such as \textit{role} to represent that the player was asked about which role they believe that they currently have, or \textit{none} to represent that the player has not been asked anything or has answered the question posed to them. At the moment, agents can only have one active topic, i.e. additional questions will change what the agent should respond to. Listing 6.1 shows the action of asking another player which role they believe to have, which simply sets the \textit{topic} attribute to \textit{role} for the player that was the subject of the question.

Because agents can already make statements about which role they believe to have, it is not necessary to introduce new actions for that purpose. Instead, the existing actions only need to be changed to ensure that when a question is answered that fact is recorded correctly. Listing 6.2 shows the modified \textit{claimRole} action. As before, an agent claiming to have a certain role will increase the weight of all worlds among the worlds other agents consider possible in which

\textsuperscript{1}Recall that common knowledge properties are those that are known to all agents, i.e. after asking a question every agent will know what was asked and of whom. Making this property common knowledge reduces the memory footprint of the game, and since questions are asked publicly this does not interfere with any reasoning performed by the agents.
they do not have that role, thereby decreasing how likely the other agents consider that world to be the actual world. In addition, though, the action also resets the \textit{topic} property for the agent performing the action to \textbf{none}, if that agent was asked about their role.

Finally, to tie the asking and answering of questions together, and integrate them into the existing encoding, the goals have to be modified slightly. For example, previously when an agent had a goal of convincing player \textit{A} that they are the Villager, that would only involve attempting to raise the weighted quality of the corresponding belief over a certain level. Now, each goal additionally also requires that the \textit{topic} quality is reset to \textbf{none}, as shown in listing 6.3. This has the result that plans that successfully change the \textit{topic} of an agent to \textbf{none} will be preferred over plans that would result in an otherwise equal quality that do not answer the question, but it does not force the agent to answer, and it also allows agents to construct plans that just incidentally contain an answer to a question. For example, consider that an agent was asked what they believe their role to be. Because of the knowledge they obtained during the night phase, they may want to construct a plan to achieve a goal of convincing the other players that they know the identity of a Werewolf. Ordinarily, this plan would involve stating that they started as the Seer, saw the card of a certain player and that the card they saw was a Werewolf card, but to achieve the addition to the goal of answering the question, they may add an action to claim that they believe that they are still the Seer.

These three components, question actions, the modified actions to answer questions, and goals that involve the answering of questions, allow the agents to appear more interactive, while

\begin{verbatim}
claimRole(p: Players, r: Roles, r1(p): Roles)
{
    precondition role(p) == r1;
    if (role(p) != r)
    {
        addWeight(1);
    }
    else
    {
    }
    ;
    if (topic(p) == role)
    {
        common topic(p) = none
    }
    else
    {
    }
}
\end{verbatim}

\textbf{Listing 6.2} The “claim to be a Role” action for One Night Ultimate Werewolf, with question answering.
\[ r = \text{initrole}(\text{self}), \quad p = \text{any}(\text{Players}) \]

\[
\begin{align*}
\text{in} & \quad (W \text{[}>\$\text{cert}](\text{self}) : (\text{role}(\text{self}) \neq \text{Werewolf})) \text{ and} \\
& \quad (\text{phase}(\text{game}) \text{ == DiscussionPhase}) \text{ and} \\
& \quad (\text{eq}(\text{self}) \neq p): \\
& \quad \quad (W \text{[}>0.7](p) : (\text{initrole}(\text{self}) == r) \text{ and} \\
& \quad \quad \quad (\text{topic}(\text{self}) == \text{none}); \\
\end{align*}
\]

**Listing 6.3** A potential goal pattern for an agent: Convince another player what role you started with.

preventing the human player from being able to completely control their behavior. Note that even if the agent comes up with a plan that involves answering a question, that action may not be the first action in the plan. Agents may also be too committed to their existing plans to respond to the player’s question, because that would involve changing to a plan that is overall worse. Fanatical agents, in particular, are not expected to ever answer questions posed to them, unless that is already part of their existing plan, or they finish executing whatever plan they had. As I will discuss below, this is more apparent to players more familiar with the game, because they are more likely to ask questions that the agents were not already going to answer anyway.

### 6.2 Study Design

To evaluate the effect of different commitment levels, I performed a human subject study in which participants recruited online played the game with different agents in a web browser. In this section I will first discuss the technical implementation of the study, including how the game was actually presented to the subjects, and how I integrated the Ostari system with the web frontend. I will then describe the procedure used for the study, which treatments the participants experienced and which questions I asked them.

#### 6.2.1 Game Setup

As described in chapter 4, the Ostari runtime system is implemented in Haskell, and runs as a single console process that reads an Ostari input file and produces the corresponding output, optionally accepting input for player actions. Because of the relative ease of subject recruitment it was desirable to have the game playable in a web browser. To bridge between the console application and a web frontend, I used a simple HTTP server implemented in python, that managed the individual games that were running, created Ostari input files according to the initial role setup and then started an Ostari process that was responsible for managing agent beliefs and their actions. The python server process would then accept requests for game actions via HTTP, communicate them to the Ostari process using a pipe for standard input and interpreted the
output read from another pipe and translated it into an appropriate HTTP response, encoded in JSON. While it would in principle have been possible to present this information in an HTML page and have players play that way, I also implemented a user interface in Unity to make game play more enjoyable for the user. Figure 6.1 shows an overview over how this setup was deployed for the study. In the next section I will briefly describe how the Unity application is implemented.

6.2.2 User Interface

The interaction point for users with the game consists of a Unity application that was exported to WebGL and can therefore be played in any webbrowser with JavaScript support without the need for any plugins. The implementation of this user interface is very straightforward, because all the actual reasoning is done inside the Ostari process on the server. Figure 6.2 shows a screenshot of the game during the discussion phase. To make the game more intuitive, the players were given names, corresponding to their seat in the game, with Anna always going first, followed by Brian, Carol, Dave, and finally the human player. Any action taken by the player is sent as an HTTP request to the Python web server, and the JSON response displayed appropriately. For example, if a card is revealed to the player, the JSON response will contain the name of the card and which player it belongs to and the user interface will change the texture of the corresponding card accordingly and flip it over. This texture change is done because the Unity application has no knowledge of which player has which card until the player explicitly gains knowledge of it, to avoid security issues by players sniffing network traffic or using a JavaScript debugger. While the probability of players actually resorting to such measures during the course of my experiment was low, and security is not the focus of this thesis, I believe this is worth mentioning because it represents an extension of hidden actions and information. The verification of security measures like this one could be an interesting application for Dynamic Epistemic Logic in future work.

Figure 6.1 A system diagram of the deployment of One Night Ultimate Werewolf for the study.
Figure 6.2 A screenshot of the Unity implementation of One Night Ultimate Werewolf (Art for the roles kindly provided by Hui-Yin “Helen” Wu).
The main part of the game, and also the main focus of my work, is the discussion phase, which is what is depicted in Figure 6.2. Note that the communicative actions available to the player are represented by dropdown menus which the player can use to construct statements and questions from predefined templates. Each statement consists of a subject, a verb and an object. The verbs correspond to the actions in the Ostari encoding of the game, such as “started as” corresponding to the claimInitRole action, while the subject and object correspond to the parameters. Questions, on the other hand, consist of an addressee, a question word and a question. Once a player has made a statement of asked a question, the AI agents will make their own statements in order, until it is the player’s turn again. In the top left corner of the screen the game has little tokens that players can use to keep track of what the AI agents claimed to be. The AI agents completely ignore these tokens, but their use is recorded in the game logs, and could be used to gauge how believable a statement made by the agents is. In the next section I will describe how the games the participants of the study played were determined and what other information I collected.

6.2.3 Study Procedure

To compare different levels of commitment, I decided to have the participants play against three different agents, capricious, fanatical, and balanced with $\alpha = 1$, because these represent the two extreme cases and a middle ground that performed reasonably well in the computational study described in chapter 5. Similar to the Hanabi study described in chapter 3, I controlled the starting conditions, i.e. the cards dealt to the players, to be in one of a limited set of possibilities:

- The participant starts out as the Rascal, with the Anna having a Werewolf and Dave having a Villager card, respectively, which the participant will swap with each other if they so choose. Brian starts as the Seer and Carol is given the Robber card. In this scenario it is impossible for the participant to end up as the Werewolf, but they have the option of swapping Anna and Dave which leads to some strategic choice on whether and when to reveal that information.

- The participant starts as the Seer, Dave as the Rascal, and Carol as the Werewolf. Anna and Brian start as the Insomniac and a Villager, respectively. This scenario allows the player to learn the identity of one of the other players, which is especially useful when they see the Rascal or the Werewolf, because they can deduce that they might end up as the Werewolf. The key aspect of this setup is that the player does not start as the Werewolf but may end up being the Werewolf player at the end.

- The participant starts as the Werewolf, Anna as a Villager, Brian as the Insomniac, Carol as the Rascal and Dave as the Seer. In this scenario, the player is guaranteed to stay the
Werewolf until the end, but the other players have a lot of information available between them.

Each participant was asked to play three games, one of each of these scenarios, in random order. Additionally, they were assigned one of the three treatments, with capricious, fanatical and balanced agents to play against for each of the three games, also in random order, with no repeats. In other words, each player played three of the nine possible combinations of scenario and agent type, with no repeats in either category. After each game the participants were asked to answer the following prompts:

1. The AI agents seemed to change what they were doing based on my actions.
2. The AI agents’ actions made sense.
3. Which of the following statements by <Agent name> did you believe at the time they made it?
4. Which of the following statements by <Agent name> made strategic sense to you at the time they made it, whether or not you believed it?
5. The AI agents played well.
6. It was fun to play with the AI agents.

For questions 3 and 4, <Agent name> was replaced with the name of the agent that started as the Werewolf, or the Seer in the scenario in which the participant started as the Werewolf. In each of these two questions the participant was presented with the actual statements made by the agent in question, and, to help them place the statement in the flow of the game, a reminder of what the participant’s own statement that followed was. The other four prompts were answerable on a five point Likert scale, with options of Strongly disagree, Somewhat disagree, Neither disagree nor agree, Somewhat agree, and Strongly agree. In addition to these questions, participants were provided with a text input field with the prompt Do you have any comments about the behavior of the AI agents?, in order to collect additional feedback.

After participants played three games with the setup described above, they were asked for some basic demographic information, with the answer options shown in parenthesis:

1. What is your age? (18-29, 30-39, 40-49, 50-64)
2. How familiar are you with the board and card games in general? (never play, rarely play, sometimes play, often play)
3. Do you consider yourself to be a (board) gamer? (yes/no)
4. How familiar are you with the game One Night Ultimate Werewolf? (never played, played 1-10 times, played 10-50 times, played over 50 times)

5. When was the last time that you played One Night Ultimate Werewolf before this experiment? (never, over a year ago, 3 months to a year, less than 3 months)

6. What is your estimate of what percentage of games is won by the Werewolf team? (0-25, 26-50, 51-75, 76-100, can’t tell)

7. The user interface was usable (Strongly disagree, Somewhat disagree, Neither disagree nor agree, Somewhat agree, Strongly agree)

8. For this study we have recorded your answers to this survey, as well as a log of actions that you performed in the game. We have not recorded your IP address or any other information that could be linked back to you. Do you agree that we make your answers to the survey and the game log publicly available for future research? (yes/no)

Note that the last question, similar to the Hanabi study, grants permission to make the game logs and survey answers publicly available to further future research attempts. Question 6 was not directly relevant to the study, but provides an interesting comparison for players' expectations versus how well the AI agents perform.

6.3 Results

For my experiment, I recruited participants online on social media, including Facebook, Twitter, BoardGameGeek and reddit forums dedicated to board games. Of the 115 participants that started the experiment, 71 finished at least one game and answered the survey questions related to that game. Among participants that provided the information, 42 self-identified as gamers, and only 5 did not. Relatedly, 12 participants stated that they play board games sometimes, and 33 that they play board games often, but only 1 participant stated that they rarely play board games, and none of the participants answered that they never play games. Experience with One Night Ultimate Werewolf in particular was more spread out, but our participant pool still leaned towards those that had played the game. Only 5 participants had never played the game before, and 11 had played between 1 and 10 times, while 14 stated that they had played 10 – 50 games before, and 16 played more than 50 times before participating in the study. However, because people that are more familiar with games can provide more detailed insights, this biased sample is not a limitation of the study.

The data was analyzed in several dimensions: Quantitatively, for each of the six survey questions, the results for games played with the balanced agents was compared with the results
for the capricious and the fanatical agents, and the results for the fanatical agents was compared
to the results for the capricious agents as well. This comparison was done using a Mann-Whitney-
Wilcoxon test to test for a difference in means [Man47], and a χ² test to test a difference
in distribution [Tal87]. To account for multiple testing, the Holm-Bonferroni correction was
used [Hol79]. Additionally, the same tests were run on subsets of the data corresponding to
factors that were found to have an effect, such as gaming experience. Qualitatively, the free
form comments provided by the participants were analyzed, and assigned to several groups.
Additionally, individual game logs were used to analyze which statement types and situations
were preferred by players. In the next section I will provide an overview over the most important
insights from the results of this study. This is then followed by a section with more detailed
information, including graphs relevant survey question. Finally, I will discuss the feedback that
was obtained from the free form comments and game logs.

6.3.1 Summary

As described above, the six questions asked in the post-game survey consist of four questions on
a 5 point Likert scale, and two questions in which the participants were presented with actual
statements made during game play and asked whether they believed those statements, or if the
statements made strategic sense to them. For the latter two questions, I used a count of how
many options were selected by the participant as the response variable, but I will also discuss
below what the distribution of selected statements looks like. Figure 6.3 shows a summary of
the responses for each combination of the six questions and three treatments. Graphs for the
responses to the individual questions are provided in section B.2 in the appendices. In addition,
a select few graphs will be presented below.

For none of the questions the difference in the mean was statistically significant between
treatments, but for several questions the difference in distribution was statistically significant.
Concretely, for the questions regarding which statements made by the agents made sense to the
players, the difference between the balanced agent and the fanatical agent was different with
p = 0.0022 (χ² = 29.23), and the difference between the balanced agent and the capricious agent
was also found to be different with p = 0.043 (χ² = 21.84), after the Holm-Bonferroni correction.
Additionally, the number of statements the players believed was significantly different between the
fanatical and the balanced agents with p = 0.00003 (χ² = 39.3), in distribution. The difference
in perceived skill of the agents was only weakly statistically significant between the balanced and
fanatical agents (p = 0.051, χ² = 15.722), as well as between the balanced and the capricious
agents (p = 0.09, χ² = 14.27). While these results do not fall under the chosen significance
threshold of 0.05 and may require additional testing to be confirmed or refuted, they constitute
interesting scenarios to analyze in more detail. In particular, while the Holm-Bonferroni method
Figure 6.3 A summary of the responses to the six survey questions for each of the three treatments. Each question is identified by a descriptor, where the last letter indicates whether the game was played against (f)anatical, (b)alanced or (c)apricious agents.
helps eliminating type I errors, if there is no negative dependency between the responses it may lead to type II errors, especially with such small numerical differences to significance.

6.3.2 Detailed Results

In this section I will present individual graphs and interpretations of the significant differences described above. Each participant was presented with a list of statements made by the AI agent that started as the Werewolf, or the Seer, if the participant started as the Werewolf, and was asked which of the statements made strategic sense, regardless of whether the participant believed it or not. Figure 6.4 shows the distribution of how many statements each participant rated as making sense, for each of the three agent types they played against. The difference in the distribution of these counts between the balanced agent and the capricious is statistically significant with $p = 0.002707$ ($\chi^2 = 21.84$), and between the balanced agent and the fanatical agent with $p = 0.000131$ ($\chi^2 = 29.23$), before taking the Holm-Bonferroni correction into account. As the graphs show, there are two key differences: Firstly, the number of games in which the participants rated no statements to make strategic sense is significantly lower for the balanced agent than for the other two agent types. Secondly, the number of games in which all statements made by the agent was rated as being strategically sound is higher for the balanced agent than for the other two agent types. Notably, though, the fanatical agent has a number of games in which all of its statements made sense to the participant, but very few in which only five or six did. This makes sense because new information that would require a change of plans is more likely to be revealed early, to give the players time to discuss it. However, if the initial plan formed by the fanatical agent is not affected by this new information, the entirety of it is likely to make sense. Interestingly, the capricious agent, which just uses whatever plan is best during any turn, does not do well at generating many statements that were rated as making sense over the course of the game. This provides additional support for the thesis that humans expect the agents to commit to their intentions, rather than changing their story every turn.

So far, the results I presented only took into account how many statements the participants rated as making sense, but not which. Figure 6.5 shows how many players rated each statement, from turn 1 to turn 7, to make sense. Note that the actual statements were not identical across games, because they depended on the actions performed during the night and what each other player said. Because all agents start identically, without an existing plan, and little information given to them, the first step of each plan was generally considered to make strategic sense by more participants than any following steps. However, as the game progresses, and especially towards the end of the game, participants found the statements made by the capricious, as well as the fanatical agent, to make less sense than the statements by the balanced agent. This is another indicator that it is advantageous to strike a balance between sticking to one’s story, and
Figure 6.4 Distributions for the number of statements made by an AI agent that participants rated as making sense.
adapting to new information. Since the fanatical agent would never change its plan, and therefore not take into account anything the other agents said, its story would become less believable over time. However, the capricious agent, which performs whichever action starts the best plan for the current turn, without even considering a continuation of its existing plan, creates stories that may no longer be consistent with the plan the agent came up before.

In addition to asking which statements made sense to the participants, they were also asked which statements they believed. Figure 6.6 shows the distribution for how many statements the participants claimed to have believed for each of the three agent types. For the fanatical agent, this distribution is particularly interesting, in that most of the participants believed either all or none of the statements. For the balanced agent, on the other hand, the ratings were more spread out. As with statements that were rated as making strategic sense, this can be explained by the fact that if the fanatical agent comes up with a plan that is not influenced by new information,
Figure 6.6 Distributions for the number of statements made by an AI agent that participants stated that they believed.

As mentioned above, the difference in how participants rated the skill of the AI at the game was only weakly statistically significant, but the responses still give additional insight into players’ preferences. Figure 6.7 shows the participants’ responses to the question of whether the AI agents played well, on a 5 point Likert scale. For any agent type, there is a large contingent of players that rated the agents as playing very badly, and I will go into more detail about that below, but for now I want to focus on the other participants. The most noticeable difference is that players either rated the fanatical agents as playing poorly or as playing well, whereas for balanced agents the ratings are almost the opposite, with more participants rating them neutral, rather than good or bad. This result is consistent with the insights presented above, where there are several
Figure 6.7 Player ratings for how well the different AI agents played, as ranked on a 5 point Likert scale.

games in which the plan the fanatical agent comes up stays believable and sensical over the entire course of the game, and the commitment they have to the plan makes them very convincing. However, when this process fails, the fanatical agent performs poorly. On the other end of the spectrum, the capricious agent avoids committing to a plan that might become invalidated, but the unstable nature of its behavior often leads to scenarios where it changes plans too quickly to be believable. However, when the influx of information is very rapid, this flexibility is an asset, and it is in such games that the capricious agent is perceived as doing well. In a way, the balanced agent is an attempt to get the best of both worlds, and provide more stable behavior, but does so at the expense of not excelling as often. To provide further insight into what the participants perceived to be missing and characterize where this work can go in the future, the next section will provide a summary of the free form text responses the participants provided.
6.3.3 Free Form Prompt

After each game, in addition to answering the survey questions, participants were also given the option to provide textual feedback about the AI agents' performance. Because of the comparatively higher effort of providing free form text input than simply answering Likert scale questions, only between 21 and 26 participants per agent type provided a response. Three central themes emerged from the comments:

- Participants expressed confusion about how the AI agents determined who to vote for. For example, one participant wrote *no clue why they votes me besides human discrimination*. Overall, 7 participants noted some variation of this for the balanced agents, 9 for the capricious, and only 1 participant did for the fanatical agents. This is indicative of two things. First, the AI agents perform their votes based purely on which worlds they consider possible, and with which weights. Since these weight updates are not transparent to the player, it sometimes results in surprising results. Second, what surprised participants was often that the agents did not just flat-out believe them, especially since they were often telling the truth, but the agents still considered that the human player might have been lying.

- Participants were often frustrated that the agents did not respond to their questions. 4 participants noted this for the balanced agent, 3 for the capricious agent, and 8 did for the fanatical agent, including *They do not respond to questions well if at all. It would be more fun to play with them if it felt like they could react to what I asked.*. This result is not entirely unexpected, since the agents could choose to answer questions if that was part of a plan that they found to be beneficial, but were not required to do so, or do so immediately. In particular, the higher number of comments for the fanatical agent that noted the lack of responsiveness is expected and consistent with the other results.

- Several participants noted a lack of explicit responses to other agents' actions, such as one participant stating, *The AI seems to be mostly talking to itself and doesn't try to deduce any new information from the events that occurred. It's a bit like a FSM or a modal logic agent that forgot to move from their starting state and is just digging into the same tracks*, after they played with the fanatical agent. Among all participants, 4 had a comment related to this for the balanced agent, 5 for the capricious agent, and 5 for the fanatical agent. The participants particularly noted how agents did not seem to care when two stories conflicted, because they did not explicitly call out such discrepancies. Since the focus of this work was in commitment to the agent's own plan, this decision was made deliberately, but I believe integrating more interactivity is of great importance for future work.
Apart from these three reoccurring themes, a few participants also noted that the lack of body language constituted a limitation for their enjoyment of the game. Several comments also noted improvements in agent behavior compared to previous games the participant played, some of the specifically mentioning how agents started to change their story. Despite the relatively low number of participants that provided free form comments, they provide some additional supporting evidence for the importance of agent commitment on the players’ perception of the agents, and also guidance for future work. Most of the players that provided comments also gave permission to release them publicly, and these responses can be found in section B.3 in the appendices. In the next section I will conclude with a discussion of insights that were gained from this study, and what areas for improvement there are.

### 6.3.4 Discussion

While the results presented above provide some evidence that the level of commitment matters to how humans perceive the agents to play One Night Ultimate Werewolf, they also revealed that there are several confounding factors that provide opportunities for future work, or simply influenced the results unduly. Additionally, inspecting the actual game logs also allowed me to gain additional insight into why the participants answered the way they did. One influencing factor that became immediately clear from the game logs was the limitations imposed by asking the participants which statements they believed. In the scenario where the player starts as the Seer, Carol, who starts as the Werewolf, often starts by coming up with a plan to convince the other players that she started as the Seer. This plan makes strategic sense, and was typically rated as such by the human players, but the human player also knows for a fact that Carol did not start as the Seer, and therefore they answered that they did not believe her statements. While there might be better choices for the AI agent, such as waiting which roles remain unclaimed, the plan they chose is not unreasonable, either. Furthermore, since waiting often leads to situations in which no one wants to be the first to say something, which is especially unfortunate in my variant of the game that is limited to seven rounds, encouraging the AI agents to make statements was a deliberate design decision.

Another limitation of how the AI agents appeared to the human players is with the repetition of statements. While this is quite common in actual game play, especially for players with roles such as the Villager, the turn-based structure of the game and the fixed sentence constructs made this more obvious to the players. In actual game play, a player could use a sentence like I really am the Villager, to reinforce a previous statement, whereas my implementation used the same surface realization for both occurrences. One way to address this could be to add more sophisticated sentence generation that would make the statements less repetitive, but further experiments are needed to determine whether this would change humans’ perception.
Figure 6.8 Player ratings for how much fun it was to play with the different AI agents, among players with moderate or high board game experience.

A few participants also provided elaborate explanations for why the AI agents were “wrong”, based on the premise that they should have believed what the player was saying. From the player’s perspective, this made sense, because they were indeed telling the truth, but the AI agents had no way of knowing that. One particular participant wrote that the game was an *easily solveable game*, when that was predicated on the agents assuming that the participant told the truth. While it would be interesting if player experience with games in general or One Night Ultimate Werewolf in particular has an influence on player perception, many of our participants did not provide any demographic information. Figure 6.8 shows how players that stated to have moderate to high board game experience rated the fun of playing with the different agents. While the graph seems to indicate that the participants prefer the more reactive capricious agents, the sample size is too small to draw any definitive conclusions, and further investigation would be needed.
The work I presented in this document investigates how AI agents can play games that involve communication with human players, focusing on the agents’ commitment to their plans. In this chapter I will first summarize my findings to reiterate the key contributions my work made to the field, and then close with an outlook on some of the directions future work could go.

7.1 Summary

The main point to take away from my work is that agent intentions and commitment matter. To reach this conclusion, I first presented hand-crafted, goal-directed AI agents for the cooperative card game Hanabi, that directly compared to prior agents that had no concept of intentions [Ege17d]. When playing with human players, the goal-based agents performed significantly better, providing the starting point for a more generalized agent design, that I will summarize in the next section.

7.1.1 Intentional Agents for Doxastic Games

While the Hanabi playing agent performed very well with human players, the definition and implementation were specific to the game, and could not be reused for other application domains easily. Therefore, I presented a more general agent design, that works on the basis of how well goals can be achieved, and uses this quality measure to determine agent commitment. In
games involving communication, and therefore reasoning about what other agents know, in particular, the quality of a belief can be measured as how likely an agent thinks a certain fact is. A possible worlds model lends itself naturally to such a measure, because each agent will consider a number of worlds possible, in some of which a fact may hold while in others it does not, and the ratio of worlds in which a fact holds is an indicator for how likely the agent considers it. This quality measure can be developed further by assigning weights to each world, making some to be considered less likely to begin with, and therefore the facts that hold in them will also be considered less likely by the agent. This weighted quality measure of beliefs can then be used by the agent to assess how well a plan reaches a goal. Once an agent adopts a plan to reach a goal with the associated expected quality, they can decide when to change their commitment by comparing how well a new candidate plan would reach its goal to how well the currently held plan is expected to reach the current goal.

The numerical nature of the weighted quality of a goal that is expected to be reached from a plan directly leads to a straightforward way to control how committed an agent should be to a goal, by introducing an additional numerical factor into the comparison. If an agent has a plan, which it expects to reach a goal with weighted quality $w$, and subsequently comes up with a new plan that reaches a goal with expected weighted quality $w'$, the agent will switch to that new plan if and only if

$$w' \geq \alpha \cdot w$$

By setting $\alpha$ to 0, the agent will change to a new plan regardless of its quality, called a capricious agent, whereas $\alpha = \infty$ would cause the agent to disregard the new plan in all cases, and keep pursuing whatever plan it already has, as a fanatical agent. Values of $\alpha$ between these two extremes, especially values close to 1, result in agents that achieve a balance between staying committed to a plan and changing to a better plan when one is found. Accordingly, I call the result balanced agents.

### 7.1.2 Ostari

The actual implementation of balanced agents is based on a framework, called Ostari [Ege17c], which provides a language that includes belief manipulation operations based on Dynamic Epistemic Logic. The framework provides several execution and querying capabilities to use such actions, as well as a mode specifically for game play. The agent design presented above, and discussed in more detail in section 4.2, is implemented in Ostari by means of a planning process that uses a weighted quality of beliefs as the guiding heuristic, and as a measure for how well goals can actually be achieved. When Ostari is directed to play a game, each agent will perform this planning process on each of their turns, and then decide whether to keep their current plan
or switch to a new one as described above.

While the main focus of my work was the game One Night Ultimate Werewolf, Ostari, as a framework, is not specific to any particular game. One of the first test cases for Ostari was a joke about three logicians in a bar, where the bar keeper asks the logicians “Do all of you want a beer?”, and the logicians answer, in order, with “I don’t know”, “I don’t know” and “yes”. To replicate the reasoning of the logicians in the joke, Ostari’s capabilities for reasoning about agent beliefs and their beliefs about other agents’ beliefs is used. I also presented how Ostari can be used in the process of generating a short detective story, in which the asymmetry in knowledge between the detective and the murderer plays an important role [Ege17b], a concept which was then expanded upon to generate a detective game, in which a player collects evidence to catch a murderer [Moh18]. These applications are meant to showcase how Ostari can be used beyond One Night Ultimate Werewolf.

7.1.3 One Night Ultimate Werewolf

The main application domain for my agents presented in this document is the game of One Night Ultimate Werewolf, in which each player is secretly assigned a role from one of two factions: The Citizens or the Werewolves. Some roles provide the players with information about other players’ cards, while others allow them to switch players’ role cards around, which also changes the players’ faction association, often without their knowledge. After each player has performed their special action, the game moves to a discussion phase in which players want to deduce which special actions were actually performed, and which players are on which team. At the end of the discussion phase, players vote who to eliminate, and then win depending on which player was eliminated. The main challenge in this game is that players are allowed to lie freely during the discussion phase, so no information is inherently trustworthy. However, by players corroborating each others’ stories, or identifying contradictory information, an agent can estimate how likely each piece of information is to be true. These estimates then inform the agent’s decision of which goal to pursue, and are taken into account when the agent comes up with a plan to achieve that goal. The control over the agent’s level of commitment described above provides a measure of how much the agent will stick to their story, versus how much they adapt to what the other players say.

To investigate the effect the level of commitment has on actual game play, I performed two experiments. In the first, agents with different levels of commitment played against each other in a variety of scenarios, ranging from game states that were intentionally designed to minimize the impact the level of commitment should have on the outcome, to scenarios that heavily favored one faction to observe how the agents behaved in these corner cases. The most important findings from this experiment are [Ege18]:
• In games in which level of commitment should - by design - have little to no impact, no statistically significant difference in performance was found across the different agent types.

• In typical games the level of commitment of the agents has an effect on win rate, resulting in a difference of slightly under 5% between the different levels of commitment.

• In games that heavily favor one faction, the Citizens, the Werewolf player can increase their win rate by up to 4% by being more suspicious of other player’s statements.

• The planning process the agents use performs in a degenerate way when very few states are expanded, but as soon as 25 or more states are expanded, the agents perform as desired.

With the insights gained from the computational experiment, I then extended the implementation by a user interface using Unity to allow human players to play against the agents in a web browser. In addition to the game as played by the agents, I also extended the available actions to include more interaction options for the human players, in the form of allowing them to ask the agents questions, rather than just stating facts about the world. The agent’s choice of which goal to adopt also takes into account if it would answer a question they were asked, but whether or not they actually do so is part of the agent’s decision making process for which goal to pursue. The second experiment I performed involved human participants playing against three different types of agents, fanatical, capricious and balanced, and answering questions about their experience. The most important results from this experiment are:

• While the average number of statements made by the AI agents that the participants rated as making strategic sense was no statistically significantly different between the different agent types, their distribution was. For the fanatical agent this typically meant that either all seven statements that the agent made were rated as making sense, or zero to one, while for the balanced agent participants stated more often that five or six statements made sense.

• Similarly, when investigating which statements made by the agents the participants rated as making strategic sense, the statements capricious and fanatical agents made later in the game were chosen less often, compared to the balanced agent.

• When asked how many statements the participants believed for each agent type, the fanatical agent was also believed more often in all cases, or none, as opposed to the capricious and balanced agents which had more games in which some of their statements were believed.

• Likely as a consequence of this behavior, when participants were asked to rate the skill of the AI agents, the rating given for the fanatical agents was either very bad or very good, whereas more people gave a neutral rating to the balanced agent.
In a free form text response field, several participants noted how some of the agents refused to answer questions that were answered, which was particularly noted for the fanatical agents, while a point repeatedly mentioned about the balanced and capricious agents was that it was not clear enough how they determined who they voted for.

In summary, fanatical agents can be perceived really well, but also have many fail cases, while capricious agents are generally perceived as being too erratic. Balanced agents are rarely perceived quite as well as the fanatical ones are, but also fail to provide a reasonable experience less often. Overall, as the difference in raw game performance determined in the computational study was less than a 5% change in win rate, it was expected that the difference between the agents when playing with humans would also be small. The fact that the participants rated the agents as behaving differently despite them only having these small difference provides additional evidence that agent commitment matters. The participants’ comments about the agents’ behavior also provides additional information about other factors that affected their experience. I will address how these comments could be incorporated into future work in the next section.

### 7.2 Future Work

In this section I want to address how my work can be improved upon in the future by addressing some of its limitations, but also what avenues of further research were opened up, and how these could be approached. I will start with one of the main points of criticism from the participants in the study involving using the agents for One Night Ultimate Werewolf: The limited interactivity of the agents.

#### 7.2.1 Interactivity

While this thesis presents evidence that the commit levels of intentional agents has an effect on human perception of the agents, it also revealed other factors that affect player perception and enjoyment. One major point that was noted by several participants was a perceived lack of interaction between the players. In a way, this was intentional, since the goal of my work was an investigation of commitment to a plan, rather than a dialog model. However, I believe in order to improve how the agents are perceived by players, it will be necessary to address this limitation. In order to improve the perceived interaction between the agents, they would need to react to each other’s statements more directly. For example, currently, when one player claims to be the Seer, while another player also has a plan to convince the other players that they are the Seer (perhaps because they actually are), they will never address the first player’s claim directly. Instead, the agent should adopt a plan that not only has a goal to convince other players of their own role, but also to discredit the competing player’s claim. On a basic level, this could be as
simple as addressing that player directly, to convey the conflict more directly. A more advanced way would be to adopt a goal to find other players to corroborate one owns story. In the case where two players claim to be the Seer, stating what was seen can help gain the trust of the player whose card was seen.

Because the agents’ main task was coming up with plans to convince others of their faction affiliation, they were also not inquisitive in the sense of asking other players questions, and the onus of trying to get players to state specific facts fell on the human player. Future iterations of the agents could focus more on knowledge seeking behavior to better inform their own plans. Due to the limited number of turns, the decision of how much to ask, and when to actually make claims to convince other players will be important to balance. I believe this balance could be modeled using my current approach by adding goals that refer to knowledge that should be gathered, and comparing the expected quality of obtaining this knowledge and the agent making claims about themselves. However, additional experimentation will be needed to determine the details of this balance.

Another aspect that can be improved in future work is the voting pattern of the AI agents. The agents presented in this document all use the same voting strategy, to make the results between different agents comparable, and draw conclusions about their commitment levels more directly. However, as noted, human players do not always understand why the AI agents vote the way they do. Future work could improve this by integrating means to discuss how the players claim that they will vote, and also improve the voting itself to account for strategies like attempting to split the vote between two players, and ensuring that Werewolves do not vote for each other inadvertently.

Finally, while the setup I used for my studies pits one human player against multiple AI agents, an interesting variation of the experiment for future work would be to have a more even mix of human players and AI agents, and ask the participants to guess which other player was human and which was controlled by an AI agent, similar to a Turing Test [Tur50]. This setup would help determine which perceived behaviors are artifacts of the game encoding, and which are actual traits of the agents.

7.2.2 Goal Derivation

As presented, my agents still need to be provided with domain expert knowledge in the form of the goals they should consider pursuing. While these goals can be very generic, and the agents can determine how to reach them, this is still an area that introduces bias into how the agents approach the game. For example, in One Night Ultimate Werewolf, the goal for the Werewolf is to convince as many other players as possible that they are not the Werewolf. However, perhaps a more appropriate goal would be to only convince two other players, which would be enough
for a majority, but to achieve that goal with more certainty. Ideally, though, no domain expert knowledge would be required at all. It would be interesting to use the formalization of the game in Ostari, and derive optimal goals, or at least a measure to compare how well suited different sets of goals are. Alternatively, since Ostari provides a forward simulation model, a machine learning algorithm could be used to find goals using an appropriate encoding of the state of possible goals. Genetic algorithms, for example, have been used to learn logical formulas for classification tasks before [Aug95]. Rather than using classification accuracy or precision as the objective function, the fitness of each individual in the population would be measured in terms of win rate over a large number of games played in Ostari, which requires significant computational resources.

### 7.2.3 Epistemic States

One key difference between my Hanabi agents and the agents that played One Night Ultimate Werewolf is that the latter are based on a framework that is general enough that it can also be used for other applications. However, the space of games that can be modeled this way is still limited, because the current representation of epistemic states requires enumerating all possible worlds, the number of which is typically exponential, or at least factorial, in some value that is determined by the game, and thus prohibitively large. The Hanabi agents, in contrast, use a data structure for their beliefs that is specific to the game, but also much more compact. It would be convenient to devise an epistemic state representation that minimizes the size for common cases\(^1\). Such a representation would only need to expose ways to interpret formulas and flip truth values. Especially when the accessibility relation is constrained to be e.g. an equivalence relation, it is possible to devise more efficient encodings of the possible worlds [Lew13], which could be enough for many applications. The exact requirements for such a data structure, and the exact specifications are a topic for extensive future work.

### 7.2.4 Application Domains

While I presented how my agents play One Night Ultimate Werewolf, the underlying mechanism and framework could be adapted to other social deduction games. In The Resistance [Esk09], for example, two teams, the Resistance and the Government Spies, play against each other. The Government Spies form the smaller faction, with around 30% of players belonging to it, but they know who the other spies are, while the Resistance members do not get any information about other players' roles. Game play consist of passing a mission leader token around, with the mission leader selecting a subset of players to go on a mission with. All players get to vote whether the selected subset actually gets to go on the mission, and if that vote passes, the selected players

\(^1\)This is likely impossible in the general case.
can (secretly) choose to sabotage the mission, which causes it to fail. The goal of the Resistance is to pass three missions, while the Government Spies are trying to get three missions to fail. In contrast to One Night Ultimate Werewolf, the roles are assigned statically and can not change over the course of the game. Additionally, most game play actions, such as which player voted to send a particular team on a mission, is public and can provide information to the players. However, since the players on the Resistance team have no information about the other players they may inadvertently vote in favor of the Government Spies. The game actions of this game could be encoded in Ostari in a similar fashion as I did for One Night Ultimate Werewolf, but the agents would need to be provided with the appropriate goals. Additionally, because discussion is interleaved with other game play actions, which reveal additional information, long-term commitment may be less important than reacting to the knowledge that was obtained. I believe that Ostari’s capabilities to represent beliefs and the (weighted) quality of belief would be an asset for agents to play this game, though. Other social deduction games, such as Battlestar Galactica: The Boardgame [Kon08] add even more classical board game elements, which can be modeled as simple non-epistemic actions with public effects (if applicable). This likely increases the complexity of the planning problem, as well as the size of the epistemic state of the game, though, and would therefore benefit from the state representation optimizations discussed above.

Beyond recreational games, my approach of using a theory of mind and intentionality to communicate with humans could also be used in a wide range of other applications, such as educational games that teach programming or logic, or intelligent tutoring systems. Typically, such applications teach concepts in the form of puzzles that the students/players have to solve and the player has the option of obtaining hints from the system, which are usually obtained from how other players solved the same level or puzzle the player is stuck on [Bar08]. This is a challenge when not enough previous data is available to generate hints. Valls-Vargas et al., for example, describe a procedural content generation system based on graph grammars for a game that teaches students about parallel and concurrent programming [VV17]. Since the puzzles are procedurally generated according to what a particular student should learn, no data from other players is available. However, since the system also has a model of what they want the player to learn, this can serve as the basis for an intention that a hint-giving agent can work towards. The goal in this application is to give enough information to the player for them to be able to solve the puzzle, without revealing so much that they don’t learn anything from it.

### 7.2.5 Communication Modes

One theme I cover in my work is that of communication, as defined by the rules of the games I investigated. However, even when following these restrictions, humans exhibit subtle behavior that affects how the communication is perceived. For example, in Hanabi, a hint that is given
with hesitation is perceived differently by other humans than one that is given instantaneously. While hesitation and/or rushed decisions are discussed extensively among professional poker players \(^2\), they remain understudied in the field of game AI. Interestingly, one of the nominees for the German 2018 Spiel des Jahres (Game of the Year) award is the game The Mind [War18], in which players are dealt a subset of the cards numbered from 1 to 100, and have to collaboratively play them in ascending order, without communicating about their cards explicitly. Game play for this game relies almost entirely on timing and hesitation. While playing this game between AI agents is trivial, since they have perfect timing, it would be interesting to investigate how to play it with human players, by using non-verbal communication modes.

Another communication channel with great importance for humans is gaze [Lan01]. Using eye tracking hardware that is becoming more easily available it has already been noted that a human user’s intentions and mental model have an effect on their gaze, and can therefore be utilized to improve computer-human interaction [Bad11]. In Hanabi, for example, gaze can indicate decision making or intentions for what the player would like the agent to do. I mentored an undergraduate student that implemented a version of Hanabi that includes eye tracking [Got18] as a summer project. For the future, I plan to extend this implementation with the AI agents described in chapter 3, and perform experiments to determine if there is a correlation between the player’s gaze and their intentions in the game, and if that is the case, how to use this correlation to improve the agents.

---

\(^2\)See, for example, this poker strategy blog post: https://upswingpoker.com/poker-tells-live-common-list/
BIBLIOGRAPHY


Suber, P. “Open access overview” (2007).


APPENDICES
A.1 Ostari Input File

Types:
Indices = a, b, c, d, e, center1, center2, center3, tmp, game
Cards = Werewolf1, Werewolf2, Troublemaker, Villager1, Villager2, Insomniac,
    Seer, Robber
Roles = Werewolf, Troublemaker, Villager, Insomniac, Seer, Robber
GoodRole = Troublemaker, Villager, Insomniac, Seer
Players = a, b, c, d, e
Locations = a, b, c, d, e, center1, center2, tmp
PlayLocations = a, b, c, d, e, center1, center2
Centers = center1, center2
Phases = Init, WWPhase, TMPhase, SPhase, IPhase, RPhase, DiscussionPhase,
    VotingPhase, DeathPhase, Done
Constants = game, tmp
Truth = honest, lying
Count = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Bool = True, False
Vars = p, x, y, p1, p2, r, s, s1, v, w, self
Properties:
succ :: Count -> Count (0: 1; 1: 2; 2: 3; 3: 4; 4: 5; 5: 6; 6: 7; 7: 8; 8: 9)
max :: Constants -> Count ()
maxat :: Constants -> Locations ()
votes :: Players -> Count ()
voted :: Players -> Players ()
tmpv :: Constants -> Count ()
maxvotes :: Constants -> Count ()
dead :: Players -> Bool ()
role :: Locations -> Roles ()
initrole :: Locations -> Roles ()
card :: Locations -> Cards ()
left :: Players -> Players (a: e; b: a; c: b; d: c; e: d)
right :: Players -> Players (a: b; b: c; c: d; d: e; e: a)
roleassignment :: Cards -> Roles (Werewolf1: Werewolf; Werewolf2: Werewolf;
    Insomniac: Insomniac; Seer: Seer; Troublemaker: Troublemaker; Villager1:
    Villager; Villager2: Villager; Robber: Robber)
phase :: Constants -> Phases ()
votefor :: Players -> Players ()
eq :: Players -> Players (a: a; b: b; c: c; d: d; e: e)
eqt :: Truth -> Truth (honest: honest; lying: lying)
next :: Phase -> Phase (Init: WWPhase; WWPhase: SPhase; SPhase: TMPhase;
    TMPhase: RPhase; RPhase: IPhase; IPhase: DiscussionPhase; DiscussionPhase:
    VotingPhase; VotingPhase: DeathPhase; DeathPhase: Done)
seen :: Players -> Players ()
seenRole :: Players -> Roles ()
swapped :: Players -> Bool ()
Actions:
assignCard(p: Players, c(p): Cards)
{
    precondition Forall x in Players: (card(x) != c);
    card(p) = c
}
assignRole(p: Players)
{
    public (Players) role(p) = roleassignment(card(p));
    public (Players) initrole(p) = roleassignment(card(p));
    learn (p): Which r in Roles: role(p) == r;
    public (Players) seen(p) = p
}
causeTrouble(p(p): Players, t(p): Truth)
{
    precondition initrole(p) == Troublemaker;
    if (eqt(t) == honest)
    {
        role(tmp) = role(left(p));
        role(left(p)) = role(right(p));
        role(right(p)) = role(tmp);
    }
swapped(p) = True
}
else
{
    swapped(p) = False
}
}
rob(p(p): Players, p1(p): Players)
{
    precondition initrole(p) == Robber;
    if (eq(p) != p1)
    {
        role(tmp) = role(p);
        role(p) = role(p1);
        role(p1) = role(tmp);
        learn(p): Which r in Roles: role(p) == r
        robbed(p) = p1;
    }
    else
    {
        robbed(p) = p
    }
}
insomnia(p: Players)
{
    precondition initrole(p) == Insomniac;
    learn(p): Which r in Roles: role(p) == r
}
see(p(p): Players, p1(p): PlayLocations)
{
    precondition initrole(p) == Seer;
    learn(p): Which r in Roles: role(p1) == r;
    seen(p) = p1;
    seenRole(p) = role(p1)
}
recognizeWerewolves(p: Players)
{
    precondition initrole(p) == Werewolf;
    tell(p): Each p1 in Players: role(p1) == Werewolf
}
advancePhase()
{
    phase(tmp) = next(phase(game));
phase(game) = phase(tmp)

claimInitRole(p: Players, r: Roles, r1(p): Roles)
{
    precondition initrole(p) == r1;
    if (initrole(p) != r)
    {
        addWeight(1)
    }
    else
    {
    }
}

claimRole(p: Players, r: Roles, r1(p): Roles)
{
    precondition role(p) == r1;
    if (role(p) != r)
    {
        addWeight(1)
    }
    else
    {
    }
}

claimSeen(p: Players, s: Players, s1(p): Players)
{
    precondition seen(p) == s1;
    if (seen(p) != s)
    {
        addWeight(1)
    }
    else
    {
    }
}

claimSeenRole(p: Players, r: Roles, r1(p): Roles)
{
    precondition seenRole(p) == r1;
}
if (seenRole(p) != r)
{
    addWeight(1)
}
else
{

}

claimSwapped(p: Players, s: Bool, t(p): Bool)
{
    precondition swapped(p) == t;
    if (swapped(p) != s)
    {
        addWeight(1)
    }
    else
    {
    }
}

claimRobbed(p: Players, p1: Players, t(p): Players)
{
    precondition robbed(p) == t;
    if (robbed(p) != p1)
    {
        addWeight(1)
    }
    else
    {
    }
}

voteCertain(p: Players, v: Players)
{
    precondition B (p): role(v) == Werewolf;
    precondition eq(p) != v;
    tmpv(game) = votes(v);
    votes(v) = succ(tmpv(game))
}

vote(p: Players, v: Players)
vote0 \( p: \text{Players}, \ v: \text{Players} \) 
\{
 precondition \( \forall p_1 \in \text{Players}: (eq(p_1) == p) \text{ or } (W [<0.9] (p): \text{role}(p_1) == \text{Werewolf}) \);
 precondition \( W [>0.8] (p): \text{role}(v) == \text{Werewolf}; \)
 precondition \( \text{eq}(p) \neq v; \)
\( tmpv(\text{game}) = votes(v); \)
\( votes(v) = \text{succ}(tmpv(\text{game})) \)
\}
vote1 \( p: \text{Players}, \ v: \text{Players} \) 
\{
 precondition \( \forall p_1 \in \text{Players}: (eq(p_1) == p) \text{ or } (W [<0.8] (p): \text{role}(p_1) == \text{Werewolf}) \);
 precondition \( W [>0.7] (p): \text{role}(v) == \text{Werewolf}; \)
 precondition \( \text{eq}(p) \neq v; \)
\( tmpv(\text{game}) = votes(v); \)
\( votes(v) = \text{succ}(tmpv(\text{game})) \)
\}
vote2 \( p: \text{Players}, \ v: \text{Players} \) 
\{
 precondition \( \forall p_1 \in \text{Players}: (eq(p_1) == p) \text{ or } (W [<0.7] (p): \text{role}(p_1) == \text{Werewolf}) \);
 precondition \( W [>0.5] (p): \text{role}(v) == \text{Werewolf}; \)
 precondition \( \text{eq}(p) \neq v; \)
\( tmpv(\text{game}) = votes(v); \)
\( votes(v) = \text{succ}(tmpv(\text{game})) \)
\}
vote3 \( p: \text{Players}, \ v: \text{Players} \) 
\{
 precondition \( \forall p_1 \in \text{Players}: (eq(p_1) == p) \text{ or } (W [<0.5] (p): \text{role}(p_1) == \text{Werewolf}) \);
 precondition \( W [>0.3] (p): \text{role}(v) == \text{Werewolf}; \)
 precondition \( \text{eq}(p) \neq v; \)
\( tmpv(\text{game}) = votes(v); \)
\( votes(v) = \text{succ}(tmpv(\text{game})) \)
\}
voteRandom \( p: \text{Players}, \ v: \text{Players} \) 
\{
 precondition \( \forall p_1 \in \text{Players}: (eq(p_1) == p) \text{ or } (W [<0.3] (p): \text{role}(p_1) == \text{Werewolf}) \);
 precondition \( \text{eq}(p) \neq v; \)
\( tmpv(\text{game}) = votes(v); \)
\( votes(v) = \text{succ}(tmpv(\text{game})) \)
\}
== Werewolf);
precondition eq(p) != v;
tmpv(game) = votes(v);
votes(v) = succ(tmpv(game))
}
die(p: Players)
{
    precondition (votes(p) == 3) or (votes(p) == 4) or (votes(p) == 5);
dead(p) = True
}
die2(p: Players)
{
    precondition ((votes(p) == 2) and (Forall p1 in Players: ((votes(p1) != 3)
                        and (votes(p1) != 4) and (votes(p1) != 5))));
dead(p) = True
}
tally()
{
    public (Players) phase(game) = DeathPhase
}
finish()
{
    public (Players) phase(game) = Done
}
showCards()
{
    print(WerewolfCards)
}
showResult()
{
    print(Result)
}
Initial:
phase(game) = Init()
votes(a) = 0()
votes(b) = 0()
votes(c) = 0()
votes(d) = 0()
votes(e) = 0()
tmpv(game) = 0()
max(game) = 0()
maxat(game) = tmp()
swapped(a) = False()
swapped(b) = False()
swapped(c) = False()
swapped(d) = False()
swapped(e) = False()
looks like (a,b,c, d, e): Initial

Execute:

assignCard(a, $rolea)
assignRole(a)
assignCard(b, $roleb)
assignRole(b)
assignCard(c, $rolec)
assignRole(c)
assignCard(d, $roled)
assignRole(d)
assignCard(e, $rolee)
assignRole(e)

advancePhase()


showCards()
query: Which p in Players: role(p) == Werewolf
showResult()
query: Each p in Players: dead(p) == True

Game:
phase: WWPhase
1 turns
actions: recognizeWerewolves (mandatory)
next: advancePhase()

phase: TMPhase
1 turns
actions: causeTrouble (mandatory)
next: advancePhase()

phase: RPhase
1 turns
actions: rob (mandatory)
next: advancePhase()
phase: SPhase
1 turns
actions: see (optional)
next: advancePhase()

phase: IPhase
1 turns
actions: insomnia (mandatory)
next: advancePhase()

phase: DiscussionPhase
7 turns
actions: claimRole, claimInitRole, claimRobbed, claimSeen, claimSeenRole,
         claimSwapped (optional)
next: advancePhase()

phase: VotingPhase
1 turns
actions: voteCertain, vote, vote0, vote1, vote2, vote3, voteRandom (mandatory)
next: tally()

phase: DeathPhase
1 turns
actions: die (mandatory)
next: finish()

Goals:
(W [＞$cert] (self): role (self) == Werewolf) and (phase (game) == DiscussionPhase
  ) : Forall p in Players: ((eq (p) == self) or (seen (p) == self) or W [＞0.7] (p
  ) : (role (self) == Villager));
(W [＞$cert] (self): role (self) == Werewolf) and (phase (game) == DiscussionPhase
  ) : Forall p in Players: ((eq (p) == self) or (seen (p) == self) or W [＞0.7] (p
  ) : (role (self) == Seer));
(W [＞$cert] (self): role (self) == Werewolf) and (phase (game) == DiscussionPhase
  ) : Forall p in Players: ((eq (p) == self) or (seen (p) == self) or W [＞0.7] (p
  ) : (role (self) == Troublemaker));
(W [＞$cert] (self): role (self) == Werewolf) and (phase (game) == DiscussionPhase
  ) : Forall p in Players: ((eq (p) == self) or (seen (p) == self) or W [＞0.7] (p
  ) : (role (self) == Insomniac));
x = seen (self), y = seenRole (self) in (initrole (self) == Seer) and (phase (game)
   == DiscussionPhase) and (seen (self) != self) and (eq (self) != a) and (seen (self)
   != a): W [＞0.7] (a): (initrole (x) == y);
x = seen (self), y = seenRole (self) in (initrole (self) == Seer) and (phase (game)
   == DiscussionPhase) and (seen (self) != self) and (eq (self) != b) and (seen (self)
   != b): W [＞0.7] (b): (initrole (x) == y);
x = seen (self), y = seenRole (self) in (initrole (self) == Seer) and (phase (game)
== DiscussionPhase and (seen(self) != self) and (eq(self) != c) and (seen(self) != c): W [>0.7] (c): (initrole(x) == y);

x = seen(self), y = seenRole(self) in (initrole(self) == Seer) and (phase(game) == DiscussionPhase) and (seen(self) != self) and (eq(self) != d) and (seen(self) != d): W [>0.7] (d): (initrole(x) == y);

x = seen(self), y = seenRole(self) in (initrole(self) == Seer) and (phase(game) == DiscussionPhase) and (seen(self) != self) and (eq(self) != e) and (seen(self) != e): W [>0.7] (e): (initrole(x) == y);

r = role(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != a): W [>0.7] (a): (role(self) == r);

r = role(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != b): W [>0.7] (b): (role(self) == r);

r = role(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != c): W [>0.7] (c): (role(self) == r);

r = role(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != d): W [>0.7] (d): (role(self) == r);

r = role(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != e): W [>0.7] (e): (role(self) == r);

r = initrole(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != a): W [>0.7] (a): (initrole(self) == r);

r = initrole(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != b): W [>0.7] (b): (initrole(self) == r);

r = initrole(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != c): W [>0.7] (c): (initrole(self) == r);

r = initrole(self) in (W [> $cert] (self): (role(self) != Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != d): W [>0.7] (d): (initrole(self) == r);

x = random(Players) in (W [> $cert] (self): (role(self) != Werewolf)) and (W [>0.7] (self): (role(x) == Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != a): W [>0.7] (a): (role(x) == Werewolf);

x = random(Players) in (W [> $cert] (self): (role(self) != Werewolf)) and (W [>0.7] (self): (role(x) == Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != b): W [>0.7] (b): (role(x) == Werewolf);

x = random(Players) in (W [> $cert] (self): (role(self) != Werewolf)) and (W [>0.7] (self): (role(x) == Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != c): W [>0.7] (c): (role(x) == Werewolf);

x = random(Players) in (W [> $cert] (self): (role(self) != Werewolf)) and (W [>0.7] (self): (role(x) == Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != d): W [>0.7] (d): (role(x) == Werewolf);
x = random(Players) in (W [>$cert] (self): (role(self) != Werewolf)) and (W [>0.7] (self): (role(x) == Werewolf)) and (phase(game) == DiscussionPhase) and (eq(self) != e): W [>0.7] (e): (role(x) == Werewolf);

x = random(Players) in (initrole(self) == Seer) and (phase(game) == SPhase):
    seen(self) == x;

x = random(Players) in (initrole(self) == Robber) and (phase(game) == RPhase):
    robbed(self) == x;

y = random(Bool) in (initrole(self) == Troublemaker) and (phase(game) == TMPhase):
    swapped(self) == y;

Done.

Listing A.1 The Ostari input file for One Night Ultimate Werewolf. Note that the tokens starting with a dollar sign will be replaced by the actual agent type, role assignment and certainty threshold.
A.2 Intentional Agent Deliberation to Generate Next Move

\[
\text{intentionalMove} :: \text{GameState} \rightarrow \text{Strategy} \rightarrow \\
\quad \text{Commitment} \rightarrow \left[ \text{Move} \right] \rightarrow \text{State IntentionState Move}
\]

\[
\text{intentionalMove} \text{ gs strat Fanatical moves} = \text{do} \\
\quad (\text{intention, plan}) \leftarrow \text{get} \\
\quad \text{if null plan then do} \\
\quad \quad \text{let int} = \text{strat gs} \\
\quad \quad \text{let newplan} = \text{planMoves gs moves int} \\
\quad \quad \text{if null newplan then return } \left( \text{head moves} \right) \\
\quad \quad \text{else do} \\
\quad \quad \quad \text{put } (\text{int, tail newplan}) \\
\quad \quad \quad \text{return } \left( \text{head newplan} \right) \\
\quad \text{else do} \\
\quad \quad \text{let next} = \text{head plan} \\
\quad \quad \text{put } (\text{intention, tail plan}) \\
\quad \quad \text{return next}
\]

\[
\text{intentionalMove} \text{ gs strat Capricious moves} = \text{do} \\
\quad \text{let int} = \text{strat gs} \\
\quad \text{let newplan} = \text{planMoves gs moves int} \\
\quad \text{if null newplan then return } \left( \text{head moves} \right) \\
\quad \text{else do} \\
\quad \quad \text{put } (\text{int, tail newplan}) \\
\quad \quad \text{return } \left( \text{head newplan} \right)
\]

\[
\text{intentionalMove} \text{ gs strat Balanced moves} = \text{do} \\
\quad (\text{intention, plan}) \leftarrow \text{get} \\
\quad \text{if (null plan) || (reconsider gs plan) then do} \\
\quad \quad \text{let int} = \text{strat gs} \\
\quad \quad \text{let newplan} = \text{planMoves gs moves int} \\
\quad \quad \text{if null newplan then return } \left( \text{head moves} \right) \\
\quad \quad \text{else do} \\
\quad \quad \quad \text{put } (\text{int, tail newplan}) \\
\quad \quad \quad \text{return } \left( \text{head newplan} \right) \\
\quad \text{else do} \\
\quad \quad \text{let next} = \text{head plan} \\
\quad \quad \text{put } (\text{intention, tail plan}) \\
\quad \quad \text{return next}
\]

Listing A.2 Haskell code for the determination of the next move by an agent. Note that there are three variants: Fanatical, where the agent commits to a plan until it is done, Capricious, where the agent reconsiders its options every turn, and Balanced, where the decision on when to adopt a new intention is determined by the game state.
ONE NIGHT ULTIMATE WEREWOLF

B.1 Simulation Results
Table B.1 Simulation results for a balanced game, in which the Werewolf card cannot be exchanged for any other card. As expected does the level of commitment not have any significant effect on agent performance.

<table>
<thead>
<tr>
<th>Village</th>
<th>Werewolf</th>
<th>r</th>
<th>c</th>
<th>b(0.5)</th>
<th>b(0.66)</th>
<th>b(0.8)</th>
<th>b(1)</th>
<th>b(1.25)</th>
<th>b(1.66)</th>
<th>b(2.5)</th>
<th>b(5)</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td></td>
<td>89.73</td>
<td>89.95</td>
<td>89.6</td>
<td>90.18</td>
<td>90</td>
<td>89.48</td>
<td>89</td>
<td>89.55</td>
<td>89.38</td>
<td>88.43</td>
<td></td>
</tr>
<tr>
<td>capricious</td>
<td></td>
<td>0</td>
<td>48.78</td>
<td>49.65</td>
<td>50.1</td>
<td>50.13</td>
<td>50.63</td>
<td>49.78</td>
<td>49.88</td>
<td>49.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.5)</td>
<td></td>
<td>15.03</td>
<td>49.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.66)</td>
<td></td>
<td>14.7</td>
<td>49.38</td>
<td>50.33</td>
<td>49.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.8)</td>
<td></td>
<td>15.08</td>
<td>49.18</td>
<td>49.33</td>
<td>49</td>
<td>51.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(1)</td>
<td></td>
<td>15.78</td>
<td>51.75</td>
<td>50.63</td>
<td>49.85</td>
<td>49.38</td>
<td>48.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(1.25)</td>
<td></td>
<td>15.5</td>
<td>50.7</td>
<td>49.15</td>
<td>50.15</td>
<td>50.08</td>
<td>50.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(1.66)</td>
<td></td>
<td>15.15</td>
<td>49.33</td>
<td></td>
<td>49.03</td>
<td>50.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(2.5)</td>
<td></td>
<td>15.3</td>
<td>49.43</td>
<td>50.25</td>
<td></td>
<td>50.58</td>
<td>50.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(5)</td>
<td></td>
<td>15.43</td>
<td>49.58</td>
<td>51</td>
<td></td>
<td>50.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fanatical</td>
<td></td>
<td>14.75</td>
<td>51.23</td>
<td>50.3</td>
<td>48</td>
<td>49.35</td>
<td>49.18</td>
<td>51.13</td>
<td>49.6</td>
<td>49.45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table B.2 Simulation results for a balanced game, in which the Werewolf card can be exchanged for another card. In this case, the level of commitment of the agents does have an effect on agent performance.

<table>
<thead>
<tr>
<th>Village</th>
<th>Werewolf</th>
<th>r</th>
<th>c</th>
<th>b(0.1)</th>
<th>b(0.2)</th>
<th>b(0.5)</th>
<th>b(0.66)</th>
<th>b(1)</th>
<th>b(1.33)</th>
<th>b(4)</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td></td>
<td>55.6</td>
<td>56.3</td>
<td>53.8</td>
<td>54.2</td>
<td>53</td>
<td>57.2</td>
<td>54.7</td>
<td>55.2</td>
<td>40.1</td>
<td></td>
</tr>
<tr>
<td>capricious</td>
<td></td>
<td>27.4</td>
<td>48.9</td>
<td>51.3</td>
<td>50.1</td>
<td>48.7</td>
<td>46.9</td>
<td>52.1</td>
<td>49.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.1)</td>
<td></td>
<td>30.7</td>
<td>48.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.2)</td>
<td></td>
<td>26.7</td>
<td>51.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.5)</td>
<td></td>
<td>29.6</td>
<td>51.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(0.66)</td>
<td></td>
<td>24.8</td>
<td>49.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(1)</td>
<td></td>
<td>29.1</td>
<td>49.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(1.33)</td>
<td></td>
<td>27.1</td>
<td>51.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanced(4)</td>
<td></td>
<td>29.1</td>
<td>49.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fanatical</td>
<td></td>
<td>29.1</td>
<td>52</td>
<td>51.6</td>
<td>49.4</td>
<td>50.7</td>
<td>50.1</td>
<td>51.2</td>
<td>51.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table B.3 Simulation results for a game which heavily favors the non-Werewolf players. The level of commitment does have an effect on agent performance, because it is very likely that cards are switched around, making being overly committed to any particular plan a liability.

<table>
<thead>
<tr>
<th>Village</th>
<th>Werewolf balanced</th>
<th>Villagers balanced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>c</td>
</tr>
<tr>
<td>random</td>
<td>66.20</td>
<td>66.80</td>
</tr>
<tr>
<td>capricious</td>
<td>5.20</td>
<td>4.40</td>
</tr>
<tr>
<td>balanced(0.1)</td>
<td>6.20</td>
<td>3.00</td>
</tr>
<tr>
<td>balanced(0.2)</td>
<td>4.20</td>
<td>3.80</td>
</tr>
<tr>
<td>balanced(0.66)</td>
<td>1.80</td>
<td>3.20</td>
</tr>
<tr>
<td>balanced(2)</td>
<td>1.60</td>
<td>0.00</td>
</tr>
<tr>
<td>fanatical</td>
<td>1.20</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table B.4 Simulation results for a game which heavily favors the non-Werewolf players, comparing fanatical agents to balanced agents with $\alpha = 0.66$, with different certainty thresholds.

<table>
<thead>
<tr>
<th>Certainty</th>
<th>Werewolf balanced</th>
<th>Villagers balanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.6%</td>
<td>1.2%</td>
</tr>
<tr>
<td>0.3</td>
<td>2.8%</td>
<td>2.8%</td>
</tr>
<tr>
<td>0.5</td>
<td>4%</td>
<td>2.6%</td>
</tr>
<tr>
<td>0.7</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>0.9</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
B.2 Participant Survey Responses

For each of the 6 questions in the survey, this section contains a graph of the responses, as well as a table containing the $\chi^2$ statistic values and associated $p$-values, as well as the mean of each response. Note that for the Likert scale items the mean may not be as meaningful.

The AI agents seemed to change what they were doing based on my actions (5 point Likert)

![Bar graphs](image)

**Figure B.1** Player ratings for how reactive the agents’ behavior was, as ranked on a 5 point Likert scale
Table B.5 $\chi^2$ statistic values, $p$-values and means for player ratings for how reactive the agents' behavior was.

<table>
<thead>
<tr>
<th>$\chi^2(p)$</th>
<th>capricious</th>
<th>fanatical</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>capricious</td>
<td></td>
<td></td>
<td>-0.653</td>
</tr>
<tr>
<td>fanatical</td>
<td>6.640896 (0.156126)</td>
<td></td>
<td>-0.868</td>
</tr>
<tr>
<td>balanced</td>
<td>5.221008 (0.265363)</td>
<td>11.073333 (0.025752)</td>
<td>-0.904</td>
</tr>
</tbody>
</table>
The AI agents’ actions made sense (5 point Likert)

Figure B.2 Player ratings for how much sense the agents’ statements made, as ranked on a 5 point Likert scale
Table B.6 $\chi^2$ statistic values, $p$-values and means for player ratings for how much sense the agents’ statements made.

<table>
<thead>
<tr>
<th>$\chi^2(p)$</th>
<th>capricious</th>
<th>fanatical</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>capricious</td>
<td></td>
<td>-0.959</td>
<td></td>
</tr>
<tr>
<td>fanatical</td>
<td>8.388158 (0.078351)</td>
<td>-0.868</td>
<td></td>
</tr>
<tr>
<td>balanced</td>
<td>9.177632 (0.056810)</td>
<td>2.191176 (0.700645)</td>
<td>-0.769</td>
</tr>
</tbody>
</table>
Which of the following statements by <Agent name> did you believe at the time they made it? (select 0-7 statements; count)

![Figure B.3](image)

**Figure B.3** Distribution of the individual statements made by an AI agent that participants rated as being believed.
Table B.7 $\chi^2$ statistic values, $p$-values and means for how many statements by an AI agent the participants believed.

<table>
<thead>
<tr>
<th>$\chi^2(p)$</th>
<th>capricious</th>
<th>fanatical</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>capricious</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fanatical</td>
<td>13.444444 (0.061992)</td>
<td></td>
<td>2.774</td>
</tr>
<tr>
<td>balanced</td>
<td>7.916667 (0.340000)</td>
<td>39.302564 (0.000002)</td>
<td>2.481</td>
</tr>
</tbody>
</table>
Which of the following statements by <Agent name>made strategic sense to you at the time they made it, whether or not you believed it? (select 0-7 statements; count)

![Distribution of the individual statements made by an AI agent that participants rated as making sense.](image)

**Figure B.4** Distribution of the individual statements made by an AI agent that participants rated as making sense.
Table B.8 $\chi^2$ statistic values, $p$-values and means for how many statements by an AI agent the participants rated as making sense.

<table>
<thead>
<tr>
<th>$\chi^2(p)$</th>
<th>capricious</th>
<th>fanatical</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>capricious</td>
<td></td>
<td></td>
<td>2.104</td>
</tr>
<tr>
<td>fanatical</td>
<td>9.283883 (0.232912)</td>
<td></td>
<td>2.000</td>
</tr>
<tr>
<td>balanced</td>
<td>21.840293 (0.002707)</td>
<td>29.233333 (0.000131)</td>
<td>2.827</td>
</tr>
</tbody>
</table>
The AI agents played well (5 point Likert)

Figure B.5 Player ratings for how well the different AI agents played, as ranked on a 5 point Likert scale
Table B.9 $\chi^2$ statistic values, $p$-values and means for player ratings for how well the agents played.

<table>
<thead>
<tr>
<th>$\chi^2(p)$</th>
<th>capricious</th>
<th>fanatical</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>capricious</td>
<td></td>
<td></td>
<td>-1.020</td>
</tr>
<tr>
<td>fanatical</td>
<td>2.406116 (0.661522)</td>
<td></td>
<td>-0.849</td>
</tr>
<tr>
<td>balanced</td>
<td>14.268227 (0.006486)</td>
<td>15.722078 (0.003416)</td>
<td>-0.865</td>
</tr>
</tbody>
</table>
It was fun to play with the AI agents (5 point Likert)

Figure B.6 Player ratings for how much fun it was to play with the AI agents, as ranked on a 5 point Likert scale
Table B.10 $\chi^2$ statistic values, $p$-values and means for player ratings for how much fun it was to play with the agents.

<table>
<thead>
<tr>
<th>$\chi^2(p)$</th>
<th>capricious</th>
<th>fanatical</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>capricious</td>
<td></td>
<td></td>
<td>-0.714</td>
</tr>
<tr>
<td>fanatical</td>
<td>1.097222 (0.894713)</td>
<td>-0.830</td>
<td></td>
</tr>
<tr>
<td>balanced</td>
<td>1.375000 (0.848528)</td>
<td>1.732656 (0.784777)</td>
<td>-0.808</td>
</tr>
</tbody>
</table>
Holm-Bonferroni Correction

To account for multiple testing, the Holm-Bonferroni correction was used, which makes no assumptions about the underlying dependence structure of the response variables. It is performed by ordering the $p$ values of all comparisons in ascending order, and multiplying the first one with $n$, the second one with $n - 1$, etc., where $n$ is the number of comparisons. The resulting values are then compared to the chosen significance threshold, here $p \leq 0.05$. Note that because this process makes no assumptions about any dependence between responses, it may lead to Type II errors if such a dependence is actually present.

As a result of this process, the following comparisons were determined to be statistically significantly different:

- believe statements, balanced vs. fanatical: $0.000002 \cdot 18 = 0.000031$
- strategic statements, balanced vs. fanatical: $0.000131 \cdot 17 = 0.002230$
- strategic statements, balanced vs. capricious: $0.002707 \cdot 16 = 0.043305$

The following comparisons were weakly statistically significantly different, meaning a $p$ value of less than 0.1:

- aiskill, balanced vs. fanatical: $0.003416 \cdot 15 = 0.051235$
- aiskill, balanced vs. capricious: $0.006486 \cdot 14 = 0.090810$

And finally, the following comparisons were not statistically significantly different:

- reactive, balanced vs. fanatical: $0.025752 \cdot 13 = 0.334777$
- sensical, balanced vs. capricious: $0.056810 \cdot 12 = 0.681716$
- believe statements, fanatical vs. capricious: $0.061992 \cdot 11 = 0.681913$
- sensical, fanatical vs. capricious: $0.078351 \cdot 10 = 0.783508$
- reactive, fanatical vs. capricious: $0.156126 \cdot 9 = 1.405138$
- strategic statements, fanatical vs. capricious: $0.232912 \cdot 8 = 1.863295$
- reactive, balanced vs. capricious: $0.265363 \cdot 7 = 1.857541$
- believe statements, balanced vs. capricious: $0.340000 \cdot 6 = 2.039997$
- aiskill, fanatical vs. capricious: $0.661522 \cdot 5 = 3.307612$
- sensical, balanced vs. fanatical: $0.700645 \cdot 4 = 2.802580$
• fun, balanced vs. fanatical: $0.784777 \cdot 3 = 2.354332$

• fun, balanced vs. capricious: $0.848528 \cdot 2 = 1.697057$

• fun, fanatical vs. capricious: $0.894713 \cdot 1 = 0.894713$

Note that even without significance these results can provide direction for future research. For example, even though the perceived difference in how reactive the agents behaved was not statistically significantly different, the $p$ value of this comparison before the Holm-Bonferroni correction was 0.0257. Rather than testing for broad hypotheses, future research could focus on determining whether this is indeed a statistical fluke, or a significant difference.

### B.3 Participant Comments

As part of the study described in chapter 6, participants were asked to provide comments about the games they played with the different agent types. They could also choose to make their responses publicly available. Below I am providing all free form text responses verbatim from the participants that agreed to their publication, separated by agent type.

#### B.3.1 Balanced Agent

- I’m not sure how the voting ended the way it did. Anna claimed she started as a Werewolf, I claimed I swapped Anna and Dave. Dave was clearly now a Werewolf.

- Carol seemed more developed than the other AI agents.

- I counter claimed the rascal (who claimed rascal) but did not get his vote. That seems like a reversal from what should happen.

- They repeated a lot of things without trying to learn more. They should be predisposed to ask more questions.

- There was no response to my attempts to gather information or response to my statement that I had started as what someone else claimed (They should have realized that at least one of us was lying)

- Some of the agents lied about their identities when they had no reason to believe that there cards had been swapped.

- Twice I asked a question of the AI agents only for them to just make a statement unrelated to my question but asking again they would occassionally respond. The Seer never seems to expand upon what card they saw. I believed Brian was the Seer but he just kept telling
me Brian had seen the Robber. I could believe Carol was the Robber but Brian made no attempt to also inform that he had seen Carol’s card.

This setup was a perfect example of an easily solveable game. Anna opens with being Insomniac(?), Brian opens with seeing the Robber, Carol opens with being the Robber and David opens with being a village role. I say I’m the Rascal.

Anna responds saying she is the Werewolf (perfectly logical, her only doubt can be my lying of Rascal). Brian and Carol do nothing useful, merely repeating their statements. David also responds with something non-sensical (I was a village role). He should respond that he is also a Werewolf and explicitly lie about his role.

Because David says he’s a Village role, they originally started on different teams. If they both started as Werewolf, saying they’re a Werewolf doesn’t help their cause (because I swapped them still into a Werewolf) and if they were both Villagers, Anna has no reason to claim being a Werewolf. The only remaining question is whether Carol stole the Werewolf as the Robber so I asked Carol what she stole. She refused to respond to this question but I took this as the AI being incompetent. Claiming to steal the Seer from Brian is the obvious choice for a statement that can’t be disputed.

So I have already identified Anna started as the Werewolf. In the final round I make the statement that I swapped the roles (arguably I should do this in round 6 but I wasted so many actions asking questions that the AI refused to answer) so the AIs should vote for David.

The AIs voted for Anna (while I correctly voted for David). In fairness, this is plausible, the other players have no way to refute if I was the Rascal or the other Werewolf but I believe the odds here are actually 25% of lone Werewolf (3 centre cards + my seat where only my seat is 2 Werewolves).

The problem is that the AI just repeats itself continually and refuses to answer questions.

- The AI simply ignores the other players. The first round both Carol and I claimed to have started as the seer. Nobody seemed to notice nor questioned the fact. I asked Carol twice what she saw. She claimed to be the seer 6 more times in a row. I asked Dave if he swapped his neighbours cards after he claimed to be a rascal. He confirmed twice that he indeed was the rascal before finally denying that he had swapped the cards when I had stopped asking. Most of the players just repeated their initial claims a few times over and didn’t react to new information, didn’t try to uncover new information through questioning the opponents, didn’t really do much until the game was over where they killed me. Probably because Carol was so convincing, after all she did state she was the seer 7 times in a row: and who would lie so often? It obviously had to be me! In a social deduction game relying
on interaction the AI needs to actually both interact and deduct to pass as convincing humans.

- They seemed to ignore me and voted for me for no reason

- I think I got an answer to one of my questions this time (from Brian), but the other agents totally declined to investigate the discrepancy between what I and Carol were saying. We both declared that we started as the seer, but none of the other agents asked about this, and they again failed to accuse any of the particularly suspicious players (me and Carol, since our claims obviously didn’t line up) of being the werewolf, seeming to make a totally random choice instead.

- I’m not sure if the AI agents are actually trying to win. They seem to lie for no reason. They change stories randomly. They don’t respond to questions well.

- It was hard to get information from the AI to work with, even after I started sharing information.

- The AIs said the same thing way too often. Dave, for example, simply spammed "I saw the Robber". As a good player, he would have said where he saw it. Furthermore, why did Anna say she was an Insomniac? Of course it also makes sense for Villagers to lie, but that was unnecessary and got her lynched.

- Again, how are they deciding to vote? And the lies they decide to tell make no sense. Sometimes it feels like they might be responding to my answers but they do so just by throwing out something random that is likely not true, even when they have no reason to lie. Once again, I stopped trying to talk to them before the end of the game.

- Nothing they do makes sense. Even when I swapped their cards, they still voted for the old werewolf.

- Randomness of responses decreasing as expected if learning from a purely random start point. However, AI is still lying pointlessly when it is Village.

- again, no clue why they votes me besides human discrimination. there needs to be a method for the ai to claim other people’s role when they are seer/robber

- Kept saying the same things, did not answer questions.
B.3.2 Capricious agent

- They never ask questions and rely on me puzzling together what happened. After a bit it felt very much like they were just repeating statements, not utilizing any logical deduction. Anna f.i I felt only stated that she is a rascal, or that she started as a warewolf.

- I swapped Anna and Dave, and Dave ended up being the werewolf. This means Anna had to know Dave was a werewolf, but didn’t vote for him. Also, I suspected Brian was a werewolf, because he claimed to be the rascal, while I was the rascal. Why would a villager AI lie like that, when it just makes it harder for other villagers to figure out who the werewolf is. The AI didn’t seem to make logical choices.

- They didn’t seem to respond to my questions, or even to one another’s statements. They also failed to detect me as the werewolf, even though I thought I was behaving pretty suspiciously by declining to say anything during the last 5 rounds. It seems like their votes in the final round were totally random.

- The AI agents like to tell me they’ve seen my card (even though I am the Seer and my first discussion is to tell them I’ve seen Brian’s card). I don’t understand the voting intentions of them either. The Rascal tells us that it’s swapped the cards of it’s neighbours (Carol and I) and yet people vote for Anna (presumably who started as the Werewolf and Carol was the Insomniac and swapped into it) without any real justification.

- They often failed to respond to questions, instead repeating a statement of little if any value. They also voted correctly against me despite none making any sort of accusation which was strange especially since several claimed to have knowledge of roles other than mine but did not state who possessed them.

- It is hard to derive the AI’s intended voting patterns

- Robber and Seer seemed to ignore swap claim. Werewolf came out as starting werewolf too quickly.

- Once again, the AI lies when it has no reason to. Village won this game but it was a complete fluke. Also the Seer said it had seen my card but never said I was a Werewolf, which is insanely stupid.

- The AIs tend to ignore questions, and never ask any of their own. Backgspace sdkey doesnot work in this text box.

- They don’t change what they’re saying at all.
• I didn’t see why Carol was more believable than I was

• It felt very frustrating to see most of them not react to what I say.

• Much better! Now they changed their minds. However, it would be interesting to know why more people voted for me than Carol. Was it because she said "I started as Werewolf" more often? A real life player could be "read" to find it out, but against AIs it is impossible. I have no idea how they decide on who to vote for :p

• Playing as a Werewolf was too easy. The AI claimed their roles out of the gate, I could easily decipher what roles were in the center and then claim to be one of them the entire game.

• Is there a notion of counter claiming, or throwing shade on similar claims? For example, in this game, carol claimed my role and was switched with me.

It’s also unclear why the AI voted for me in this game (claimed stole rascal from rascal, when robber was in the center)

• I gave up trying to talk to them after a few rounds. It feels like trying to talk to a brick wall. I also don’t understand how the AI come to a decision on who to vote for? It made no sense for so many of them to vote for Anna in the last game.

• I’m not sure whether my action helped pointing the AI:s in the wrong direction but they certainly didn’t seem to put any effort into finding out that I was a werewolf.

• Asking if someone is the werewolf repeatedly garnered all the votes against them. Is this chance or an easy manipulation?

• The AI agents seem to have gotten more clever in this last game.

• AI agents don’t seem to know what they are talking about. If they lack rationale for their actions, it’s hard to draw conclusions about their role.

B.3.3 Fanatical agent

• The text only seemed to change once based on the questions I asked

• This game made some sense, though it’s unclear how much the AI had figured out.

• The agents just repeated the same statements every turn and thus so did I since they weren’t progressing the game and I was a Werewolf so I didn’t believe they would have any justification for voting for me.
• They completely ignore me. For a social game that probably isn’t ideal. Carol was asked I think three times if she was the rascal. She never directly answered. Dave stated 6 times in a row that he had seen the robber, and then in the very last turn changed to "i have seen your card". The AI seems to be mostly talking to itself and doesn’t try to deduce any new information from the events that occured. It’s a bit like a FSM or a modal logic agent that forgot to move from their starting state and is just digging into the same tracks.

• Their decisions seem to be completely random and not based on any knowledge they should actually have about the cards they have seen.

• It’s challenging to play a deceitful game without looking at someone’s face.

• I’m still having a hard time believing that their actions make sense. Their actions seem to bear little relationship to their nominal goals.

• more an interface thing, after dave said he saw my card, I couldn’t ask him what card he saw. Carol didn’t make it clear she robbed me.

• Again they didn’t seem to respond to my actions, and again they seemed to make a totally random choice of who to kill, apparently disregarding suspicious behavior like multiple players claiming to have the same role (or, in Anna’s case, changing which role she claimed to have partway through).

• The AI Agents lie at the appropriate time and tell the truth at the appropriate time dependant on their role. However they do not adapt or try to draw truth through lies, which is an important part of the game.

• It is not clear how Carol knew she was the Seer and then when I asked questions nothing happened.

• None of the 4 AIs changed the sentence they said... They did not respond to my questions and sentences at all D: Idk, maybe I did something wrong. Gonna retry it now!

• They do not respond to questions well if at all. It would be more fun to play with them if it felt like they could react to what I asked.

• They didn’t really help localizing the werewolf so I could only base my game decision on which person lied the most.

• Seemed like the Seer claim before the Rascal claim was a lucky blind call, if I understood the order right.
• They repeat claims without purpose. The seer lied for unclear reason, and by telling the same lie as the werewolf made it impossible to determine which was which. This was not to the seer's advantage.

• They didn't respond almost at all.

• Two AI agents claimed to be Werewolves, one truthfully. This indicates a fundamental flaw in the programming of the AI, sufficient to invalidate the study.