THE UNIVERSITY
of EDINBURGH

Utilising Policy Types for Effective Ad Hoc Coordination in Multiagent Systems

Stefano Vittorino Albrecht

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

The University of Edinburgh

2015

Abstract

This thesis is concerned with the *ad hoc coordination* problem. Therein, the goal is to design an autonomous agent which can achieve high flexibility and efficiency in a multiagent system that admits no prior coordination between the designed agent and the other agents. Flexibility describes the agent's ability to solve its task with a variety of other agents in the system; efficiency is the relation between the agent's payoffs and time needed to solve the task; and no prior coordination means that the agent does not a priori know how the other agents behave. This problem is relevant for a number of practical applications, including human-machine interaction tasks, such as adaptive user interfaces, robotic elderly care, and automated trading agents.

Motivated by this problem, the central idea studied in this thesis is to utilise a set of policies, or *types*, to characterise the behaviour of other agents. Specifically, the idea is to reduce the complexity of the interaction problem by assuming that the other agents draw their latent type from some known or hypothesised space of types, and that the assignment of types is governed by an unknown distribution. Based on the current interaction history, we can form posterior beliefs about the relative likelihood of types. These beliefs, combined with the future predictions of the types, can then be used in a planning procedure to compute optimal responses. The aim of this thesis is to study the potential and limitations of this idea in the context of ad hoc coordination.

We formulate the ad hoc coordination problem using a game-theoretic model called the *stochastic Bayesian game*. Based on this model, we derive a canonical algorithmic description of the idea outlined above, called *Harsanyi-Bellman Ad Hoc Coordination* (HBA). The practical potential of HBA is demonstrated in two case studies, including a human-machine experiment and a simulated logistics domain. We formulate basic ways to incorporate evidence (i.e. observed actions) into posterior beliefs and analyse the conditions under which the posterior beliefs converge to the true distribution of types. Furthermore, we study the impact of prior beliefs over types (that is, before any actions are observed) on the long-term performance of HBA, and show empirically that automatic methods can compute prior beliefs with consistent performance effects. For hypothesised (i.e. "guessed") type spaces, we analyse the relations between hypothesised and true type spaces under which HBA is still guaranteed to solve its task, despite inaccuracies in hypothesised types. Finally, we show how HBA can perform an automatic statistical analysis to decide whether to reject its behavioural hypothesis, i.e. the combination of posterior beliefs and types.

Lay Summary

Assume a machine has to interact with some human (or other machine) on a given task to achieve some goal. To be successful in this setting, the machine should account for the following three aspects: (1) different humans may have different behaviours for the given task, (2) humans typically have little patience when interacting with machines, and (3) the machine may not know ahead of time how the human behaves. Therefore, the machine should be able to learn quickly how to interact effectively with humans whose behaviours are initially unknown. We refer to these kinds of problems as *ad hoc coordination* problems.

One way to deal with such a situation is sometimes referred to as "pigeon-holing". In it, we guess several potential behaviours (e.g. from past experience) that the human could have, and compare the predictions of these behaviours with the actions taken by the human. We can then plan our own actions with respect to those guessed behaviours which we believe are most likely, based on the human's observed actions. In this thesis, we essentially study machines that use this kind of reasoning.

Specifically, we use a mathematical model called the *stochastic Bayesian game* to formalise the interaction problem. Based on this model, we define an algorithm, called *Harsanyi-Bellman Ad Hoc Coordination* (HBA), which corresponds to the pigeon-holing reasoning outlined above. The practical potential of HBA is demonstrated in experiments involving humans as well as other machines. Furthermore, we analyse the conditions under which HBA's beliefs about the likelihood of behaviours will eventually be correct, and we also study the impact of initial beliefs (similar to "anticipation" or "prejudice") on the long-term performance of HBA. For guessed behaviours, we study the theoretical conditions under which HBA can solve its task even when the guessed behaviours are incorrect, and we show how HBA can itself decide whether or not to trust its guessed behaviours during the interaction.

Acknowledgements

Throughout my doctoral studies, I was fortunate to receive support from a number of individuals and organisations:

I wish to thank my doctoral adviser, Subramanian Ramamoorthy, for his guidance and advice, for many stimulating conversations about research and academic life, and for giving me the freedom to pursue my own ideas. Thanks to his advice I was able to present my M.Sc. dissertation at a high-profile conference, which gave me a valuable head start in my academic career.

It was at that conference that I first met Jacob Crandall, who would later become a close collaborator. I wish to thank him, likewise, for many stimulating conversations about research and academia, and for giving me the opportunity to visit his research group at Masdar Institute in Abu Dhabi.

I would like to thank my secondary adviser, Michael Rovatsos, and my external adviser, Ed Hopkins. Their comments and suggestions helped me in the development and presentation of my ideas. I would also like to thank my examiners, Alex Lascarides and Sarit Kraus, for their valuable comments and suggestions.

The great majority of my simulations were performed on the "Eddie" computing cluster of the Edinburgh Compute and Data Facility (ECDF). I would like to thank the staff at ECDF for their excellent support.

During my time as a doctoral student, I had the privilege of being invited to attend several prestigious science events. This includes the First Heidelberg Laureate Forum, the Fifth Lindau Meeting on Economic Sciences, and the final of the EPSRC UK ICT Pioneers 2014 competition. I would like to thank the corresponding organising bodies for giving me these unique opportunities.

I am grateful for the financial support of the German National Academic Foundation, which allowed me to focus most of my time on my research. I am also grateful for the travel grants I received from the Institute of Perception, Action and Behaviour, the International Foundation for Autonomous Agents and Multiagent Systems, the Association for Uncertainty in Artificial Intelligence, and the Association for the Advancement of Artificial Intelligence.

My foremost thanks go to my family and parents-in-law, for their continual support in all matters of life. In particular, I am deeply grateful to my brother for enabling me to embark on this wonderful journey in the first place.

Finally, my dear wife, I thank you for all your love and support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Stefano Vittorino Albrecht*)

Table of Contents

List of Figures

List of Algorithms

Common Abbreviations and Symbols

(See Section 2.8 for synonymous terminology.)

SBG	Stochastic Bayesian Game
HBA	Harsanyi-Bellman Ad Hoc Coordination
α	Ad hoc agent
S	State space
T	State transition function
A_i	Set of actions available to player i
π_i	Strategy function of player i
u_i	Payoff function of player i
Θ_j	Type space of player j
Θ_j^+	Set of true types of player j
Θ_j^*	Set of hypothesised types for player j
H^t	History of states and joint actions until time t
$\mathbb{H}$	Set of all histories H^t, for all t
Υ	Type distribution
$\mathbb{T}$	Set of type distributions
P_j	Prior belief about types of player j
$\Pr_j$	Posterior belief about types of player j
Δ	Set of all probability distributions
T	Test statistic

Chapter 1

Introduction

This thesis is concerned with *ad hoc coordination* in multiagent systems. Therein, the goal is to design an autonomous agent which is able to achieve high flexibility and efficiency in a multiagent system that admits no prior coordination between the designed agent and the other agents. Flexibility describes the agent's ability to solve its task with a variety of other agents in the system; efficiency is the relation between the agent's payoffs and time needed to solve the task; and no prior coordination means that the agent does not a priori know how the other agents behave. In particular, there is no prior coordination of behaviours in terms of predefined action protocols, shared world models, explicit communication of beliefs, etc.

This problem is relevant for a number of important applications. For example, consider an adaptive user interface which interacts with a user to complete a certain task. Depending on the complexity of the task, the user may exhibit a variety of different behaviours and the interface should be flexible enough to accommodate such a variety. Furthermore, the user typically has little patience for lengthy learning periods based on trial and error, hence the interface should quickly learn what the user intends to do and how to assist the user adequately. Finally, unless the user is already trained with the interface and can be identified during the interaction, the interface will typically not know a priori how the user behaves.

The characterisation of flexibility, efficiency, and no prior coordination is shared more generally by many human-machine interaction problems. Another example in this category is given by robotic elderly care, in which a mobile robot is used to assist residents of an elderly home to accomplish their daily tasks. Again, the residents may have a variety of behaviours for a given task, which the robot should account for by being flexible in its interaction. Moreover, given that a single robot may be used to

assist a larger group of residents, there may be little time for extensive learning trials, hence efficient interaction is required. Finally, unless the robot has access to behavioural profiles for each resident, it may not know how the residents behave and assumptions regarding prior knowledge and communication can be problematic.

As a final example in which ad hoc coordination arises naturally, consider an electronic trading market where a collective of trading agents exchange resources. A task in this setting could be to acquire a specific amount of a certain commodity, subject to constraints about prices and time. To be successful, a trading agent should be able to interact with other agents who may exhibit a range of different trading strategies. In addition, given that time is usually an important constraint, the agent should be able to find an efficient trade-off between time needed to accomplish its task and trade expenses. Finally, prior coordination of behaviours may not be desirable (e.g. agents do not want others to know their strategy) or even allowed (e.g. market laws may forbid certain kinds of prior coordination between market participants).

Unfortunately, current methods for multiagent interaction are not suitable for ad hoc coordination, due to a number of limitations. For example, many methods have been designed in the context of homogeneous systems, in which all agents use the same algorithm (Albrecht and Ramamoorthy, 2012). This significantly reduces the flexibility of such methods, and it can be viewed as a strong form of prior coordination. Some methods attempt to learn models of other agents based on past observations. However, such methods typically require long learning periods in order to produce reliable models, and the models themselves may be based on restrictive assumptions. The former reduces the efficiency of such methods while the latter reduces their flexibility. Therefore, what is needed is an agent that can quickly learn to interact effectively with a variety of other agents, yet without knowing a priori how they behave.

A related problem is known in game theory as the incomplete information game. Therein, each player has some "private" information which is relevant to the decision making of the player. This information may pertain to any structural aspects of the game, such as payoff functions, strategy spaces, etc. The fact that this private information is unknown to the other players means that here, likewise, players have to interact with other players whose behaviours are unknown. Harsanyi (1967, 1968a,b) proposed to reduce the complexity of this problem by assuming that the private information, or *type*, of each player is drawn from some space of types, and that the assignment of types is governed by a probability distribution over this space. This transformation gives rise to what is now commonly known as a *Bayesian game*, and much research has been

devoted to the study of learning and equilibria in such games.

Can we, similarly, reduce the complexity of ad hoc coordination problems by assuming that the other agents draw their unknown behaviours, or types[1], from some space of types? Of course, game theorists are primarily interested in equilibrium attainment, in which a collective of players play in some sense optimally against each other. In contrast, our interest is in the flexibility and efficiency of an individual agent, which do not require an equilibrium concept. Furthermore, the normative assumptions of concepts such as the Nash equilibrium (Nash, 1950), including perfect rationality with respect to one's utilities, may be difficult to justify in ad hoc coordination in which we assume that the behaviour of other agents is a priori unknown. Nonetheless, the idea of types is appealing even as a best-response rule from an individual perspective. Specifically, when interacting with an agent whose behaviour is unknown, we may hypothesise a set of types, each specifying a complete behaviour, and plan our own actions with respect to those types which most closely match the agent's observed actions.

To illustrate this idea, consider a simple example in which a machine plays a game of Rock-Paper-Scissors against a human. The human may employ a variety of different strategies unknown to the machine. However, the machine may hypothesise a set of possible strategies that the human could use, such as "play previous action of other player", "play action that would have won in previous round", etc. These hypothesised strategies, or types, may be provided by a domain expert or generated automatically. By comparing the predictions of the types with the actual actions taken by the human, the machine can compute a posterior belief to describe the relative likelihood of types. The posterior belief and type predictions can then be used to reason about subsequent turns in the game, allowing the machine to plan optimal responses.

This method has a number of features particularly suited for ad hoc coordination problems: First of all, the fact that we may hypothesise *any* types of behaviours, including behaviours which are not necessarily "rational" with respect to an agent's utilities, gives us the flexibility to deal with a variety of agents. Furthermore, by committing to a predefined set of types, we typically require only a fraction of the observations that other forms of learning, such as opponent modelling, require. Even if the other agents change their types over time, we may be able to adjust our posterior beliefs relatively quickly, while other learning methods may have to start from scratch. Finally, the fact that the types specify complete behaviours means that we can make predictions even

[1] Our interpretation of types as "behaviours" is consistent with the original definition in (Harsanyi, 1967, cf. "normalised strategies", Section 7 (p. 180)), which defines types as parameters in both strategy and payoff functions (as we do in our work; see Section 3.1 for further discussion.)

for unseen situations, while other forms of learning may only be able to make predictions for situations that have been encountered previously. These features can enable an agent to plan its actions efficiently and with great flexibility.

On the other hand, there are several concerns and questions associated with this method. An important concern is the possibility that the hypothesised types may be incorrect, in the sense that their predictions deviate from our observations. Such inaccuracies may have a profound impact on the quality of planning, which raises the question of how much and what kind of inaccuracies are acceptable, and if it is possible to ascertain whether to trust or altogether reject the hypothesised types. Other important questions pertain to the dynamics of beliefs. What impact do prior beliefs, before observing any actions, have on our long-term performance? And, in the ideal case in which the true type of an agent is among the hypothesised types, can we be certain that the posterior belief will eventually reveal the true type? These questions are essential to the method outlined above, and they are the central topics of the present thesis.

1.1 Aims and Contributions

This thesis is a comprehensive study of the type-based methodology outlined above. Specifically, the goal of the thesis is to study the potential and limitations of this idea in the context of ad hoc coordination problems, both theoretically and empirically. We do so by identifying and addressing a spectrum of central questions, from formalising the type-based methodology to deriving a canonical algorithm; from empirical evaluations in simulated and human-machine experiments to theoretical analyses of various properties pertaining to beliefs and types (Figure 1.1). The following is a brief summary of the questions addressed and the contributions made.

The thesis sets out by addressing the basic question of how to formalise the type-based method. We propose a game-theoretic model, called the *stochastic Bayesian game* (SBG), which combines Harsanyi's notion of types with stochastic state transitions. Therein, each player has one of a number of types which specify the player's preferences and behaviour, and the assignment of types is governed by some distribution over types. Given the players' actions, the system transitions stochastically between different states. Based on this model, we give precise definitions of an agent's flexibility and efficiency, and we define the ad hoc coordination problem as designing an agent which optimises its flexibility and efficiency in a SBG in which the true type spaces and type distribution are unknown to the agent.

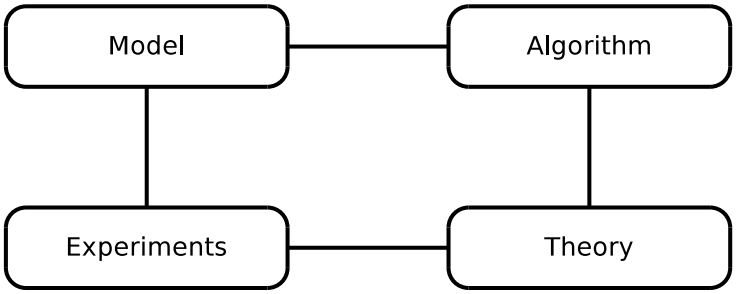


Figure 1.1: Components of thesis.

Moreover, we derive a best-response rule called *Harsanyi-Bellman Ad Hoc Coordination* (HBA), which can be viewed as a canonical algorithmic description of the type-based method. HBA utilises a set of hypothesised types, where each type specifies a possible behaviour for an agent. Specifically, given the current interaction history, each type generates a probabilistic prediction of the agent's next action. By comparing these predictions with the actions taken by the agent, HBA computes a posterior belief to quantify the relative likelihood of types. The posterior belief and the type predictions are then used to make predictions about future trajectories in the interaction, based on which HBA computes an optimal response.

To show how HBA can be applied, we present two empirical case studies. In the first case study, we used HBA to interact with human subjects in one of two simple matrix games. The implementation of HBA was "direct" in that it generated a complete tree of all possible future trajectories, up to a certain depth. The results show that a handful of simple types was sufficient to interact effectively with humans who exhibited a range of behaviours. In the second case study, we used a more complex implementation of HBA in a simulated logistics domain. Therein, the goal is to collect a number of different objects by collaborating with other agents whose behaviours are initially unknown. We show how HBA can be implemented as a reinforcement learning procedure to solve such complex problems. The procedure uses stochastic sampling to "roll-out" possible interaction trajectories, based on the types and posterior beliefs.

One of the central questions associated with HBA is the evolution of beliefs. Specifically, how should HBA incorporate evidence in its beliefs, and under what conditions will the beliefs converge to the true distribution of types? These questions are of central importance, since incorrect beliefs may lead to suboptimal action choices. We introduce three basic formulations of posterior beliefs, each suitable for a different class of type distributions, and provide theoretical conditions under which they are guaranteed to converge to the true type distribution. These theoretical insights enable the user to

choose an appropriate posterior formulation for a given problem.

Before HBA observes any evidence on which to base its posterior belief, it will have to make a prior judgment of how likely the types are. Could this prior belief have a significant impact on the long-term performance of HBA? If so, how? And, importantly, can we *automatically* compute prior beliefs with the goal of improving our long-term performance? To find answers to these questions, we conducted a comprehensive series of experiments which compared 10 methods to automatically compute prior beliefs from a given set of types. The results show that prior beliefs can indeed have a significant impact on the long-term performance, and that the depth of HBA's planning horizon plays a central role. Moreover, we show that automatic methods can compute prior beliefs with consistent performance effects across a variety of scenarios.

A possible concern associated with HBA is the fact that the hypothesised types may be incorrect. This may range from slight deviations in predicted action probabilities, to predicting entirely different actions from what was observed. Therefore, an important question is how accurate the hypothesised types have to be in order for HBA to be able to solve its task? To address this question, we describe a methodology whereby we formulate a hierarchy of desirable termination guarantees using a probabilistic temporal logic, and analyse the conditions under which they are met. Furthermore, we provide a novel characterisation of optimality which is based on the notion of probabilistic bisimulation (Larsen and Skou, 1991). In addition to concisely defining what constitutes optimal type spaces, this allows the user to apply efficient model checking algorithms to verify optimality in practice.

The preceding analysis of optimal type spaces is done before any interaction and with respect to the true types of the other agents. How can we decide whether or not to trust our hypothesised types *during* the interaction? Unfortunately, posterior beliefs are only a measure of relative likelihood between types, but they are no measure of absolute truth. That is, even if our posterior belief points to one type, it does not mean that the agent is indeed of this type. To complement the beliefs, we propose an automatic statistical analysis in the form of a frequentist hypothesis test. The proposed method can incorporate multiple statistical criteria in the test statistic and learns its distribution during the interaction process, with asymptotic correctness guarantees. We present results from a comprehensive set of experiments, demonstrating that the algorithm achieves high accuracy and scalability at low computational costs.

1.2 Scope and Limitations

Throughout this work, we will make a number of assumptions pertaining to what agents can "see" and "do". These assumptions largely determine the scope and limitations of our work. To calibrate the reader's expectations, it is worth pointing out the most important assumptions before proceeding with the main material. Note that all assumptions will be made precise and discussed in detail in the later chapters.

First of all, we assume that all agents can make decisions based on the *entire* interaction history, which includes all past states and joint actions. An agent is not required to base its decisions on the entire history, but it can do so if it chooses to. Furthermore, we do not restrict the behaviour of agents by requiring a specific implementation, such as decision trees, finite state machines, etc. Rather, an agent's behaviour (or type) may implement any logic, and we will simply view it as a *black-box programme*. In addition, agents are allowed to specify *probabilities* over actions rather than choosing actions deterministically. Finally, we do not require that agents' preferences are aligned in any sense (that is, they may have common or conflicting interests). Together, these features mean that our work is applicable to a very broad spectrum of problems.

Our model makes a number of informational assumptions. Firstly, we assume that all state and action spaces are finite (i.e. discrete). We believe it is possible to extend many of our results to infinite (e.g. continuous) state and action spaces, but this may require extra analysis and increase the computational complexity of our methods. Furthermore, we assume that the state and action spaces, as well as the dynamics by which the system transitions between states, are a priori known to us. This allows us to use a relatively simple but effective planning procedure in our analysis. Finally, we assume that we can always observe the current system state and the past actions chosen by all players (this is sometimes referred to as *full observability*). Therefore, our work may not be directly applicable to problems in which the system state or the players' past actions are not directly observed. Nonetheless, we expect that many of the observations we make regarding the type-based methodology will still hold.

1.3 Thesis Structure

The thesis consists of 9 chapters in total. Chapters 3 to 8 contain the main contributions, following the summary in Section 1.1. The structure of the thesis is shown in Figure 1.2, and is briefly summarised as follows:

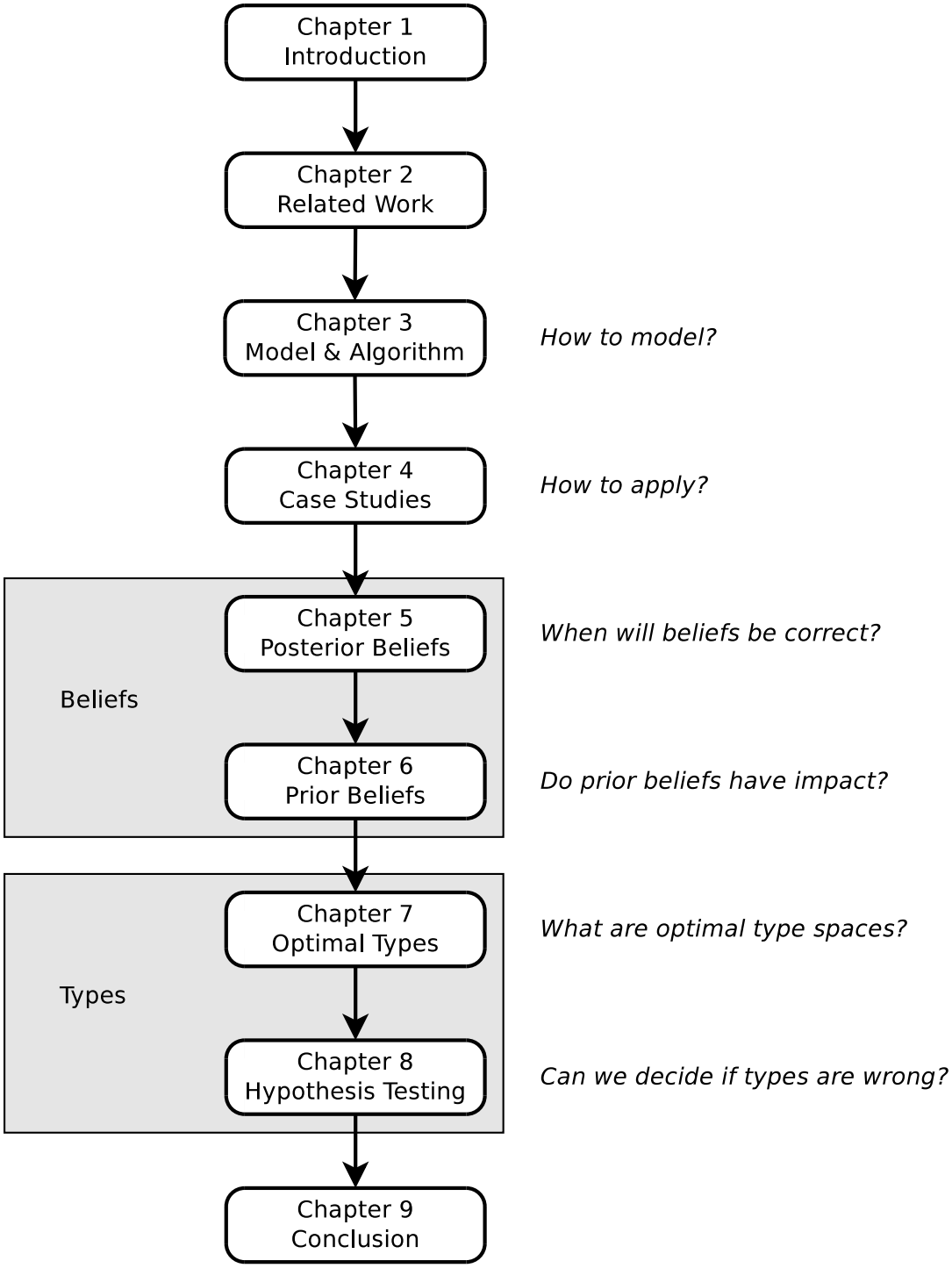


Figure 1.2: Structure of thesis.

- **Chapter 2** situates this thesis in the research literature and elaborates on some of the claims made at the beginning of Chapter 1.
- **Chapter 3** introduces the basic model, SBG, and algorithm, HBA, which are used as the basis for all subsequent chapters.
- **Chapter 4** presents two empirical case studies to demonstrate how HBA can be applied in practice.
- **Chapter 5** provides an analysis of convergence and correctness properties for different formulations of posterior beliefs.
- **Chapter 6** investigates the practical long-term impact of prior beliefs and whether they can be computed automatically.
- **Chapter 7** considers inaccuracies in hypothesised types and introduces a formal notion of optimality.
- **Chapter 8** describes an automatic statistical analysis to decide during the interaction if the hypothesised types should be rejected.
- Finally, **Chapter 9** concludes this thesis, highlights key results, and discusses directions for future research.

The recommended reading order is shown by the arrows in Figure 1.2. However, the only mandatory part is Chapter 3, after which the reader may choose to proceed with any of the subsequent chapters. Each chapter begins with a brief summary of the questions addressed in the chapter as well as the contributions made therein.

1.4 Publications

Parts of this thesis appeared in the following publications:

- S.V. Albrecht, S. Ramamoorthy. Are You Doing What I Think You Are Doing? Criticising Uncertain Agent Models. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, Amsterdam, Netherlands, 2015.

  → Chapter 8

- S.V. Albrecht, J.W. Crandall, S. Ramamoorthy. An Empirical Study on the Practical Impact of Prior Beliefs over Policy Types. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, 2015.
  → Chapter 6

- S.V. Albrecht, S. Ramamoorthy. On Convergence and Optimality of Best-Response Learning with Policy Types in Multiagent Systems. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, Quebec City, Canada, 2014.
  → Chapters 5 and 7

- S.V. Albrecht, S. Ramamoorthy. A Game-Theoretic Model and Best-Response Learning Method for Ad Hoc Coordination in Multiagent Systems (Extended Abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, St. Paul, Minnesota, USA, 2013.
  → Chapters 3 and 4

Currently under review:

- S.V. Albrecht, J.W. Crandall, S. Ramamoorthy. Belief and Truth in Hypothesised Behaviours. Submitted for review to *Artificial Intelligence Journal* in July 2015.
  → Chapters 5, 6, 7, and 8

Related to this thesis, though not officially part of it, are the following publications:

- S.V. Albrecht, J.W. Crandall, S. Ramamoorthy. E-HBA: Using Action Policies for Expert Advice and Agent Typification. In *Proceedings of the AAAI-15 Workshop on Multiagent Interaction without Prior Coordination*, Austin, Texas, USA, 2015.

- S.V. Albrecht, S. Ramamoorthy. Comparative Evaluation of MAL Algorithms in a Diverse Set of Ad Hoc Team Problems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain, 2012.

Chapter 2

Related Work

This chapter situates this thesis within the wider research literature. We will discuss a number of related works and point out differences and similarities to our work. However, we note that this chapter is not intended to be an exhaustive listing of all related works, and we also refer to the references made within the cited works.

2.1 Ad Hoc Teamwork

This thesis was originally motivated by a challenge paper written by Stone et al. (2010), in which they describe the problem of *ad hoc teamwork*. Therein, the goal is to design an autonomous agent, called the *ad hoc agent*, which can collaborate with an existing team of other agents with no or only limited opportunity for prior coordination.[1] This problem was motivated by the fact that much prior research had focused on teamwork based on prior coordination (e.g. Kaminka and Frenkel, 2007; Grosz and Kraus, 1999; Tambe, 1997; Grosz and Kraus, 1996). See also (Wooldridge, 2009, Chapter 8).

Similar to ad hoc coordination, the key aspects of ad hoc teamwork are that the ad hoc agent should be flexible enough to interact efficiently with a variety of other agents, with no prior coordination. However, there are several differences to ad hoc coordination[2]: First of all, ad hoc teamwork assumes that the interests of all agents (including the ad hoc agent) are perfectly aligned, which means that their payoffs are identical (Stone et al., 2010). In contrast, ad hoc coordination does not assume that

[1] For completeness, it should be pointed out that similar problems were discussed earlier under the names of "pickup teams" (Dias et al., 2006) and "impromptu teams" (Bowling and McCracken, 2005).

[2] Initially, we too used the term "ad hoc teamwork"; see (Albrecht and Ramamoorthy, 2012). We decided to use the term "ad hoc coordination" after an anonymous reviewer pointed out that ad hoc teamwork assumes common payoffs, which we do not. The new term also serves to differentiate our formal definition of the problem from the informal description in (Stone et al., 2010).

the interests of the involved agents are necessarily aligned. Furthermore, the problem description in (Stone et al., 2010) is of a procedural nature, whereas ad hoc coordination is based on a precise formal definition (see Chapter 3). Finally, ad hoc coordination has a very strict notion of "no prior coordination", namely that the ad hoc agent has *no prior knowledge* of the other agents' behaviours. On the other hand, much work in ad hoc teamwork has assumed that the ad hoc agent a priori knows the other agents' behaviours (e.g. Genter and Stone, 2014; Genter et al., 2013; Agmon and Stone, 2012; Stone and Kraus, 2010) or the structure of their behaviours (e.g. Agmon et al., 2014; Chakraborty and Stone, 2013).

Closely related to our work is the work of Barrett (2014), who studied a variant of the type-based method in the context of ad hoc teamwork. In (Barrett et al., 2011), an algorithm similar to the one discussed in Section 4.2 is evaluated in a version of the pursuit domain. This algorithm is later augmented by learning agent models (or types) from past interactions (Barrett et al., 2013). In the same work, they also show how transfer learning may be used to improve the learned models based on observations from the current interaction. Finally, in (Barrett and Stone, 2015), they further augment the algorithm by learning both agent models and corresponding response policies offline from past interactions. We view these works as complementing our work, in the sense that they are focused on utilising past experience whereas we are focused on issues pertaining to the evolution of beliefs and optimality of types. Another notable difference is that Barrett et al. assume no adaptation in the behaviour of other agents, while we explicitly take learning behaviours into account.

2.2 Multiagent Learning

Multiagent learning is concerned with the design and analysis of agents that can learn to interact with other agents. There exists a large body of research in this area, and a number of different approaches have been studied (e.g. Banerjee and Sen, 2007; Conitzer and Sandholm, 2007; Foster and Young, 2003; Hu and Wellman, 2003; Bowling and Veloso, 2002; Hart and Mas-Colell, 2001; Claus and Boutilier, 1998). We refer to (Young, 2004) for an excellent discussion of different learning methods. The principal algorithm studied in this thesis, HBA, can be viewed as a learning algorithm as well, in the sense that the evolution of its beliefs form a continual learning process.

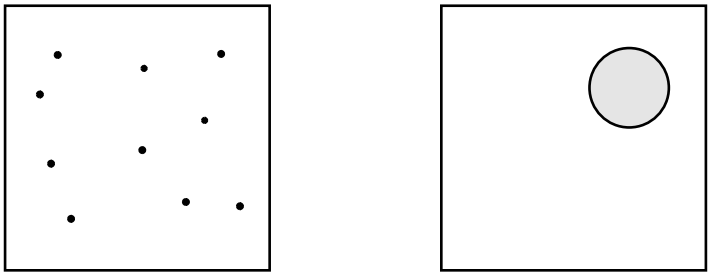
Multiagent learning algorithms can be categorised according to several criteria. One criterion is whether the algorithm controls a single agent or a collective of agents.

This thesis is concerned with the ad hoc coordination problem, in which the algorithm controls a single agent to interact with previously unknown other agents. Hence, our algorithm, HBA, controls a single agent. Another criterion is whether the algorithm seeks an equilibrium solution or primarily tries to optimise its own payoffs (the latter is sometimes called a *best-response* algorithm). In this work, we seek to optimise the flexibility and efficiency of the controlled agent, hence our algorithm belongs to the latter category. In fact, it can be argued whether it makes sense at all to seek an equilibrium solution in ad hoc coordination problems. This is because solution concepts such as the Nash equilibrium (Nash, 1950) implicitly make strong assumptions about the behaviour and preferences of the involved players, including perfect rationality and selfishness. However, in ad hoc coordination, there is no guarantee that the involved agents satisfy such assumptions. They may, in fact, not seek any equilibrium solution at all.

A reason why ad hoc coordination is an important open problem is the fact that many multiagent learning algorithms are inadequate for such problems. In a comprehensive experimental study (Albrecht and Ramamoorthy, 2012), we compared several salient methods of multiagent learning in ad hoc coordination problems and pointed out a number of limitations. Perhaps the most crucial limitation is the fact that many multiagent learning algorithms were designed in the context of homogeneous systems, in which all agents use the same algorithm. This may be implicit, in the sense that performance guarantees hold only in self-play (i.e. the same algorithm playing against itself), or explicit, by introducing additional coordination mechanisms such as communication protocols (e.g. Sen et al., 2003). However, in the context of ad hoc coordination, this may significantly reduce the flexibility of such algorithms. Another limitation is the fact that many algorithms, in particular those based on reinforcement learning, require long learning periods, even for the simplest tasks (e.g. Claus and Boutilier, 1998). This, in turn, reduces their efficiency (as defined in this work; see Chapters 1 and 3).

2.3 Opponent Modelling

Opponent modelling attempts to learn a model of an agent's behaviour based on the agent's observed actions. Formally, a model can be viewed as a function which takes as input the current (or any) interaction history and returns a deterministic or probabilistic prediction of the agent's next action. The learning usually takes place in an iterative fashion, after each observed action, by setting a number of parameters in some predefined model structure. The resulting model can be used in a planning procedure to compute



(a) Individual types (b) Class of models

Figure 2.1: Visualisation of type-based and "traditional" opponent modelling. The square represents the space of all possible behaviours. Each point in the square corresponds to a complete behaviour. (a) The type-based method allows us to hypothesise any types of behaviours. (b) Traditional methods cover an entire class (circle) of behaviours, but no behaviours outside that class.

optimal responses. Many different methods for opponent modelling have been studied (e.g. Hindriks and Tykhonov, 2008; van den Herik et al., 2005; Wendler and Bach, 2004; Steffens, 2004; Gmytrasiewicz and Durfee, 2000; Carmel and Markovitch, 1993).

The type-based method studied in this thesis can be viewed as a special case of opponent modelling. Here, the model structure consists of some finite number of hypothesised types, each of which is itself a model, and the parameters are the probabilities in the beliefs. (Other works define continuous type spaces, e.g. Southey et al. (2005); Chalkiadakis and Boutilier (2003).) In contrast to this, more "traditional" methods for opponent modelling assume a skeleton implementation of a behaviour, such as a decision tree or a finite state machine, and set implementation-specific parameters, such as node values and edges, to account for the observed actions. One of the earliest formulations of such a traditional method is *fictitious play* (Brown, 1951), in which the other agent's behaviour is modelled as a stationary probability distribution over the agent's actions, and the learning takes place by monitoring the action frequencies. Figure 2.1 visualises the difference between type-based and "traditional" methods.

In the context of ad hoc coordination, the type-based method has several key advantages over traditional methods: Firstly, traditional methods are by definition confined to a limited class of behaviours (those which can be represented by the assumed implementation), whereas the type-based method is free to hypothesise any types of behaviours, regardless of their implementation. This potentially gives us the flexibility to account for a greater variety of behaviours. Secondly, since types specify complete behaviours,

they can be used to make predictions for unseen situations. In contrast, models produced by traditional methods may only make reliable predictions for situations that have been encountered before (or similar situations, depending on the method's ability to generalise observations). In addition, if the observed agent switches to an entirely different behaviour, the type-based method may be able to respond quickly to this change by updating its belief over types (provided that the new behaviour is accounted for by another type), while the traditional method would typically have to start its learning from scratch. The latter two properties of the type-based method are useful in achieving high efficiency in ad hoc coordination problems.

Of course, as with any opponent modelling technique, there is a risk that the underlying assumptions do not hold, which means that the model may be inaccurate. Translated to the type-based method, this means that the hypothesised types may be incorrect, and this is one of the main concerns addressed in this thesis. Nonetheless, in this regard, the type-based method has one more useful feature: it allows us to include traditional modelling methods as a special kind of type. Therefore, if none of hypothesised types are correct, we can fall back to models created by the traditional methods.

2.4 Bayesian Games

Perhaps the earliest formulation of the type-based methodology was in the form of *Bayesian games* by Harsanyi (1967, 1968a,b). Bayesian games are an attempt to address incomplete information games, in which certain aspects of the game (typically the payoff distributions of other players) are unknown. Harsanyi proposed to model this "private information" as *types*: every player has one of a number of types unknown to the other players, and the assignment of types is governed by some distribution over types. By assuming that the type spaces and type distribution are common knowledge, this reduces the incomplete information game to a complete (but imperfect) information game, admitting a solution in the form of the *Bayesian Nash equilibrium*. While this idea was controversial at the time[3], Bayesian games have become a firm part of game theory and there exists a substantial body of work on learning and equilibria in Bayesian games (e.g. Nachbar, 2005; Dekel et al., 2004; Kalai and Lehrer, 1993; Jordan, 1991).

This thesis can be viewed as applying Harsanyi's notion of types to the problem

[3]The controversy was centred around the assumption that players a priori know the true distribution of types. (From a personal conversation with Reinhard Selten, a former collaborator and Nobel co-recipient of John Harsanyi, and major contributor to the theory of Bayesian games.) Note that we do not make such an assumption in our work; in fact, other players may not even use a type-based reasoning.

of ad hoc coordination in multiagent systems. Indeed, much of the work presented in this thesis is inspired by the body of work on Bayesian games, and this can be seen in the notation and language we use. Moreover, in Chapter 5, we show how the seminal result by Kalai and Lehrer (1993) can be extended to account for our work as well. Nonetheless, there are a number of important differences between Bayesian games and our work: Formally, the most crucial difference is that Bayesian games, in their original form, assume that the type spaces and type distribution are common knowledge, whereas we assume that they are unknown to us. Therefore, the principal body of this thesis is concerned with beliefs over hypothesised ("guessed") type spaces. Furthermore, much of the work on Bayesian games is focused on equilibrium attainment, and the usual assumption is that all players use the same type-based Bayesian reasoning. In contrast, we are interested in flexibility and efficiency, and we make no assumption about the behaviour of other agents (i.e. they may use an entirely different reasoning).

2.5 Interactive POMDPs

Interactive partially observable Markov decision processes (I-POMDP) (Gmytrasiewicz and Doshi, 2005) are related to stochastic Bayesian games (the model used in this thesis; see Chapter 3) in the sense that they, too, assume that agents have a latent model (or type). Technically, I-POMDPs extend the framework of POMDPs (Kaelbling et al., 1998) to the multiagent domain by incorporating the model spaces into the state space. Therefore, agents have to make decisions in the presence of uncertainty regarding both the state of the environment and the models of other agents. The models themselves are categorised into two groups: *sub-intentional* models, which can be any kinds of behaviours (similar to types in our work); and *intentional* models, which can be viewed as an I-POMDP of a lower recursive level. Hence, I-POMDPs explicitly consider recursive belief nesting of the form "I believe that you believe that I believe...".[4]

A number of contributions have been made to the theory and practice of I-POMDPs. For example, several solution techniques have been developed (e.g. Doshi et al., 2009; Doshi and Gmytrasiewicz, 2009; Doshi and Perez, 2008) and there are documented attempts to use I-POMDPs to model human reasoning (Doshi et al., 2010) and address money laundering (Ng et al., 2010). An interesting parallel to our work is that the convergence result of Kalai and Lehrer (1993) (cf. Chapter 5) has also been extended to

[4]While we do not explicitly model nested beliefs, they can be modelled *implicitly* within the types. Hence, our work also accounts for agents that reason about other agents' beliefs.

the framework of I-POMDPs (Doshi and Gmytrasiewicz, 2006). Another interesting connection is the use of behavioural equivalences (Rathnasabapathy et al., 2006), which is similar to our use of bisimulation equivalences (cf. Chapter 7).

However, despite these similarities, there are several notable differences: First of all, I-POMDPs assume partially observable states whereas we assume fully observable states. While this makes I-POMDPs more general, it can make their solutions very complex and expensive (e.g. Ng et al., 2010). Furthermore, I-POMDPs assume that agents have a fixed model throughout the interaction process, whereas we also allow for other dynamics such as mixed and correlated type distributions. However, perhaps the most important difference is in the focus of our work: I-POMDPs have primarily focused on the possibility of nested beliefs, which further complicates their solutions (Doshi and Gmytrasiewicz, 2009). In contrast, much of our work is focused on the possibility of *incorrect* hypothesised types. In this regard, we believe that many of the results reported in this thesis are also relevant to the framework of I-POMDPs.

2.6 Plan Recognition

A related area of research, commonly referred to as *plan recognition*, is concerned with inferring an agent's goals and intentions based on the agent's observed actions. The traditional methodology in plan recognition is to assume that the observed agent follows some recipe of actions, called a *plan*, and that the correct plan can be found in a given library of plans. One of the principal questions in plan recognition is how to structure such plans to facilitate efficient search and inference. For a slightly dated but nevertheless useful survey, we refer to (Carberry, 2001).

The notion of plans and plan libraries is similar to that of types and type spaces. However, plans often have intricate auxiliary structure, including properties to describe when actions can be applied and what effects they have, as well as temporal and causal orderings, etc. (e.g. Avrahami-Zilberbrand and Kaminka, 2007; Albrecht et al., 1998; Charniak and Goldman, 1993). In contrast to this, we define types as black-box functions and leave their precise structure open. Another central issue in plan recognition is the narrowing of competing plan hypotheses, using methods such as user dialogs, utility estimation, and various heuristics (Sukthankar et al., 2014). On the other hand, the selection (or mixing) of types is done via the posterior beliefs.

An interesting phenomenon which may complicate the search for the correct plan is the possibility that observed actions may have been taken with a certain intention

but led to an unintended result. On the other hand, some actions may be *unintentional* (e.g. Bonchek-Dokow et al., 2009) in the sense that they do not contribute towards achieving any goal. While important in the context of plan recognition, these issues are less relevant in the type-based methodology because types are not used to infer an agent's goals or intentions. However, they may become relevant if we consider the possibility that an adversarial agent might attempt to deceive us into believing that it is of a certain type. This possibility is not considered in the present thesis.

Plan recognition methods can be categorised into unobtrusive and obtrusive methods. In the first case, sometimes called *keyhole* recognition, the plan recogniser has no influence over the observed agent's actions. This is similar to the method described in Chapter 8, which performs automated statistical analyses to decide whether or not to reject a hypothesised type. This method may also be useful in the context of plan recognition, for the case in which plans allow for stochastic action choices.

2.7 Model Criticism & Checking

A potential concern in the type-based methodology is the fact that the hypothesised types may be incorrect, in the sense that they do not accurately describe the observed agent's behaviour. Therefore, much of our work is concerned with the implications of incorrect types and the possibility to reject types.

There exists a large body of literature on what is often referred to as *model criticism* (e.g. Bayarri and Berger, 2000; Meng, 1994; Rubin, 1984; Box, 1980). Model criticism attempts to answer the following question: given a data set D and model M, could D have been generated by M? This is analogous to our question, in which D is a sequence of observed actions of some agent and M is a hypothesised behaviour for that agent. However, in contrast to our work, model criticism usually assumes that the data are independent and identically distributed, which is not the case in the interactive setting we consider. In Chapter 8, we show how a particular form of model criticism, namely *frequentist hypothesis testing*, can be used to decide whether or not to trust our hypothesised types, or the combination of beliefs and types.

A related area of research is that of *model checking*, which attempts to verify that a given system (or model) satisfies certain formal properties (Clarke et al., 1999). For the case of stochastic processes, one particular form of model checking is given by the concept of *probabilistic bisimulation* (Baier, 1996; Larsen and Skou, 1991). Essentially, two stochastic processes are considered bisimilar if they can match each others tran-

sitions on average. In Chapter 7, we use the concept of probabilistic bisimulation to address the question of "incorrect" hypothesised types, showing that a certain form of optimality is preserved if a bisimulation relation exists.

Another related problem, sometimes referred to as *identity testing*, is to test if a given sequence of data was generated by some given stochastic process (Ryabko and Ryabko, 2008; Basawa and Scott, 1977). Instead of independent and identical distributions, this line of work assumes other properties such as stationarity and ergodicity. Unfortunately, these assumptions are also unlikely in interaction processes, and the proposed solutions are very costly. Model criticism and identity testing are not to be confused with *model selection*, in which two or more alternative models are under consideration (e.g. Vehtari and Ojanen, 2012). Similarly, we do not actively decide between different types when planning our own actions. Instead, actions are planned with respect to all types, weighted by the posterior beliefs ascribed to them. Therefore, our analysis of incorrect types in Chapter 7 is with respect to our beliefs over all types. However, the method in Chapter 8 can be applied to individual types or to the combination of beliefs and types.

2.8 Synonymous Terminology

As shown in this chapter, our work is at the intersection of a number of research areas. As a result, there exist different terminologies for essentially equivalent concepts. The following is a brief listing of terms with synonymous meaning in this thesis:

player, agent — Formally, we use the term "player" to denote a certain role or position in a game (e.g. Player 1 or "Goal Keeper") while the term "agent" is used to refer to an instance of a decision making algorithm which controls a player. Hence, we may say that Player 1 is controlled by an HBA agent. However, for all practical purposes, the terms may be used synonymously.

payoff, utility, reward — These terms all refer to the numerical signal received by the agent (or player) after taking an action. We normally prefer the term "payoff".

strategy, policy, model, and type — The terms "strategy", "policy", and "model" all refer to a function that governs a player's behaviour (i.e. how actions are chosen based on the current interaction history). Formally, a "type" is an abstract parameter which specifies a player's preferences (i.e. payoffs) and strategy. However, for convenience, we may use the term "type" to refer to a player's behaviour. Thus, when referring to behaviours, the four terms may be used synonymously.

Chapter 3

Model and Algorithm

This thesis studies the type-based method in the context of ad hoc coordination. As outlined in Chapter 1, the idea is to interact with previously unknown agents by comparing their observed actions with a set of hypothesised behaviours, called *types*. In this chapter, we will introduce formal definitions to make this idea precise. Specifically, we will address the following questions: How can we model types in interactive processes? What precisely does ad hoc coordination mean in this context? And, finally, how can we utilise types to address ad hoc coordination?

We begin by introducing a game-theoretic model called the *stochastic Bayesian game* (SBG). Therein, each player has one of a number of types which specify their preferences and behaviours, and the assignment of types is governed by a distribution over types. Given the players' actions, the game transitions between different states until some terminal state is reached. Based on this model, we give precise definitions of an agent's flexibility and efficiency, and we define the ad hoc coordination problem as designing an agent, called the *ad hoc agent*, which is able to optimise its flexibility and efficiency in a SBG in which the true type spaces and distribution are unknown.

Moreover, we derive a best-response rule called *Harsanyi-Bellman Ad Hoc Coordination* (HBA), which can be viewed as a canonical algorithmic description of the type-based method. HBA utilises a set of hypothesised types, where each type specifies a possible behaviour for an agent. Specifically, given the current interaction history, each type generates a probabilistic prediction of the agent's next action. By comparing these predictions with the actions taken by the agent, HBA computes a posterior belief to quantify the relative likelihood of types. The posterior belief and the type predictions are then used to make predictions about future trajectories in the interaction, based on which HBA computes an optimal response.

3.1 Stochastic Bayesian Games

As discussed earlier, we will use the notion of types in Bayesian games to formalise ad hoc coordination. However, in their original form (Harsanyi, 1967), Bayesian games are not descriptive enough to allow us to model the kinds of problems we are interested in, as they do neither include states nor time. Therefore, we combine Bayesian games with the concept of *stochastic games* (Shapley, 1953) to obtain a more descriptive model which we call the *stochastic Bayesian game*:

Definition 1. A *stochastic Bayesian game* (SBG) consists of:

- finite state space S with initial state $s^0 \in S$ and terminal states $\bar{S} \subset S$
- players $N = \{1, ..., n\}$ and for each $i \in N$:
  - finite set of actions A_i (where $A = A_1 \times ... \times A_n$)
  - infinite type space Θ_j (where $\Theta = \Theta_1 \times ... \times \Theta_n$)
  - payoff function $u_i : S \times A \times \Theta_i \to \mathbb{R}$
  - strategy $\pi_i : \mathbb{H} \times A_i \times \Theta_i \to [0,1]$
- state transition function $T : S \times A \times S \to [0,1]$
- type distribution $\Upsilon : \mathbb{N}_0 \times \Theta^+ \to [0,1]$, where Θ^+ is a finite subset of Θ.

$\mathbb{H}$ is the set of all *histories* $H^t = \langle s^0, a^0, s^1, a^1, ..., s^t \rangle$ with $t \geq 0$, such that $s^0, ..., s^t \in S$ and $a^0, ..., a^{t-1} \in A$. Hence, each player may choose actions based on entire histories.

A stochastic Bayesian game defines the interaction process as follows:

Definition 2. A SBG starts at time $t = 0$ in state s^0:

1. In state s^t, the types $\theta_1^t, ..., \theta_n^t$ are sampled from Θ^+ with probability $\Upsilon(t, \theta_1^t, ..., \theta_n^t)$, and each player i is informed only about its own type θ_i^t.
2. Based on the history H^t, each player i chooses an action $a_i^t \in A_i$ with probability $\pi_i(H^t, a_i^t, \theta_i^t)$, resulting in the joint action $a^t = (a_1^t, ..., a_n^t)$.
3. The game transitions into a successor state $s^{t+1} \in S$ with probability $T(s^t, a^t, s^{t+1})$, and each player i receives an individual payoff given by $u_i(s^t, a^t, \theta_i^t)$.

This process is repeated until a terminal state $s^t \in \bar{S}$ is reached, after which the game stops. We then say that the *task is solved*.

We also define several classes of type distributions:

Definition 3. A type distribution Υ is called *static* if $\forall t, \hat{t}\ \forall \theta \in \Theta^+ : \Upsilon(t,\theta) = \Upsilon(\hat{t},\theta)$. A type distribution which is not static is called *dynamic*.

Unless stated otherwise, we consider static type distributions by default. For convenience, we may write static type distributions as $\Upsilon(\theta)$.

Definition 4. A type distribution Υ is called *pure* if $\forall t\ \exists \theta \in \Theta^+ : \Upsilon(t,\theta) = 1$. A type distribution which is not pure is called *mixed*.

Our definition of types follows the original definition of Harsanyi (1967). This means that a type determines a player's payoffs and strategies (see also Dekel et al., 2004). However, since we define strategies with respect to a *history* of states and actions (rather than just the current state), a type may in fact specify strategies which change over time, and we thus also refer to it as *behaviour*. Therefore, our interpretation of types is that of a "programme" which governs the behaviour of a player.

In contrast to original Bayesian games, our type spaces Θ_i contain *all* possible behaviours for player i. This set is infinite and uncountable because the strategy π_i assigns probabilities to actions, and the interval $[0,1]$ is itself infinite and uncountable. Therefore, in order for Υ to be a well-defined probability distribution, we define it over a finite subset $\Theta^+ \subset \Theta$. To differentiate the two spaces, we sometimes refer to Θ_i as the full type space and to Θ_i^+ as the *true* types of player i. For convenience (and by abuse of notation), we will allow $\Upsilon(t,\theta)$ for any $\theta \in \Theta$, with $\Upsilon(t,\theta) = 0$ if $\theta \neq \Theta^+$.

3.2 Assumptions

Each player in the game may correspond to a specific role. For instance, if we model a soccer team, player 1 may correspond to the goal keeper. In the following sections, we implicitly assume that the ad hoc agent, denoted α, controls the player of interest, denoted i, by which we mean that α chooses the strategy π_i. Furthermore, i has a fixed type which is defined by α, hence we may write $u_i(s^t, a^t, \alpha)$ and $\pi_i(H^t, a_i, \alpha)$, or we may simply drop the α for compactness. Finally, we will use j and $-i$ to refer to the other players (e.g. $A_{-i} = \times_{j \neq i} A_j$).

Throughout this work, we will make the following general assumptions:

Assumption 1. We assume *full observability* of states and actions. That is, every player is always informed of the current history H^t before making a decision.

Assumption 2. For any type $\theta_j \in \Theta_j$ and history $H^t \in \mathbb{H}$, there exists a *unique* sequence $(\chi_{a_j})_{a_j \in A_j}$ such that $\pi_j(H^t, a_j, \theta_j) = \chi_{a_j}$ for all $a_j \in A_j$.

We refer to this as *external* randomisation and to the opposite (when there is no unique χ_{a_j}) as *internal* randomisation. Technically, Assumption 2 is implied by the fact that π_j is a function, which means that any input is mapped to exactly one output. However, in practice this can be violated if randomisation is used "inside" a type implementation, hence it is worth stating it explicitly. Nonetheless, it can be shown that under full observability, external randomisation is equivalent to internal randomisation. Hence, Assumption 2 does not limit the types we can represent.

Example 1. Let there be two actions, A and B, and let the expected payoffs for agent i be $E(A) > E(B)$. The agent uses ε-greedy action selection (Sutton and Barto, 1998) with $\varepsilon > 0$. If agent i randomises externally, then the strategy π_i will assign action probabilities $\langle 1-\varepsilon/2, \varepsilon/2 \rangle$. If the agent randomises internally, then with probability ε it will assign probabilities $\langle 0.5, 0.5 \rangle$ and with probability $1-\varepsilon$ it will assign $\langle 1, 0 \rangle$, which is equivalent to external randomisation.

3.3 Flexibility & Efficiency

Two important aspects of ad hoc coordination are *flexibility* and *efficiency*. We now define each of them formally within the model of stochastic Bayesian games. The definitions rely on the notion of *paths* and probabilities of paths. We will use the notation Γ_Υ to say that SBG Γ uses a specific type distribution Υ.

Definition 5. (i) A *path* ρ in SBG Γ is a sequence $\langle s^0_\rho, \theta^0_\rho, a^0_\rho, s^1_\rho, \theta^1_\rho, a^1_\rho, ..., s^{t_\rho}_\rho \rangle$ where $s^\tau_\rho \in S$, $\theta^\tau_\rho \in \Theta$, $a^\tau_\rho \in A$, and $s^0_\rho = s^0$.

(ii) A path ρ is *terminating* if $s^{t_\rho}_\rho \in \bar{S}$, otherwise it is *non-terminating*.

(iii) Given a type distribution Υ for Γ, the probability of path ρ in Γ_Υ is defined as

$$\Pr(\rho|\Gamma_\Upsilon) = \prod_{\tau=0}^{t_\rho - 1} \Upsilon(\tau, \theta^\tau_\rho)\, T(s^\tau_\rho, a^\tau_\rho, s^{\tau+1}_\rho) \prod_{k \in N} \pi_k(H^\tau_\rho, (a^\tau_\rho)_k, (\theta^\tau_\rho)_k) \tag{3.1}$$

where H^τ_ρ is the history extracted from ρ until time τ.

For $\Pr(\rho|\Gamma_\Upsilon)$ to be well-defined (that is, there exists a set X such that $\forall \rho \in X$: $\Pr(\rho|\Gamma_\Upsilon) \geq 0$ and $\sum_{\rho \in X} \Pr(\rho|\Gamma_\Upsilon) = 1$), it is important to note the following two implications in the definition of SBGs. Firstly, no path ρ can be prefixed by a terminating path, meaning that there is no $s^\tau_\rho \in \rho$ such that $\tau < t_\rho$ and $s^\tau_\rho \in \bar{S}$. This is important since

otherwise $\Pr(\rho|\Gamma_\Upsilon)$ might assign positive probability to a path which is prefixed by a terminating path and, thus, could never occur. Secondly, the only paths that can occur are either terminating (and hence finite) or non-terminating and *infinite* (i.e. $t \to \infty$). Thus, if Φ is the set of all terminating paths and Ψ the set of all infinite non-terminating paths, then $\sum_{\rho \in \Phi \cup \Psi} \Pr(\rho|\Gamma_\Upsilon) = 1$.

Based on the notion of paths, we define the flexibility and efficiency of ad hoc agent α as follows:

Definition 6. Let Φ be the set of all terminating paths in SBG Γ. Given a set of type distributions $\mathbb{T}$ for Γ, the *flexibility* $F(\alpha|\Gamma,\mathbb{T})$ and *efficiency* $E(\alpha|\Gamma,\mathbb{T})$ of agent α in Γ with respect to $\mathbb{T}$ are defined as

$$F(\alpha|\Gamma,\mathbb{T}) = \frac{1}{|\mathbb{T}|} \sum_{\Upsilon \in \mathbb{T}} \sum_{\rho \in \Phi} \Pr(\rho|\Gamma_\Upsilon) \tag{3.2}$$

$$E(\alpha|\Gamma,\mathbb{T}) = \frac{1}{|\mathbb{T}|} \sum_{\Upsilon \in \mathbb{T}} \sum_{\rho \in \Phi} \overline{\Pr}(\rho|\Gamma_\Upsilon) \frac{\left(\sum_{\tau=0}^{t_\rho - 1} u_i(s_\rho^\tau, a_\rho^\tau, \alpha)\right)^{r_1}}{(t_\rho)^{r_2}} \tag{3.3}$$

where $\overline{\Pr}(\rho|\Gamma_\Upsilon) = \frac{\Pr(\rho|\Gamma_\Upsilon)}{\sum_{\rho' \in \Phi} \Pr(\rho'|\Gamma_\Upsilon)}$, and $r_1, r_2 \geq 1$ specify the relative importance (that is, trade-off) between payoff and time.

$F(\alpha|\Gamma,\mathbb{T})$ and $E(\alpha|\Gamma,\mathbb{T})$ can be interpreted as, respectively, the average probability with which agent α solves the task in Γ and the average payoff per time step α received in solved tasks, where $\mathbb{T}$ specifies all possible type distributions (and, thereby, types that can occur). In the above definitions, every type distribution $\Upsilon \in \mathbb{T}$ is given equal weight in the average. If required, this could be further generalised by introducing specific weights for each type distribution. Finally, we note that there may be problems in which flexibility is not a relevant metric, e.g. because the game terminates after a fixed number of time steps. In such cases, the primary metric is efficiency.

3.4 The Ad Hoc Coordination Problem

We are now in a position to formally define the ad hoc coordination problem. The core aspect is that there is *no prior coordination* between the ad hoc agent and the other agents in the system. We express this formally in two variants of the problem:

Definition 7. Let Γ be a SBG with true type spaces Θ_j^+, and let $\mathbb{T}$ be a set of type distributions over Θ^+. Player i is controlled by ad hoc agent α.

Algorithm 1 Evaluation procedure to approximate flexibility and efficiency

Input: SBG Γ, set of type distributions $\mathbb{T}$, ad hoc agent α,
player i (to be controlled by α), $K \in \mathbb{N}^+$

Output: estimates of flexibility $F(\alpha|\Gamma,\mathbb{T})$ and efficiency $E(\alpha|\Gamma,\mathbb{T})$

$F \leftarrow 0$

$E \leftarrow 0$

Repeat K **times:**

Randomly draw type distribution $\Upsilon \in \mathbb{T}$

Generate path ρ in Γ_Υ (α controls i)

If ρ terminates **do**

$F \leftarrow F + 1$

$E \leftarrow E + \left(\sum_{\tau=0}^{t_\rho - 1} u_i(s_\rho^\tau, a_\rho^\tau, \alpha)\right)^{r_1} * (t_\rho)^{-r_2}$

$F(\alpha|\Gamma,\mathbb{T}) \leftarrow F/K$

$E(\alpha|\Gamma,\mathbb{T}) \leftarrow E/K$

(i) The *semi ad hoc coordination problem* is to optimise the flexibility $F(\alpha|\Gamma,\mathbb{T})$ and efficiency $E(\alpha|\Gamma,\mathbb{T})$ of α, subject to the constraint that α knows all elements of Γ (including Θ_j^+) except for the type distributions Υ used in Γ_Υ (for all $\Upsilon \in \mathbb{T}$).

(ii) The (full) *ad hoc coordination problem* is to optimise the flexibility $F(\alpha|\Gamma,\mathbb{T})$ and efficiency $E(\alpha|\Gamma,\mathbb{T})$ of α, subject to the constraint that α knows all elements of Γ except for Θ_j^+ and the type distributions Υ used in Γ_Υ (for all $\Upsilon \in \mathbb{T}$).

Computing $F(\alpha|\Gamma,\mathbb{T})$ and $E(\alpha|\Gamma,\mathbb{T})$ exactly is infeasible for all but the simplest games. We propose to approximate these by using the procedure given in Algorithm 1. The procedure generates K samples $F_k \sim F(\alpha|\Gamma,\mathbb{T})$ and $E_k \sim E(\alpha|\Gamma,\mathbb{T})$, based on which it approximates $F(\alpha|\Gamma,\mathbb{T}) = \frac{1}{K}\sum_k F_k$ and $E(\alpha|\Gamma,\mathbb{T}) = \frac{1}{K}\sum_k E_k$. Since all F_k and E_k, respectively, come from the same distribution, by the law of large numbers this will converge to the true values of $F(\alpha|\Gamma,\mathbb{T})$ and $E(\alpha|\Gamma,\mathbb{T})$ for $K \to \infty$. The procedure needs some means to determine if a path is non-terminating. This could be done, for instance, by checking if the path reached a state space which contains no terminal states and cannot be left anymore, or by setting a maximum path length after which a path is assumed to be non-terminating (as we do in Section 4.2).

3.5 Harsanyi-Bellman Ad Hoc Coordination

The problem of incomplete information is solved in Bayesian games by assuming that the (true) type spaces Θ_j^+ and type distribution Υ are common knowledge. This admits a solution in the form of the *Bayesian Nash equilibrium* (Harsanyi, 1968a), which we here define within the context of SBGs:

Definition 8. Let H^t be the history at time t. A *Bayesian Nash equilibrium* (BNE) in state s^t is a strategy profile $(\pi_1, ..., \pi_n)$ in which, for all $i \in N$ and $\theta_i \in \Theta_i^+$, π_i maximises

$$\sum_{\hat{\theta}_{-i} \in \Theta_{-i}^+} \Upsilon(t, \hat{\theta}_{-i} | \theta_i) \sum_{a \in A} u_i(s^t, a, \theta_i) \pi(H^t, a, (\theta_i, \hat{\theta}_{-i})) \tag{3.4}$$

where

$$\Upsilon(t, \theta_{-i} | \theta_i) = \frac{\Upsilon(t, (\theta_i, \theta_{-i}))}{\sum_{\hat{\theta}_{-i} \in \Theta_{-i}^+} \Upsilon(t, (\theta_i, \hat{\theta}_{-i}))} \tag{3.5}$$

$$\pi(H^t, a, \theta) = \prod_{k \in N} \pi_k(H^t, a_k, \theta_k). \tag{3.6}$$

In ad hoc coordination problems, the ad hoc agent does not know the type distribution Υ of the game. Therefore, it cannot compute $\Upsilon(t, \theta_{-i} | \theta_i)$. However, using the history H^t, it can compute a *posterior belief* $\Pr(\theta_{-i} | H^t) = \prod_{j \neq i} \Pr_j(\theta_j | H^t)$, where $\Pr_j(\theta_j | H^t)$ is the probability that player j has type θ_j based on history H^t

$$\Pr_j(\theta_j | H^t) = \frac{L(H^t | \theta_j) P_j(\theta_j)}{\sum_{\hat{\theta}_j \in \Theta_j^+} L(H^t | \hat{\theta}_j) P_j(\hat{\theta}_j)}, \tag{3.7}$$

$L(H^t | \theta_j)$ is the *likelihood* of history H^t if player j has type θ_j, and $P_j(\theta_j)$ is the agent's prior belief that player j has type θ_j.

The likelihood L specifies how evidence (i.e. observed actions) is incorporated to form the posterior belief. Note that, in contrast to probabilities, a likelihood is merely required to be non-negative but not necessarily to sum up to 1. We will look at several likelihood specifications in Chapters 4 and 5. For the purposes of this chapter, we define the default likelihood as

$$L(H^t | \theta_j) = \prod_{\tau=0}^{t-1} \pi_j(H^\tau, a_j^\tau, \theta_j). \tag{3.8}$$

Much work has been devoted to equilibrium analysis in games in which all players maintain posterior beliefs as defined above (cf. Section 2.4). For example, it has been

shown that a Nash equilibrium (NE) (Nash, 1950) can emerge under certain conditions (e.g. Kalai and Lehrer, 1993). However, while these are encouraging theoretical results, there are several potential objections concerning the use of equilibrium concepts such as NE: Firstly, if there are multiple NE, then the players may converge to a sub-optimal equilibrium. Secondly, a NE is incomplete in that it does not specify strategies for off-equilibrium paths. Moreover, Dekel et al. (2004) have shown that if the prior beliefs of the players are not identical, then they may converge to a solution which is not a NE. However, our main concern with NE is that it makes strong assumptions regarding the players' behaviours (such as perfect rationality and selfishness) which may be difficult to justify in ad hoc coordination. For instance, human players have been shown to deviate from such assumptions (e.g. Kahneman and Tversky, 1979). The same arguments hold for solution concepts in extensive form games, such as the perfect Bayesian equilibrium and sequential equilibrium (Fudenberg and Tirole, 1991a).

Rather than attempting to converge to a Nash equilibrium, it is appealing to use (3.4) as a *best-response* rule, since it maximises the expected payoff with respect to what types the ad hoc agent believes the other players to have and their strategies for all types. However, in its current form, (3.4) only considers immediate payoffs whereas optimal behaviour may require an agent to take payoffs of future states into account. Therefore, we propose to combine (3.4) with the Bellman optimality equation (Bellman, 1957) to obtain a best-response rule which we call *Harsanyi-Bellman Ad Hoc Coordination*[1], or HBA for short. Since full ad hoc coordination requires that the ad hoc agent does not know the true type spaces Θ_j^+, we assume instead that it has access to *hypothesised* (i.e. "guessed" by the user, hence also called *user-defined*) type spaces $\Theta_j^* \subset \Theta_j$. (For semi ad hoc coordination, we would then have $\Theta_j^* = \Theta_j^+$ or $\Theta_j^+ \subset \Theta_j^*$.)

Definition 9. Let Γ be an ad hoc coordination problem (full or semi) where ad hoc agent α controls player i and has access to hypothesised type spaces $\Theta_{-i}^* = \times_{j \neq i} \Theta_j^*$. *Harsanyi-Bellman Ad Hoc Coordination* (HBA) is defined as

$$a_i^t \sim \arg\max_{a_i} E_{s^t}^{a_i}(H^t) \tag{3.9}$$

where

$$E_s^{a_i}(\hat{H}) = \sum_{\theta_{-i}^* \in \Theta_{-i}^*} \Pr(\theta_{-i}^* | H^t) \sum_{a_{-i} \in A_{-i}} Q_s^{a_i,-i}(\hat{H}) \prod_{j \neq i} \pi_j(\hat{H}, a_j, \theta_j^*) \tag{3.10}$$

is the expected long-term payoff for player i of taking action a_i in state s after history $\hat{H}$

[1] The name is explained as follows: HBA is a combination of **H**arsanyi's Bayesian Nash equilibrium (3.10) and **B**ellman's optimality principle (3.11), applied to the problem of **A**d Hoc Coordination.

Algorithm 2 Harsanyi-Bellman Ad Hoc Coordination (HBA)

Input: SBG Γ, player i, hypothesised type spaces Θ_j^*,
history H^t, discount factor $0 \leq \gamma \leq 1$

Output: Action probabilities $\pi_i(H^t, a_i, \alpha)$

1. For each $j \neq i$ and $\theta_j^* \in \Theta_j^*$, compute posterior probability

$$\Pr_j(\theta_j^*|H^t) = \frac{L(H^t|\theta_j^*)\,P_j(\theta_j^*)}{\sum_{\hat{\theta}_j^* \in \Theta_j^*} L(H^t|\hat{\theta}_j^*)\,P_j(\hat{\theta}_j^*)}$$

2. For each $a_i \in A_i$, compute expected payoff $E_{s^t}^{a_i}(H^t)$ with

$$E_s^{a_i}(\hat{H}) = \sum_{\theta_{-i}^* \in \Theta_{-i}^*} \Pr(\theta_{-i}^*|H^t) \sum_{a_{-i} \in A_{-i}} Q_s^{a_{i,-i}}(\hat{H}) \prod_{j \neq i} \pi_j(\hat{H}, a_j, \theta_j^*)$$

$$Q_s^a(\hat{H}) = \sum_{s' \in S} T(s,a,s') \left[u_i(s,a,\alpha) + \gamma \max_{a_i} E_{s'}^{a_i}\big(\langle \hat{H}, a, s' \rangle\big) \right]$$

where $\Pr(\theta_{-i}^*|H^t) = \prod_{j \neq i} \Pr_j(\theta_j^*|H^t)$ and $a_{i,-i} \triangleq (a_i, a_{-i})$

3. Distribute $\pi_i(H^t, \cdot, \alpha)$ uniformly over $\arg\max_{a_i} E_{s^t}^{a_i}(H^t)$

$(a_{i,-i} \triangleq (a_i, a_{-i}))$, and

$$Q_s^a(\hat{H}) = \sum_{s' \in S} T(s,a,s') \left[u_i(s,a,\alpha) + \gamma \max_{a_i} E_{s'}^{a_i}\big(\langle \hat{H}, a, s' \rangle\big) \right] \tag{3.11}$$

is the expected long-term payoff for player i when joint action a is executed in state s after history $\hat{H}$, with $0 \leq \gamma \leq 1$ being the discount factor.

HBA is a modification of (3.4) which replaces $\Upsilon(t, \theta_{-i}|\theta_i)$ by the posterior $\Pr(\theta_{-i}|H^t)$ (3.7), and in which the immediate payoff u_i is replaced by an altered version (3.11) of the Bellman optimality equation. The actual history H^t is used to compute the posterior, and the projected histories $\hat{H}$ are used to generate all future trajectories. A summary of HBA in algorithmic form is given in Algorithm 2.

Intuitively, HBA chooses actions which maximise the expected long-term payoff with respect to what types HBA currently believes the other agents to have. It does so by expanding a tree of all possible future trajectories of the interaction. Each trajectory is weighted by the predictions of the hypothesised types as well as HBA's current beliefs regarding the relative likelihood of types. This allows HBA to compute expected

payoffs for each of its actions, by traversing the tree from the bottom (leaves) to the top (root). After the expected payoffs have been calculated, HBA chooses one of the actions with maximum expected payoff. Of course, in order for HBA to traverse the tree and calculate expected payoffs, the trajectories in the tree have to be finite. We will discuss two different ways to implement HBA in the next chapter.

Where do the hypothesised types $\theta_j^* \in \Theta_j^*$ come from? One way is to have them specified manually by domain experts, based on their experience with the problem. We chose this method in the case studies presented in Chapter 4. Another method to obtain hypothesised types is to generate them automatically from the problem description. In Chapters 6 and 8, we use three different methods to automatically generate sets of types for any given matrix game. (The methods are specified in Appendix B.) Finally, one may use machine learning methods such as the ones discussed in Section 2.3 to extract types from a corpus of historical data (e.g. Barrett et al., 2013; Gal et al., 2004).

Chapter 4

Two Empirical Case Studies

The previous chapter laid out the formal framework within which we conduct our studies. To further familiarise the reader with the concepts presented therein, this chapter will discuss two empirical case studies. The case studies show two different ways in which HBA can be implemented in practice. Moreover, they demonstrate that the idea of utilising types has great potential in addressing ad hoc coordination.

The first case study discusses results from a human-machine experiment conducted at the Royal Society Summer Science Exhibition 2012 in London. Therein, a large number of human subjects played one of two simple games, Rock-Paper-Scissors and Prisoner's Dilemma, against HBA and alternative algorithms. HBA was implemented as a simple forward-planning procedure which expands a full tree of all possible future trajectories (up to a fixed depth). The results show that a handful (5 and 6, respectively) of simple types were sufficient to achieve flexible and efficient interaction with humans which exhibited a large variety of behaviours. Specifically, HBA achieved the same individual average payoff as the alternative algorithms but outperformed them on important other metrics, such as social welfare and winning rate.

In the second case study, we used HBA to solve complex coordination problems in a simulated logistics domain. Here, a group of agents attempt to collect a number of objects in a two-dimensional grid-world. Agents may differ in their ability to collect objects, and their behaviours are initially unknown to us. We show how HBA can be implemented as a reinforcement learning procedure to solve such problems. The procedure uses a sampling-based planning routine to simulate interaction trajectories, based on the hypothesised types and posterior beliefs. The results show that HBA achieved significantly higher flexibility and efficiency than several alternative algorithms which were based on the same general reinforcement learning framework.

4.1 Human-Machine Experiment

In the first case study, we conducted a large human-machine experiment[1] at the Royal Society Summer Science Exhibition 2012[2] in London.

4.1.1 Experimental Setup

A large public exhibition such as this one is an excellent testbed environment for ad hoc agents, since the visitors vary widely in aspects such as intelligence and behaviour. However, in order to make statistically relevant comparisons, we required data from many participants. Therefore, the games needed to be simple enough so participants would understand them quickly, yet they also needed to be interesting in terms of coordination strategies. Prisoner's Dilemma (PD) and Rock-Paper-Scissors (RPS) are two widely-studied problems in game theory which have these properties (the payoff matrices are shown in Table 4.1). In PD, the problem is that the only Nash equilibrium (NE), and hence stable outcome, is at (D,D), while (C,C) is the only outcome that has both the highest welfare (sum of payoffs) and fairness (product of payoffs) but is unstable since the players could deviate to obtain higher immediate payoffs. In RPS, the only NE is for all players to play randomly. However, even if humans attempt to play randomly, they often fall back to patterns (Wagenaar, 1972) against which the other player can coordinate his or her action choices.

	C	D
C	3,3	0,5
D	5,0	1,1

(a) Prisoner's Dilemma

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

(b) Rock-Paper-Scissors

Table 4.1: Payoff matrices for Prisoner's Dilemma and Rock-Paper-Scissors games. The upper-case letters correspond to the actions available to the players. Each cell (u_r, u_c) contains the payoffs to the row player (u_r) and the column player (u_c) if they choose the corresponding actions. All payoffs are symmetric.

[1] Caveat: There is a long tradition of human experimentation in behavioural economics, with an established methodology and protocols regarding sampling, mental bias, control groups, etc. of human subjects. We realise that our experiment does not always follow such protocols and is perhaps somewhat unprincipled in this regard. Nonetheless, we do believe that interaction with humans is an important part of ad hoc coordination, and experiments such as this one constitute a step in this direction.

[2] http://sse.royalsociety.org/2012/exhibits/robotic-soccer

Algorithm 3 Generic framework for tree-based planning

Repeat:

Observe current state s^t

For all $a_i \in A_i$ **do**:

$$\Omega(a_i) = \left\{ \langle s^t, a^t, ..., s^{t+l}, a^{t+l} \rangle \,|\, a_i^t = a_i \right\} \text{ where } l = \min[l^*, t^* - t] - 1$$

$$E(a_i) = \sum_{\omega \in \Omega(a_i)} \left[\prod_{\tau=t}^{t+l} \textsc{OppStrat}(s^\tau, a^\tau) \sum_{\tau=t}^{t+l} u_i(s^\tau, a^\tau) \right]$$

Sample action $a_i^t \sim \arg\max_{a_i} E(a_i)$

The alternative algorithms were Joint Action Learning (JAL) (Claus and Boutilier, 1998) for RPS, and Conditional Joint Action Learning (CJAL) (Banerjee and Sen, 2007) for RPS. Both algorithms learn a model of the opponent's behaviour by taking the average frequency of the opponent's past actions (similar to "fictitious play"; Brown, 1951). The difference is that JAL conditions the model only on the current system state while CJAL conditions the model on the state *and* its own action choices. All algorithms, including HBA, were implemented using the same generic framework (Algorithm 3). The framework computes expected payoffs for all actions ($E(a_i)$) by expanding a finite tree of all future trajectories ($\Omega(a_i)$), where the trajectories are bounded in length by l^* and t^*. We set $l^* = 10$ for PD, $l^* = 1$ for RPS, and $t^* = 20$ (the number of played rounds) in both games. The function $\textsc{OppStrat}(s^\tau, a^\tau)$ returns the probability that players $j \neq i$ choose actions a_j^τ in state s^τ. HBA implements this by averaging over all hypothesised types in Θ_j^* using its current posterior, and C/JAL do this using their learned actions frequencies. While PD and RPS have no states, we found that the performance of C/JAL could be further improved by introducing "artificial" states, which we simply defined as $s^t = a^{t-1}$. (In the first round, C/JAL assumed the opponent to play randomly.)

Our hypothesis for the experiment was that the human would switch between several simple behaviours, as opposed to having one complex behaviour. Therefore, we modelled the problem as a SBG with a dynamic mixed type distribution (unknown to us) which governed the type of the human, and we provided HBA with a small set of types, specified in Tables 4.2 and 4.3, which we hypothesised the human could have. In order to recognise changing types more effectively, HBA used *temporally reweighted* posterior beliefs which we define in Section 4.1.2. Finally, HBA used uniform prior beliefs and a general time weight (cf. Section 4.1.2) with $a = 10$, $b = 0.05$, and $c = 3$.

PD type	**Definition**
AlwaysC	$a_i^t = C$
TitForTat	$a_i^0 = C,\ a_i^t = a_j^{t-1}$
TitFor2Tats	$a_i^{0,1} = C,\ a_i^t = C$ **if** $a_j^{t-1,t-2} = C$ **else** D
Optimistic	$\pi_i(C, H^t) = 1$ **if** $t < 2 \vee a_j^{t-1} = C \vee \mu = 0$ **else** $0.2 + 0.8\sigma$
Pessimistic	$\pi_i(D, H^t) = 1$ **if** $t < 2 \vee a_j^{t-1} = D$ **else** $0.2 + [\mu > 0]_1 0.8\sigma$
	where $\mu = \sum_{\tau=0}^{t-2} [a_i^\tau = C]_1$ and $\sigma = \frac{1}{\mu} \sum_{\tau=0}^{t-2} [a_i^\tau = a_j^{\tau+1} = C]_1$

Table 4.2: 5 PD types. $[b]_1 = 1$ if b is true, else 0.

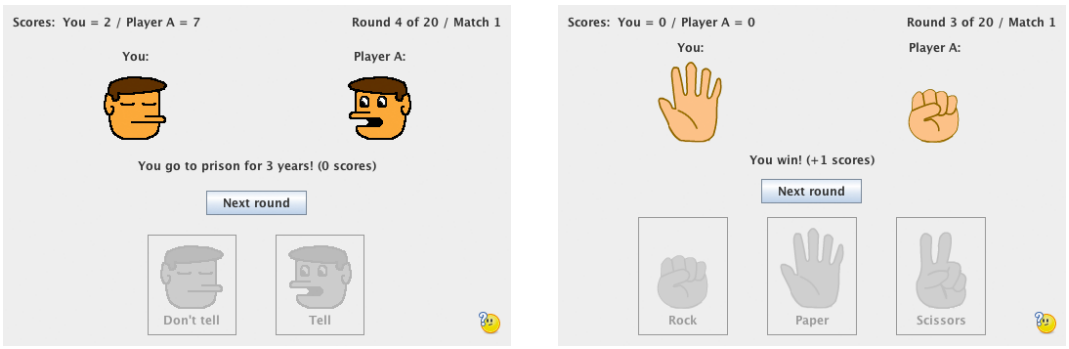
RPS type	**Definition**
Copycat	$a_i^0 \sim U(A_i),\ a_i^t = a_j^{t-1}$
RetryIfWon	$a_i^t \sim U(A_i)$ **if** $t = 0 \vee u_i(a^{t-1}) < 0$ **else** $a_i^t = a_i^{t-1}$
i-focused(h) for $h \in \{1,2\}$	$\pi_i(a_i, H^t) = g(a_i, x) / \sum_{\hat{a}_i \in A_i} g(\hat{a}_i, x),\ x = \min[t, h]$ where $g(a_i, x) = \max\big[0,\ x - \sum_{\tau=1}^{x} [a_i^{t-\tau} = a_i]_1 (x + 1 - \tau)\big]$
j-focused(h) for $h \in \{1,2\}$	$a_i^t \sim \arg\max_{a_i} \sum_{a_j \in A_j} \pi_j(a_j, H^t)\, u_i(a_i, a_j)$ where $\pi_j(a_j, H^t)$ is obtained using i-focused(h) for j

Table 4.3: 6 RPS types. U is uniform distribution. $[b]_1 = 1$ if b is true, else 0.

The procedure of the experiment was as follows: First, we randomly sampled a participant from the set of visitors which were currently at our exhibit (Figure 4.1). The participant was then brought to a dedicated table with a chair and a laptop computer. The laptop ran a programme, with an intuitive graphical user interface, which prompted the participant to choose between PD and RPS. The rules of the games were explained both textually in the programme and in person by one of our staff members to make sure the participant understood the rules. The game was then played in two matches, each lasting 20 rounds. One of the matches was against HBA and the other match against C/JAL, but this was hidden from the participant and the order was chosen randomly. The programme displayed the current match, round, and scores of all players, and also allowed to display the rules at any time (Figure 4.2). At the end of each round, the

(a) Our exhibit (b) Pool of test subjects

Figure 4.1: Royal Society Summer Science Exhibition 2012.



(a) Prisoner's Dilemma (b) Rock-Paper-Scissors

Figure 4.2: Graphical user interfaces.

participant was shown the actions and scores of both players, and at the end of each match, the participant was given a summary of the scores.

4.1.2 Temporally Reweighted Posteriors

A potential problem with the posterior defined by (3.7)/(3.8) is that it assigns zero probability to a type θ_j if $\pi_j(H^t, a_j^t, \theta_j)$ is zero for any t. This can be problematic for the following reasons: If the game uses a dynamic or mixed type distribution (as we assume in this experiment), and if $\Pr(\theta_j|H^t) = 0$ for a type θ_j that is not currently the true type of player j, then $\Pr(\theta_j|H^\tau) = 0$ for all times $\tau > t$, even if player j's type changes to θ_j. Furthermore, if we have a hypothesised type θ_j^* which approximates the true type θ_j of player j in a subset $S^* \subset S$ (i.e. $\pi_j(H^t, a_j, \theta_j^*) \approx \pi_j(H^t, a_j, \theta_j)$ for $s^t \in S^*$), but not outside S^*, then (3.7)/(3.8) might assign zero probability to θ_j^* once player j leaves S^*. However, θ_j^* may be the best approximation we have for S^*, so it would be useful if

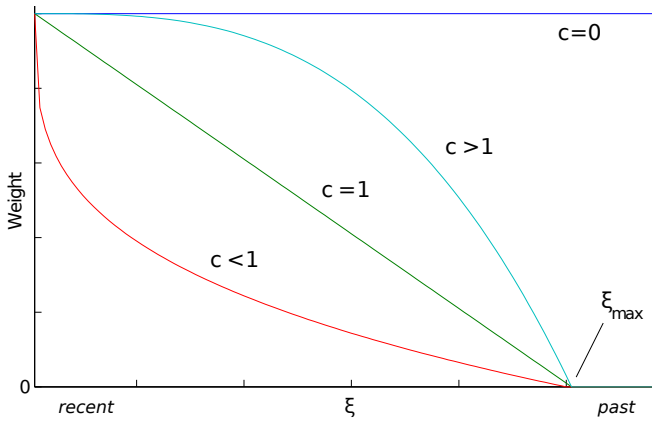


Figure 4.3: General time weight. The y-axis shows the weight (upward direction is higher weight) and the x-axis shows the time, with the most recent events on the left-hand side. Different settings of the parameter c produce different weighting schemes (parameters a and b are the same in all graphs). For $c \neq 0$, all past events after $\xi_{\max}$ are assigned zero weight and are hence ignored.

(3.7) was able to quickly reassign positive probability to θ_j^* once player j returns to S^*. To address these problems, we use *temporally reweighted posteriors*:

Definition 10. A *temporally reweighted posterior* (TR-posterior) is defined as in (3.7) where the likelihood L is defined as

$$L(H^t|\theta_j) = \sum_{\tau=0}^{t-1} f(t-\tau)\,\pi_j(H^\tau, a_j^\tau, \theta_j) \tag{4.1}$$

where $f(\xi) \geq 0$ and $f(\xi) \geq f(\xi+1)$, for all $\xi \in \mathbb{N}^+$.

The function f is called the *time weight* and can assume various forms. An example of a simple but useful time weight, called the *general time weight*, is given by $f(\xi) = \max[0, a - b\,(\xi-1)^c]$ where $a, b, c \in \mathbb{R}_0^+$. This time weight can be used to produce various behaviours, depending on the parameters a, b, c (Figure 4.3). In particular, it can be used to give greater importance to more recent events, which means that HBA is able to quickly reassign probabilities. However, the crucial aspect of (4.1) is that it defines a sum rather than a product, which means that the problems described above do not occur.

4.1.3 Results

We collected data from 427 participants, of which 186 played PD and 241 played RPS. The lowest and highest recorded ages were 9 and 72, respectively, with an average age of approximately 17. In the following, all significance statements are based on

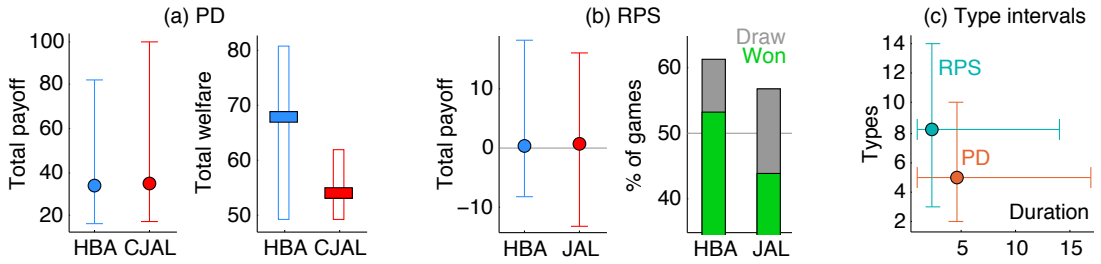


Figure 4.4: Results of the human-machine experiment. Circles and whiskers correspond to mean, minimum, and maximum values, respectively. The welfare plot in (a) shows the median value and 25%/75% percentiles.

paired t-tests with 5% significance level. Figures 4.4a and 4.4b show the results for PD and RPS, respectively. In both games, the average total payoffs of HBA and C/JAL were statistically equivalent. Since the time was fixed to 20 rounds, it means that they achieved equal efficiency. This is, in fact, a positive result considering that C/JAL are strong candidates in PD/RPS. In addition, as we will discuss in the following, HBA behaved very differently from C/JAL, with beneficial side effects.

In PD, the most desirable long-term outcome is (C,C) since it is both welfare and fairness optimal, and since it is a non-myopic equilibrium (Brams, 1993), meaning that no player has a long-term incentive to deviate. With this in mind, we point out that in over 28% of the games, HBA and the human played (C,C) in at least 50% of the final 10 rounds of the game, while CJAL did not achieve this in any game. Thus, HBA achieved a significantly higher total welfare than CJAL (Figure 4.4a). This is despite the fact that neither of them was optimised for social welfare. The reason for this is that HBA was planning more accurately than CJAL. When computing the expected payoffs $E(a_i)$, CJAL uses its learned action frequencies to obtain probabilities for each trajectory in $\Omega(a_i)$. However, these probabilities can only be accurate for states that have been visited frequently enough. Moreover, if a player changes its behaviour, CJAL requires new evidence from all states to accurately reflect the change. On the other hand, HBA uses its posterior and types to compute probabilities of trajectories. Therefore, once HBA has an accurate posterior, it can use the types to accurately plan in the entire state space of the game, including unseen states. This also allows HBA to plan the effects of its actions on the other player, which means that HBA may take actions to manipulate the player's decisions. Finally, if a player changes its behaviour, HBA only needs to update its posterior, which requires much less information than the update in CJAL.

In RPS, the crucial questions is whether a player is winning or not. Interestingly,

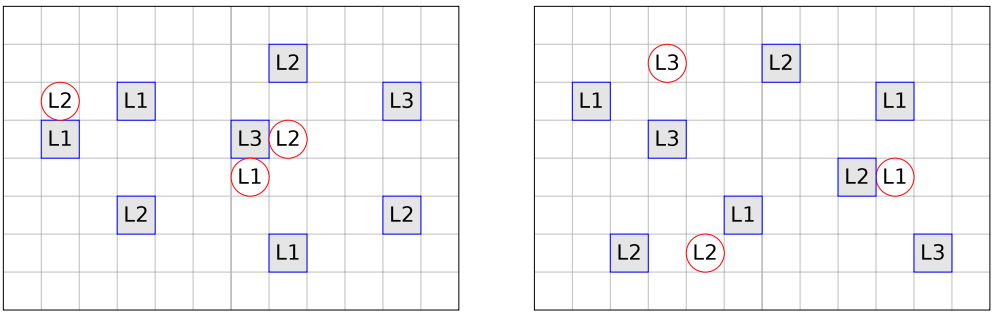
the winning rate of HBA (53.71%) was significantly higher than the winning rate of JAL (43.98%), as shown in Figure 4.4b. While in PD the good performance of HBA was due to its planning capabilities, in RPS this was not as relevant since the planning horizon was limited to trajectories of length 1. Rather, HBA's good performance was due to the fact that it recognised changed behaviours faster than JAL. Indeed, in a game such as RPS, it can be expected that the human players change frequently between different strategies. This is confirmed by the statistics shown in Figure 4.4c, which show the average number of types used by the human players and the average duration. The statistics are based on HBA's posterior beliefs, where the number of types for player i in a play corresponds to the number q in $\langle t_0, t_1, ..., t_q \rangle$, with $t_0 = 0$ and $t_q = 20$, for which $\arg\max_{\theta_i} \Pr(\theta_i | H^{\tau}) \subseteq \arg\max_{\theta_i} \Pr(\theta_i | H^{\tau+1})$ for all $t_{y-1} \leq \tau < t_y$ and $y \in \{1, ..., q\}$, and where the average duration is $\frac{1}{q} \sum_y t_y - t_{y-1}$. According to these statistics, the human players had 4.45 types with a duration of 4.96 rounds in PD, and 8.25 types with a duration of 2.46 rounds in RPS. Clearly, with a duration of only 2.46 rounds, planning was not as important as recognising changed types. By using TR-posteriors, HBA was able to do this effectively.

4.2 Simulated Experiments

In the second case study, we evaluated several configurations of HBA and alternative algorithms in a complex logistics domain called *level-based foraging*.

4.2.1 Experimental Setup

A level-based foraging problem consists of a rectangular grid with n players and m foods (Figure 4.5). Each field in the grid is either empty or occupied by one player or one food. All players and foods have a *level* ($\in \mathbb{N}^+$) where no food has a level greater than the sum of any 4 players' levels. A player can choose among 5 actions: N, E, S, W, and $load$. The first 4 actions move the player into the corresponding direction if the field is empty and inside the grid. A group of 1 to 4 players can *load* a food if they are placed on fields next to the food and if the sum of their levels is at least as high as the food's level. A player which successfully loads a food obtains a payoff equal to the level of the loaded food. At all other times, it receives a negative payoff of -0.01. To avoid conflicts and keep this solvable, the foods are placed such that the Euclidean distance between each of them is greater than 1, and no food is placed at any border



(a) Every player can load a food

(b) No player can load a food

Figure 4.5: Level-based foraging domain. Players are marked by circles and foods are marked by squares (the respective levels are shown inside).

of the grid. The players' goal is to collect all foods in minimal time, while also trying to maximise their own payoffs. Since the players have different abilities (i.e. levels) and are spatially distributed, this requires strong coordination of their behaviours. For simplicity, we assume that the ad hoc agent knows the levels of all players and foods.

We specified 6 classes of types. The first 4 classes contain types with fixed behaviours (i.e. they do not change over time). They each have a parameter σ which specifies the radius of their sight: H1 always goes to the closest visible food. H2 goes to the one visible food which is closest to the centre of all visible players. H3 always goes to the closest visible food with compatible level (i.e. it can load it) and H4 goes to the one visible food which is closest to all visible players such that the sum of their and H4's level is sufficient to load the food. H1–H4 try to load the food once they are next to it. If they do not see a food, they move into a random direction. The final two classes specify types with learning behaviours: Class 5 contains all instances of JAL and class 6 all instances of CJAL, as specified in the next paragraph.

We evaluated various configurations of HBA and three alternative algorithms: JAL (Claus and Boutilier, 1998) learns the action frequencies of each player in each state and uses them to compute expected action payoffs; CJAL (Banerjee and Sen, 2007) is similar to JAL but learns the frequencies conditioned on its own actions; WoLF-PHC (Bowling and Veloso, 2002) is a hill-climbing method in the space of mixed strategies. The algorithms have been shown to exhibit different behaviours in ad hoc coordination problems (Albrecht and Ramamoorthy, 2012).

A single framework (Algorithm 4) was used to implement each algorithm. The framework uses a table Q to learn the expected long-term payoffs of joint actions, similar to Q-learning (Watkins and Dayan, 1992). To accelerate learning, it uses an

Algorithm 4 Reinforcement learning framework with sampling-based planning

Set $Q(s,a) \leftarrow 0$ and $e(s,a) \leftarrow 0$ for all $(s,a) \in S \times A$

Repeat until $s^t \in \bar{S}$:

Observe: current state s^t

With probability $1-\varepsilon_1$ set $a_i^t = \text{CHOOSEACTION}(s^t)$, else sample $a_i^t \sim A_i$

Observe: joint action a^t, own payoff u_i^t, next state s^{t+1}

$\text{UPDATEQ}(s^t, a^t, u_i^t, s^{t+1}, e)$

Repeat x times: $\text{EXPAND}(d, s^{t+1}, \text{COPY}(e))$

$\text{EXPAND}(d, s, \hat{e})$:

Repeat d times or until $s \in \bar{S}$:

With probability $1-\varepsilon_2$ set $a_i = \text{CHOOSEACTION}(s)$, else sample $a_i \sim A_i$

$a_{-i} \leftarrow \text{OPPACTIONS}(s)$

$(u_i, s') \leftarrow \text{SIMULATE}(s, (a_i, a_{-i}))$

$\text{UPDATEQ}(s, (a_i, a_{-i}), u_i, s', \hat{e})$

$s \leftarrow s'$

$\text{UPDATEQ}(s, a, u, s', \hat{e})$:

$\delta = \beta\,(u + \gamma \max_{\hat{a}_i} \text{EXPPAY}(Q, s', \hat{a}_i) - Q(s,a))$

$\hat{e}(s,a) \leftarrow 1$

For all $(\hat{s}, \hat{a}) \in S \times A$ **s.t.** $\hat{e}(\hat{s}, \hat{a}) \geq e_{min}$ **do:**

$Q(\hat{s}, \hat{a}) \leftarrow Q(\hat{s}, \hat{a}) + \delta \hat{e}(\hat{s}, \hat{a})$

$\hat{e}(\hat{s}, \hat{a}) \leftarrow \lambda \hat{e}(\hat{s}, \hat{a})$

$\text{CHOOSEACTION}(s)$:

Return $a_i \sim \arg\max_{\hat{a}_i} \text{EXPPAY}(Q, s, \hat{a}_i)$

eligibility trace e (Sutton and Barto, 1998) to connect current payoffs with past actions. We assume that the agent has access to a simulator $\text{SIMULATE}(s,a)$ which, based on the transition (T) and payoff (u_i) functions of the game, returns a successor state s' and payoff u after taking joint action a in state s. This simulator is used in a sampling-based planning procedure (Kearns et al., 2002) $\text{EXPAND}(d, s, \hat{e})$ which, starting in state s, generates a future trajectory of length d and updates Q using the eligibility trace $\hat{e}$. The function $\text{EXPPAY}(Q, s, a_i)$ computes the expected payoff for taking action a_i in state s based on Q, and the function $\text{OPPACTIONS}(s)$ samples actions for all other players $j \neq i$ in state s. HBA implements EXPAND and OPPACTIONS using its posterior beliefs and hypothesised types. C/JAL implement these functions using their learned action

frequencies. For WoLF-PHC, the framework defines Q and e on $S \times A_i$ (rather than $S \times A$) and EXPPAY(Q,s,a_i) is simply defined as $Q(s,a_i)$. Since WoLF-PHC does not model its opponents, we implement OPPACTIONS the same way as in JAL. The function CHOOSEACTION(s) is redefined to $a_i \sim \pi(s)$, where π is the mixed strategy maintained in WoLF-PHC (cf. Tables 5 and 6 in Bowling and Veloso, 2002).

All algorithms used identical parameters: $\beta = .2$, $\gamma = .9$, $\lambda = .9$, $e_{min} = .01$, $\varepsilon_1 = 0$, $\varepsilon_2 = .2$, $x = 3$, $d = 20$. For WoLF-PHC, we used learning rates $\delta_w(t) = (1000 + \frac{t}{10})^{-1}$ and $\delta_l(t) = 2\delta_w(t)$. For HBA, we used uniform prior beliefs $P(\theta_j^*) = |\Theta_j^*|^{-1}$. To allow HBA to recognise changing types, we used a general time weight with $a = 10$, $b = .01$, $c = 3$ (cf. Section 4.1.2). In addition, to allow HBA to learn new types during the interaction, we used *conceptual types* as defined in Section 4.2.2. Estimates of flexibility and efficiency were computed using Algorithm 1 with $i = 1$, $r_1 = r_2 = 1$, $K = 1000$, where we assumed a path to be non-terminating if it reached $t = 1000$. The initial states in the foraging domain were generated with random positions and levels for all players and foods, with the maximum level being equal to the number of players. All agents were tested on the same sequence of games and random numbers.

4.2.2 Conceptual Types

If the hypothesised type space Θ_j^* for player j does not include the true type space Θ_j^+ (i.e. $\Theta_j^+ \not\subset \Theta_j^*$), then j might have a type which is unknown to HBA, causing its expected payoffs to be inaccurate. In such cases, it would be useful if HBA was able to learn new types from experience. A useful feature of HBA is that it can include methods for opponent modelling (see Section 2.3) as a special kind of type in Θ_j^*.

In this experiment, we use a combination of case-based reasoning (Gilboa and Schmeidler, 2001) and fictitious play (Brown, 1951), called *conceptual types*. Conceptual types are based on the observation that behaviours may not be specified on a state-by-state basis but rather on abstractions of state spaces. That is, there may be some *world conceptualisation* inherent in a behaviour. (Example of this are "information sets" in extensive form games (Fudenberg and Tirole, 1991b) and "feature vectors" in reinforcement learning (Sutton and Barto, 1998).) While the types in Θ_j^* are used to hypothesise behaviours directly, a conceptual type can be used to hypothesise a world conceptualisation underlying a player's behaviour. Combined with the observed actions of the player, this can be used to make predictions for previously unseen states and increase prediction accuracy for rarely visited states.

Definition 11. A *conceptual type* (c-type) θ_j^c for player j is a tuple (d_j, r, f), where $d_j : S \times S \to \mathbb{R}_0^+$ is a symmetric distance function for pairs of states, $r \in \mathbb{R}^+$ is a radius, and f is a time weight (as defined in Section 4.1.2), with

$$\pi_j(H^t, a_j, \theta_j^c) = \begin{cases} |A_j|^{-1} \text{ if } \nexists \tau < t : g(s^t, s^\tau) > 0 \text{ else} \\ \qquad \eta \sum_{a^\tau \in H^t : a_j^\tau = a_j} f(t-\tau)\, g(s^t, s^\tau) \end{cases} \tag{4.2}$$

where $g(s_1, s_2) = \max\left[0, 1 - d_j(s_1, s_2)\, r^{-1}\right]$ and η is a normalisation constant such that $\sum_{a_j} \pi_j(H^t, a_j, \theta_j^c) = 1$.

The function g is the hypothesised world conceptualisation for player j and measures how similar two states are from the perspective of player j (determined by d_j and r; examples are given in Section 4.2.3). The time weight f can be used to give greater importance to recent events, which allow c-types to adapt quickly to changing behaviours. Intuitively, $\pi_j(H^t, a_j, \theta_j^c)$ is the normalised sum of how often player j chose action a_j in states similar to the current state s^t (weighted by g) and how recent those action choices were (weighted by f). If no similar states have been encountered before, then $\pi_j(H^t, a_j, \theta_j^c)$ prescribes a uniform distribution. Note that we can include multiple c-types in Θ_j^*, each corresponding to a different world conceptualisation, and the posterior $\Pr_j$ filters out those types which do not fit the observed behaviour.

4.2.3 Results

We first tested the effectiveness of TR-posteriors by simulating the two problem situations described in Section 4.1.2. All tests were run on a 8×8 grid with 2 players and 5 foods. In Figure 4.6a, we used $\Theta_2^+ = \Theta_2^* = \{\text{H1–H4} \,|\, \sigma = \infty\}$ and a dynamic pure type distribution which changed the type of player 2 after every 10 to 20 time steps. In Figure 4.6b, we used $\Theta_2^+ = \{\text{H1–H4} \,|\, \sigma = 3, 5, 7\}$, $\Theta_2^* = \{\text{H1–H4} \,|\, \sigma = \infty\}$ (i.e the types in Θ_2^* were accurate only for subsets $S^* \subset S$) and a static pure type distribution. In both cases, the efficiency of HBA was significantly higher when using a TR-posterior with general time weight (Gtw) compared to both the default posterior defined in (3.7)/(3.8) (Unl) and a default posterior which was limited to the 9 most recent events (Lim), which is the same time frame used in Gtw. In Figure 4.6a, Gtw even achieved the same efficiency as a version of HBA which always knew the correct type of the other player (Cor). All HBA agents achieved a perfect flexibility of 1.

We also tested HBA with 4 conceptual types $\theta_j^c = (d_j^c, r, f)$ where $f(\xi) = [\xi < 10]_1$ and $r = 1$. In the following, we write $s.p_j$ ($s.f_k$) to refer to the position of player j (food

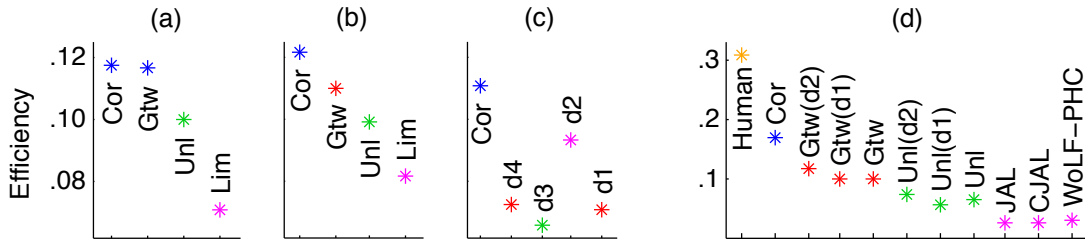


Figure 4.6: Results of simulated experiments, averaged over 1000 runs. Markers have the same colour if the difference is statistically insignificant (based on paired t-test with 5% significance level). "Cor" is HBA with correct types, "Gtw" is HBA using TR-posterior with general time weight, "Unl" is HBA with unlimited default posterior, and "Lim" is HBA with default posterior limited to 9 most recent events.

f_k) in state s, and we write $f_k \in s$ to say that food f_k is available in state s. The distance functions d_j^c are defined as

$$d_j^1(s_1,s_2) = [s_1 \neq s_2]_1 \infty \tag{4.3}$$

$$d_j^2(s_1,s_2) = [s_1.p_j \neq s_2.p_j \vee \neg\forall k : f_k \in s_1 \Leftrightarrow f_k \in s_2]_1 \infty \tag{4.4}$$

$$d_j^3(s_1,s_2) = \phi(s_1.p_j, s_2.p_j) + \sum_{k:\ f_k \in s_1 \ \veebar\ f_k \in s_2} \|s_1.f_k, \mu\|^{-\frac{3}{2}} \tag{4.5}$$

$$d_j^4(s_1,s_2) = d_j^3(s_1,s_2) + \sum_{v \neq j} \phi(s_1.p_v, s_2.p_v)\, \omega_v^{-\frac{3}{2}} \tag{4.6}$$

where

$$\phi(x_1,x_2) = \log(1+\|x_1,x_2\|)\frac{1}{2} \tag{4.7}$$

$$\mu = s_1.p_j + \frac{1}{2}(s_2.p_j - s_1.p_j) \tag{4.8}$$

$$\omega_v = \min[\|s_1.p_v, \mu\|, \|s_2.p_v, \mu\|] \tag{4.9}$$

$\|x_1,x_2\|$ denotes the Euclidean distance between x_1 and x_2, and $\veebar$ denotes the "exclusive or" operator. Intuitively, d_j^1 uses no abstraction; d_j^2 specifies that two states are equivalent if player j has the same position in both states and if all foods have the same status in both states; d_j^3 specifies that two states are more similar the closer player j's position in the states and the further a food is away from j if its status differs in the states; d_j^4 is similar to d_j^3 but also takes into account the other players' positions in the states. All tests were run on a 8×8 grid with 2 players and 5 foods, using $\Theta_2^+ = \{\text{H1–H4,JAL,CJAL} \,|\, \sigma = \infty\}$

(C/JAL used same parameters as HBA), $\Theta_2^* = \{\theta_2^c\}$ (each $c = 1,...,4$ tested separately), and a static pure type distribution. The results in Figure 4.6c show that HBA achieved good efficiency (compared to Cor) using θ_j^2, while the other c-types were less efficient. All HBA agents achieved statistically equivalent flexibilities of 0.86 ± 0.01.

Finally, we tested HBA, JAL, CJAL, and WoLF-PHC on a 10×10 grid with 3 players and 8 foods, using $\Theta_{2,3}^+ = \{\text{H1–H4,JAL,CJAL} \,|\, \sigma = 5,7,9\}$ and $\Theta_{2,3}^* = \{\text{H1–H4} \,|\, \sigma = \infty\}$. To add more realism, players 2 and 3 were "defective" with probability 0.2, where a defective player changed its type randomly every 10 to 30 time steps. While the potential of HBA is demonstrated by Cor, it would also be useful to know the optimal solution to the problem. However, with a complex problem such as this one, we were unable to compute optimal solutions. Instead, we had 6 *humans* play the game in a graphical user interface (each one played the full 1000 runs, distributed over 7 days at their own convenience), where no human was familiar with the technical details of this work. We do not necessarily claim that humans produce optimal solutions, but we expect them to perform consistently well in this setting. To cope with the increased problem size, we set the planning power of the algorithms to $x = 10$ and $d = 30$ (cf. Algorithm 4).

The results (Figure 4.6d) show that HBA clearly outperformed all alternative algorithms, with Unl and Gtw being over 100% and 200% more efficient, respectively. This is despite the fact that the hypothesised types $\Theta_{2,3}^*$ did not include any true types of the players. We also tested HBA with the c-types θ_j^1 and θ_j^3 (added separately to $\Theta_{2,3}^*$) but found that the efficiency of HBA did not improve significantly. This is since C/JAL learned similar behaviours to H1 and H3, which were already covered in $\Theta_{2,3}^*$. We found that HBA's posterior beliefs often assigned high probabilities to H1/3 when the true type of the player was in fact C/JAL. Since H1/3 ignore other players, this means that C/JAL did not effectively coordinate their behaviours with the other players. We found similar results for WoLF-PHC. As was expected, the humans achieved high efficiency (Figure 4.6d shows the best human) and outperformed even Cor. One reason for this is the fact that the humans had much greater planning power than HBA. Lastly, HBA achieved higher flexibilities ($.83 \pm .01$) than JAL (.734), CJAL (.749), and WoLF-PHC (.744), while the humans all achieved perfect flexibility (1.0).

Chapter 5

Correctness of Posterior Beliefs

A central aspect of the type-based methodology are the beliefs over types. Beginning with some initial beliefs as to the relative likelihood of types, we compare the predictions of types with the observed actions and update our beliefs to reflect the given evidence. Associated with this process are a number of key questions. In particular, how may evidence be incorporated into beliefs? And under what conditions will the beliefs be correct? These questions are crucial since incorrect beliefs may lead to wrong predictions and, thereby, to suboptimal actions.

This chapter addresses both questions. We consider three classes of type distributions to cover a broad spectrum of scenarios: pure distributions, in which all agents have a fixed type; mixed distributions, in which types are randomly re-allocated; and correlated distributions, in which type assignments may be correlated. Corresponding to these classes, we consider three formulations of posterior beliefs, called the product, sum, and correlated posteriors. Each formulation prescribes a different way to incorporate evidence into beliefs. We provide theoretical conditions under which these formulations produce *correct* beliefs, by which we mean that the beliefs converge to the true distribution of types. In addition, we provide examples to show when they may fail. These insights can be used to choose an appropriate posterior formulation.

Note that the results presented in this chapter pertain to *semi* ad hoc coordination, in which the true types are a subset of the hypothesised types (see Chapter 3). This is necessary because our definition of "correct" is with respect to the type distribution: beliefs are said to be correct if they assign the same probabilities to true types as the type distribution. Therefore, beliefs can only be correct if they can point to the types in the support of the type distribution. The case in which beliefs cannot be correct, due to incomplete or incorrect hypothesised types, is examined in Chapters 7 and 8.

5.1 Preliminaries

This chapter is concerned with convergence and correctness of posterior beliefs. Recall from Chapter 3 that the posterior belief that player j is of type θ_j^*, given history H^t, is defined as

$$\Pr_j(\theta_j^*|H^t) = \frac{L(H^t|\theta_j^*)\,P_j(\theta_j^*)}{\sum_{\hat{\theta}_j^* \in \Theta_j^*} L(H^t|\hat{\theta}_j^*)\,P_j(\hat{\theta}_j^*)} \tag{5.1}$$

where $L(H^t|\theta_j^*)$ is the likelihood of history H^t if player j has type θ_j^* (to be specified in the following sections), and P_j is the prior belief that player j has type θ_j^*. Unless explicitly stated otherwise in the theorems, prior beliefs may assign arbitrary probabilities. The combined posterior Pr is defined as $\Pr(\theta_{-i}^*|H^t) = \prod_{j\neq i}\Pr_j(\theta_j^*|H^t)$.

The theorems in this chapter tell us if and under what conditions HBA will learn the type distribution Υ of the game. As noted earlier, for this to be a well-posed learning problem, we have to assume that the posterior Pr can refer to the same elements as the type distribution Υ. Therefore, the results in this chapter pertain to *semi* ad hoc coordination (cf. Section 3.4), in which the user knows that the true type space Θ_j^+ is a subset of (or equal to) the hypothesised type space Θ_j^*. Formally, we assume:

Assumption 3. $\forall j \neq i : \Theta_j^+ \subseteq \Theta_j^*$

Finally, our analysis focuses on *static* type distributions, which we defined in Section 3.1 as being independent of time. Recall that a type distribution is called *pure* if it assigns all probability mass to a single type, otherwise it is called *mixed*.

5.2 Product Posterior

We begin our analysis with the product posterior:

Definition 12. The *product posterior* is defined as (5.1) with

$$L(H^t|\theta_j^*) = \prod_{\tau=0}^{t-1} \pi_j(H^\tau, a_j^\tau, \theta_j^*). \tag{5.2}$$

This is the standard posterior formulation used in Bayesian games (e.g. Dekel et al., 2004; Kalai and Lehrer, 1993). Furthermore, it was used in Chapters 3 and 4 as the "default" posterior formulation.

It can be shown that, under a pure type distribution and if HBA does not a priori rule out any of the types in Θ_j^*, then it will learn to make correct future predictions.

Let H^∞ be an infinite history with prefix H^τ, and denote by $P_\Upsilon(H^\tau,H^\infty)$ and $P_{\Pr}(H^\tau,H^\infty)$, respectively, the *true* probability (based on Υ) and the probability assigned by HBA (based on Pr) that H^τ will continue as prescribed by H^∞.

Theorem 1. Let Γ be a SBG with a pure type distribution Υ. If HBA uses a product posterior and if the prior probabilities P_j are positive (i.e. $\forall\theta_j^* \in \Theta_j^* : P_j(\theta_j^*) > 0$), then: for any $\varepsilon > 0$, there is a time t from which ($\tau \geq t$)

$$P_{\Pr}(H^\tau,H^\infty)(1-\varepsilon) \;\leq\; P_\Upsilon(H^\tau,H^\infty) \;\leq\; P_{\Pr}(H^\tau,H^\infty)(1+\varepsilon) \tag{5.3}$$

for all H^∞ with $P_\Upsilon(H^\tau,H^\infty) > 0$.

Proof. Kalai and Lehrer (1993) studied a model which can be equivalently described as a single-state SBG (i.e. $|S| = 1$) with a pure type distribution and product posterior. They showed that, if the player's assessment of future play is *absolutely continuous* with respect to the true probabilities of future play (i.e. any event that has true positive probability is assigned positive probability by the player), then (5.3) must hold. In our case, absolute continuity always holds by Assumption 3 and the fact that the prior probabilities P_j are positive, as well as the fact that the type distribution is pure (from which we can infer that the true types always have positive posterior probability).

In this proof, we seek to extend the convergence result of Kalai and Lehrer (1993) (henceforth KL) to multi-state SBGs with pure type distributions. Our strategy is to translate a SBG Γ into a *modified SBG* $\hat{\Gamma}$ which is equivalent to Γ in the sense that the players behave identically, and which is compatible to the model used in KL in the sense that the informational assumptions of KL ignore the differences. We achieve this by introducing a new player *nature*, denoted ξ, which emulates the transitions of Γ in $\hat{\Gamma}$.

Given a SBG $\Gamma = (S, s^0, \bar{S}, N, A_i, \Theta_i, u_i, \pi_i, T, \Upsilon)$, we define the modified SBG $\hat{\Gamma}$ as follows: Firstly, $\hat{\Gamma}$ has only one state, which can be arbitrary since it has no effect. The players in $\hat{\Gamma}$ are $\hat{N} = N \cup \{\xi\}$ where $i \in N$ have the same actions and types as in Γ (i.e. A_i and Θ_i), and where we define the actions and types of ξ to be $A_\xi = \Theta_\xi = S$ (i.e. nature's actions and types correspond to the states of Γ). The payoffs of ξ are always zero and the strategy of ξ at time t is defined as

$$\pi_\xi^t(H^\tau, a_\xi, \theta_\xi) = \begin{cases} 0 & \tau = t, a_\xi \not\equiv \theta_\xi \\ 1 & \tau = t, a_\xi \equiv \theta_\xi \\ T(a_\xi^{\tau-1}, (a_i^{\tau-1})_{i\in N}, a_\xi) & \tau > t \end{cases} \tag{5.4}$$

where H^τ is any history of length $\tau \geq t$. (H^τ allows the players $i \in N$ to use π_ξ^t for future predictions about ξ's actions. This will be necessary to establish equivalence of $\hat{\Gamma}$ / Γ.)

The purpose of ξ is to emulate the state transitions of Γ. Therefore, the modified strategies $\hat{\pi}_i$ and payoffs $\hat{u}_i$ of $i \in N$ are now defined with respect to the actions and types (since the current type of ξ determines its next action) of ξ.

Formally, $\hat{\pi}_i(H^t, a_i, \theta_i) = \pi_i(\bar{H}^t, a_i, \theta_i)$, where

$$\bar{H}^t = (\theta_\xi^0, (a_i^0)_{i\in N}, \theta_\xi^1, (a_i^1)_{i\in N}, ..., \theta_\xi^t) \tag{5.5}$$

and $\hat{u}_i(s, a^t, \theta_i^t) = u_i(\theta_\xi^t, (a_j^t)_{j\in N}, \theta_i^t)$, where s is the only state of $\hat{\Gamma}$ and $a^t \in \times_{i\in\hat{N}} A_i$.

Finally, $\hat{\Gamma}$ uses two type distributions, Υ and Υ_ξ, where Υ is the type distribution of Γ and Υ_ξ is defined as $\Upsilon_\xi(H^t, \theta_\xi) = T(a_\xi^{t-1}, (a_i^{t-1})_{i\in N}, \theta_\xi)$. If s^0 is the initial state of Γ, then $\Upsilon_\xi(H^0, \theta_\xi) = 1$ for $\theta_\xi \equiv s^0$.

The modified SBG $\hat{\Gamma}$ proceeds as the original SBG Γ, except for the following changes: *(a)* Υ is used to sample the types for $i \in N$ (as usual) while Υ_ξ is used to sample the types for ξ; *(b)* Each player is informed about its own type *and* the type of ξ. This completes the definition of $\hat{\Gamma}$.

The modified SBG $\hat{\Gamma}$ is equivalent to the original SBG Γ in the sense that the players $i \in N$ have identical behaviour in both SBGs. Since the players always know the type of ξ, they also know the next action of ξ, which corresponds to knowing the current state of the game. Furthermore, note that the strategy of ξ uses two time indices, t and τ, which allow it to distinguish between the current time ($\tau = t$) and a future time ($\tau > t$). This means that π_ξ^t can be used to compute expected payoffs in $\hat{\Gamma}$ in the same way as T is used to compute expected payoffs in Γ. In other words, the formulas (2) and (3) can be modified in a straightforward manner by replacing the original components of Γ with the modified components of $\hat{\Gamma}$, yielding the same results. Finally, since $\hat{\Gamma}$ uses the same type distribution as Γ to sample types for $i \in N$, there are no differences in their payoffs and strategies.

To complete the proof, we note that *(a)* and *(b)* are the only procedural differences between the modified SBG and the model used in KL. However, since we specify that the players always know the type of ξ, there is no need to learn the type distribution Υ_ξ, hence *(a)* and *(b)* have no effect in KL. The important point is that KL assume a model in which the players only interact with other players, but not with an environment. Since we eliminated the environment by replacing it with a player ξ, this is precisely what happens in the modified SBG. Therefore, the convergence result of KL carries over to multi-state SBGs with pure type distributions. □

Theorem 1 states that HBA will eventually make correct future predictions when using a product posterior against a pure type distribution (assuming the prior beliefs

are positive). However, there is a subtle but important asymmetry between making correct future predictions and knowing the true type distribution: while the latter implies the former, the reverse is not generally true. Therefore, while HBA is guaranteed to make correct future predictions after some time, it is not guaranteed to learn the type distribution of the game. The following example[1] illustrates this:

Example 2. Consider the repeated Prisoner's Dilemma game from Section 4.1. Assume player 1 is controlled by HBA using a product posterior while player 2 has two potential types, $\Theta_2^+ = \{\theta_{\lambda=0.1}, \theta_{\lambda=0.5}\}$, which are assigned by some pure type distribution. The two types choose action *C* if player 1 chose C in the previous round. Otherwise, with probability λ, they will forever play action D. In this case, HBA will never know the correct type with absolute certainty. Even if HBA chooses D and player 2 responds by playing D indefinitely, there is still no certainty because $\lambda > 0$ in both types.

Finally, note that Theorem 1 pertains to pure type distributions only. The following example shows that the product posterior may fail in SBGs with mixed type distributions:

Example 3. Consider a SBG with two players. Player 1 is controlled by HBA using a product posterior while player 2 has two types, $\Theta_2^+ = \{\theta_A, \theta_B\}$, which are assigned by a mixed type distribution Υ with $\Upsilon(\theta_A) = \Upsilon(\theta_B) = 0.5$. The type θ_A always chooses action A while θ_B always chooses action B. In this case, there will be a time t after which both types have been assigned at least once, and so both actions A and B have been played at least once by player 2. This means that from time t and all subsequent times $\tau \geq t$, we have $\Pr_2(\theta_A|H^\tau) = \Pr_2(\theta_B|H^\tau) = 0$ (that is, $\Pr_2$ is undefined), and HBA will fail to make correct future predictions.

5.3 Sum Posterior

We continue our analysis with the sum posterior:

Definition 13. The *sum posterior* is defined as (5.1) with

$$L(H^t|\theta_j^*) = \sum_{\tau=0}^{t-1} \pi_j(H^\tau, a_j^\tau, \theta_j^*). \tag{5.6}$$

The sum posterior was introduced in Chapter 4 (more precisely, a generalisation of the sum posterior; see Section 4.1.2) to allow HBA to recognise changed types. In other words, the purpose of the sum posterior is to learn mixed type distributions. It is easy to

[1] All examples in this chapter assume $\Theta_j^* = \Theta_j^+$ and uniform prior beliefs $P_j(\theta_j^*) = |\Theta_j^*|^{-1}$.

see that a sum posterior would indeed learn the mixed type distribution in Example 3. However, we now give an example to show that, without additional requirements, the sum posterior does not necessarily learn any (pure or mixed) type distribution:

Example 4. Consider a SBG with two players. Player 1 is controlled by HBA using a sum posterior while player 2 has two types, $\Theta_2^+ = \{\theta_A, \theta_{AB}\}$, which are assigned by a pure type distribution Υ with $\Upsilon(\theta_A) = 1$. The type θ_A always chooses action A while θ_{AB} chooses actions A and B with equal probability. While the product posterior converges to the correct probabilities Υ, the sum posterior converges to probabilities $\langle \frac{2}{3}, \frac{1}{3} \rangle$, which is incorrect.

Note that this example can be readily modified to use a mixed type distribution, with similar results. Therefore, we conclude that, without further assumptions, the sum posterior does not necessarily learn any type distribution.

Under what condition is the sum posterior guaranteed to learn the true type distribution of the game? Consider the following two quantities, which can be computed from a given history H^t:

Definition 14. The *average overlap* of player j in H^t is defined as

$$\mathrm{AO}_j(H^t) = \frac{1}{t} \sum_{\tau=0}^{t-1} \left[|\Lambda_j^\tau| \geq 2 \right]_1 \sum_{\theta_j^* \in \Theta_j^*} \pi_j(H^\tau, a_j^\tau, \theta_j^*) \, |\Theta_j^*|^{-1} \tag{5.7}$$

$$\Lambda_j^\tau = \left\{ \theta_j^* \in \Theta_j^* \,|\, \pi_j(H^\tau, a_j^\tau, \theta_j^*) > 0 \right\} \tag{5.8}$$

where $[b]_1 = 1$ if b is true, else 0.

Definition 15. The *average stochasticity* of player j in H^t is defined as

$$\mathrm{AS}_j(H^t) = \frac{1}{t} \sum_{\tau=0}^{t-1} |\Theta_j^*|^{-1} \sum_{\theta_j^* \in \Theta_j^*} \frac{1 - \pi_j(H^\tau, \hat{a}_j^\tau, \theta_j^*)}{1 - |A_j|^{-1}} \tag{5.9}$$

where $\hat{a}_j^\tau \in \arg\max_{a_j} \pi_j(H^\tau, a_j, \theta_j^*)$.

Both quantities are bounded by 0 and 1. The average overlap describes the similarity of the types, where $\mathrm{AO}_j(H^t) = 0$ means that player j's types (on average) never chose the same action in history H^t, whereas $\mathrm{AO}_j(H^t) = 1$ means that they behaved identically. The average stochasticity describes the uncertainty of the types, where $\mathrm{AS}_j(H^t) = 0$ means that player j's types (on average) were fully deterministic in the action choices in history H^t, whereas $\mathrm{AS}_j(H^t) = 1$ means that they chose actions uniformly randomly.

It can be shown that, if the average overlap and stochasticity of player j converge to zero as $t \to \infty$, then the sum posterior is guaranteed to converge to any pure or mixed type distribution:

Theorem 2. Let Γ be a SBG with a pure or mixed type distribution Υ. If HBA uses a sum posterior, then, for $t \to \infty$: If $\mathrm{AO}_j(H^t) = 0$ and $\mathrm{AS}_j(H^t) = 0$ for all players $j \neq i$, then $\Pr(\theta_{-i}|H^t) = \Upsilon(\theta_{-i})$ for all $\theta_{-i} \in \Theta^+_{-i}$.

Proof. Throughout this proof, let $t \to \infty$. The sum posterior is defined as (5.1) where L is defined as (5.6). Given the definition of L, both the numerator and the denominator in (5.1) may be infinite. We invoke L'Hôpital's rule which states that, in such cases, the quotient $\frac{u(t)}{v(t)}$ is equal to the quotient $\frac{u'(t)}{v'(t)}$ of the respective derivatives with respect to t. The derivative of L with respect to t is the average growth per time step, which in general may depend on the history H^t of states and actions. The average growth of L is

$$L'(H^t|\theta_j) = \sum_{a_j \in A_j} F(a_j|H^t)\,\pi_j(H^t, a_j, \theta_j) \tag{5.10}$$

where

$$F(a_j|H^t) = \sum_{\theta_j \in \Theta^+_j} \Upsilon(\theta_j)\,\pi_j(H^t, a_j, \theta_j) \tag{5.11}$$

is the probability of action a_j after history H^t, with $\Upsilon(\theta_j)$ being the marginal probability that player j is assigned type θ_j. As we will see shortly, we can make an asymptotic growth prediction irrespective of H^t. Given that $\mathrm{AO}_j(H^t) = 0$, we can infer that whenever $\pi_j(H^t, a_j, \theta^*_j) > 0$ for action a_j and type θ^*_j, then $\pi_j(H^t, a_j, \hat{\theta}^*_j) = 0$ for all other types $\hat{\theta}^*_j \neq \theta^*_j$ with $\hat{\theta}^*_j \in \Theta^*_j$. Therefore, we can write (5.10) as

$$L'(H^t|\theta_j) = \Upsilon(\theta_j) \sum_{a_j \in A_j} \pi_j(H^t, a_j, \theta_j)^2 \tag{5.12}$$

Next, given that $\mathrm{AS}_j(H^t) = 0$, we know that there exists an action a_j in (5.12) with $\pi_j(H^t, a_j, \theta_j) = 1$, and, therefore, we can conclude that $L'(H^t|\theta_j) = \Upsilon(\theta_j)$. This shows that the history H^t is irrelevant to the asymptotic growth rate of L. Finally, since $\sum_{\theta_j \in \Theta^+_j} \Upsilon(\theta_j) = 1$, we know that the denominator in (5.1) will be 1, and we conclude that $\Pr_j(\theta_j|H^t) = \Upsilon(\theta_j)$. □

Theorem 2 explains why the sum posterior converges to the correct type distribution in Example 3. Since the types θ_A and θ_B always choose different actions and are completely deterministic (i.e. the average overlap and stochasticity are always zero), the sum posterior is guaranteed to converge to the type distribution. On the other hand, in

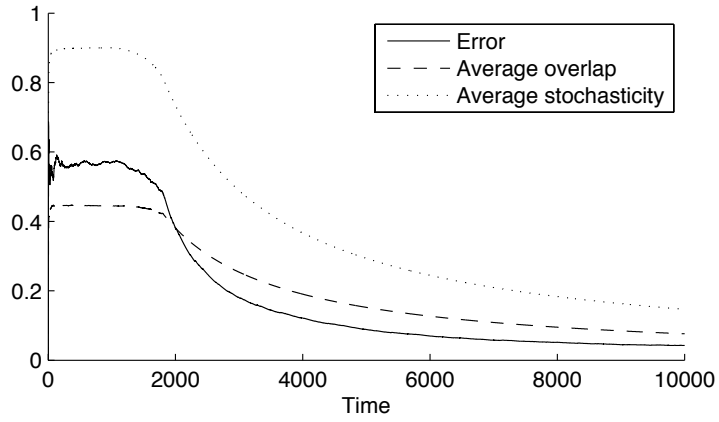


Figure 5.1: Example run in random SBG with 2 players, 10 actions, and 100 states. Player j has 3 reinforcement learning types with ε-greedy action selection (decreasing linearly from $\varepsilon = 0.7$ at $t = 1000$, to $\varepsilon = 0$ at $t = 2000$). The error at time t is computed as $\sum_{\theta_j \in \Theta_j^+} |\Pr_j(\theta_j|H^t) - \Upsilon(\theta_j)|$, where $\Pr_j$ is the sum posterior.

Example 4 the types θ_A and θ_{AB} produce an overlap whenever action A is chosen, and θ_{AB} is completely random. Therefore, the average overlap and stochasticity are always positive, and an incorrect type distribution was learned.

The assumptions made in Theorem 2, namely that the average overlap and stochasticity converge to zero, require practical justification. First of all, it is important to note that it is only required that these converge to zero *on average* as $t \to \infty$. This means that in the beginning there may be arbitrary overlap and stochasticity, as long as these go to zero as the game proceeds. In fact, with respect to stochasticity, this is precisely how the exploration-exploitation dilemma (Sutton and Barto, 1998) is solved in practice: In the early stages, the agent randomises deliberately over its actions in order to obtain more information about the environment (*exploration*) while, as the game proceeds, the agent becomes gradually more deterministic in its action choices so as to maximise its payoffs (*exploitation*). Typical mechanisms which implement this are ε-greedy and Softmax/Boltzmann exploration (Sutton and Barto, 1998). Figure 5.1 demonstrates this in a SBG in which player j has 3 reinforcement learning types. The payoffs for the types were such that the average overlap would eventually go to zero.

Regarding the average overlap converging to zero, we believe that this is a property which should be guaranteed *by design*, for the following reason: If the hypothesised type space Θ_j^* is such that there is a constantly-high average overlap, then this means that the types $\theta_j^* \in \Theta_j^*$ are in effect very similar. However, types which are very similar are likely to produce very similar trajectories in the planning step of HBA (cf. $\hat{H}$ in (3.10)

and (3.11)) and, therefore, constitute redundancy in both time and space. Therefore, we believe it is advisable to use type spaces which have low average overlap.

5.4 Correlated Posterior

An implicit assumption in the definition of (5.1) is that the type distribution Υ can be represented as a product of n independent factors (one for each player), such that $\Upsilon(\theta) = \prod_j \Upsilon_j(\theta_j)$. Therefore, since the sum posterior is in the form of (5.1), it is in fact only guaranteed to learn *independent* type distributions. This is opposed to *correlated* type distributions, which cannot be represented as a product of n independent factors. Correlated type distributions can be used to specify constraints on type combinations, such as "player j can only have type θ_j if player k has type θ_k". The following example demonstrates how the sum posterior fails to converge to a correlated type distribution:

Example 5. Consider a SBG with 3 players. Player 1 is controlled by HBA using a sum posterior. Players 2 and 3 each have two types, $\Theta_2^+ = \Theta_3^+ = \{\theta_A, \theta_B\}$, which are defined as in Example 3. The type distribution Υ chooses types with probabilities $\Upsilon(\theta_A, \theta_B) = \Upsilon(\theta_B, \theta_A) = 0.5$ and $\Upsilon(\theta_A, \theta_A) = \Upsilon(\theta_B, \theta_B) = 0$. In other words, player 2 can never have the same type as player 3. From the perspective of HBA, each type (and hence action) is chosen with equal probability for both players. Thus, despite the fact that there is zero overlap and stochasticity, the sum posterior will eventually assign probability 0.25 to all constellations of types, which is incorrect. This means that HBA fails to recognise that the other players never choose the same action.

In this section, we propose a posterior formulation which can learn any correlated type distribution:

Definition 16. The *correlated posterior* is defined as

$$\Pr(\theta^*_{-i}|H^t) = \eta P(\theta^*_{-i}) \sum_{\tau=0}^{t-1} \prod_{\theta^*_j \in \theta^*_{-i}} \pi_j(H^\tau, a^\tau_j, \theta^*_j) \tag{5.13}$$

where P specifies prior beliefs over Θ^*_{-i} (analogous to P_j) and η is a normaliser.

The correlated posterior is closely related to the sum posterior. In fact, it converges to the correct type distribution under the same conditions as the sum posterior:

Theorem 3. Let Γ be a SBG with a *correlated* type distribution Υ. If HBA uses the correlated posterior, then, for $t \to \infty$: If $\mathrm{AO}_j(H^t) = 0$ and $\mathrm{AS}_j(H^t) = 0$ for all players $j \neq i$, then $\Pr(\theta_{-i}|H^t) = \Upsilon(\theta_{-i})$ for all $\theta_{-i} \in \Theta^+_{-i}$.

Proof. The proof is analogous to the proof of Theorem 2. □

It is easy to see that the correlated posterior would learn the correct type distribution in Example 5. Note that, since it is guaranteed to learn any correlated type distribution, it is also guaranteed to learn any independent type distribution. Therefore, the correlated posterior would also learn the correct type distribution in Example 3. This means that the correlated posterior is *complete* in the sense that it covers the entire spectrum of pure/mixed and independent/correlated type distributions. However, this completeness comes at a higher computational complexity. While the sum posterior is in $O(n \max_j |\Theta_j^*|)$ time and space, the correlated posterior is in $O(\max_j |\Theta_j^*|^n)$ time and space. In practice, however, the time complexity can be reduced substantially by computing the probabilities $\pi_j(H^\tau, a_j^\tau, \theta_j^*)$ only once for each j and $\theta_j^* \in \Theta_j^*$ (as in the sum posterior), and then reusing them in subsequent computations.

Chapter 6

Practical Impact of Prior Beliefs

The previous chapter was concerned with the evolution of *posterior beliefs* as we observe more evidence. However, before we observe any evidence based on which to form our posterior beliefs, we will have to make an initial judgement as to the relative likelihood of types. This initial judgement is called the *prior belief*.

Given the lack of evidence, it may be tempting to use uniform prior beliefs in which all types have equal probability. Indeed, the fact that beliefs can change rapidly after only a few observations suggests that prior beliefs may have negligible effect. On the other hand, there is a substantial body of work in the game theory literature arguing the importance of prior beliefs (Dekel et al., 2004; Kalai and Lehrer, 1993; Jordan, 1991). However, these works consider the impact of prior beliefs on equilibrium attainment, whereas our interest is in the *practical* impact of prior beliefs, i.e. payoff maximisation. Moreover, the game theory literature on this subject assumes that all players use the same Bayesian reasoning over types, while we make no such assumption.

Thus, we are left with the following questions: Do prior beliefs have an impact on our ability to maximise payoffs in the long-term? If so, how? And, crucially, can we automatically compute prior beliefs so as to improve our long-term performance?

To find answers to these questions, we conducted a comprehensive empirical study which compared 10 methods to automatically compute prior beliefs from a given set of types. The results show that prior beliefs can indeed have a significant impact on the long-term performance, and that the depth of the planning horizon (i.e. how far we look into the future) plays a central role. Finally, and perhaps most intriguingly, we show that automatic methods can compute prior beliefs with consistent performance effects across a variety of scenarios. An implication of this is that prior beliefs could be eliminated as a manual parameter and instead be computed automatically.

6.1 Experimental Setup

This section describes the experimental setup used in our study.

6.1.1 Games

We used a comprehensive set of benchmark games introduced by Rapoport and Guyer (1966), which consists of 78 repeated 2×2 matrix games (i.e. 2 players with 2 actions). The games are *strictly ordinal*, meaning that each player ranks each of the 4 possible outcomes from 1 (least preferred) to 4 (most preferred), and no two outcomes have the same rank. Furthermore, the games are *distinct* in the sense that no game can be obtained by transformation of any other game, which includes interchanging the rows, columns, and players (and any combination thereof) in the payoff matrix of the game.

The games can be grouped into 21 *no-conflict* games and 57 *conflict* games. In a no-conflict game, the two players have the same most preferred outcome, and so it is relatively easy to arrive at a solution that is best for both players. In a conflict game, the players disagree on the best outcome, hence they will have to find some form of a compromise. A listing of all games is given in Appendix A.

6.1.2 Performance Criteria

Each play of a game was partitioned into *time slices* which consist of an equal number of consecutive time steps. For each time slice, we measured the following performance criteria:

Convergence: An agent converged in a time slice if its action probabilities in the time slice did not deviate by more than 0.05 from its initial action probabilities in the same time slice. Returns 1 (true) or 0 (false) for each agent.

Average payoff: Average of payoffs an agent received in the time slice. Returns value in $[1,4]$ for each agent.

Welfare and fairness: Average sum and product, respectively, of the joint payoffs received in the time slice. Returns values in $[2,8]$ and $[1,16]$, respectively.

Game solutions: Tests if the averaged action probabilities of the agents (averaged over the time slice) formed an approximate stage-game Nash equilibrium, Pareto optimum, Welfare optimum, or Fairness optimum in the time slice. Returns 1 (true) or 0 (false) for each game solution.

Precise formal definitions of these performance criteria can be found in (Albrecht and Ramamoorthy, 2012).

6.1.3 Algorithm

In this study, we used HBA to control player 1 and a single type in each interaction to control player 2, which was always included in the set of hypothesised types Θ_2^* provided to HBA (discussed in detail in Section 6.1.6). Hence, we used the product posterior formulation as defined in Section 5.2 to update HBA's beliefs.

We implemented HBA similarly to Section 4.1 (Algorithm 3), by expanding a finite tree of all future trajectories and weighting each trajectory using the posterior beliefs and type predictions. Formally, HBA chooses an action a_i which maximises the expected payoff $E_h^{a_i}(H^t)$, defined as

$$E_h^{a_i}(\hat{H}) = \sum_{\theta_j^* \in \Theta_j^*} \Pr_j(\theta_j^* | H^t) \sum_{a_j \in A_j} \pi_j(\hat{H}, a_j, \theta_j^*) Q_{h-1}^{(a_i,a_j)}(\hat{H}) \tag{6.1}$$

$$Q_h^{(a_i,a_j)}(\hat{H}) = u_i(a_i,a_j) + \begin{cases} 0 \text{ if } h = 0, \text{ else} \\ \max_{a_i'} E_h^{a_i'}\left(\langle \hat{H}, (a_i,a_j) \rangle\right) \end{cases} \tag{6.2}$$

where h specifies the depth of the planning horizon (i.e. HBA predicts the next h actions of player j). Note that H^t is the current history of joint actions while $\hat{H}$ is used to construct all future trajectories in the game. The only difference between this implementation and the one used in Section 4.1 is that the latter incorporated the total number of rounds to be played while this implementation does not (that is, it plans its actions irrespective of the remaining play time). This was done to avoid "end-game" effects which could complicate the analysis.

Note that (6.1) and (6.2) correspond closely to (3.10) and (3.11), respectively. The difference is that (6.1)/(6.2) use h to specify the planning depth while (3.10)/(3.11) use the discount factor γ. Hence, a "deeper" planning horizon h translates into a greater discount factor γ. All results reported in this chapter hold for both variants.

6.1.4 Types

We used three different methods to automatically generate parameterised sets of types Θ_j^* for any given game. The generated types cover a broad spectrum of adaptive behaviours, including deterministic (CDT), randomised (CNN), and hybrid (LFT) policies. Algorithmic details and parameter settings are given in Appendix B.

Leader-Follower-Trigger Agents (LFT)

Crandall (2014) described a method to automatically generate sets of "leader" and "follower" agents which seek to play specific sequences of joint actions, called "target solutions". These solutions must satisfy certain requirements, such as that the average payoffs of the solution exceed the safety (maximin) values of the players. A leader agent plays its part of the target solution as long as the other player does. If the other player deviates, the leader agent punishes the player by playing a minimax strategy. The follower agent is similar except that it does not punish. Rather, if the other player deviates, the follower agent randomly resets its position within the target solution and continues play as usual. We augmented this set by a "trigger" agent which is similar to the leader and follower agents, except that it plays its maximin strategy indefinitely once the other player deviates.

Co-Evolved Decision Trees (CDT)

We used genetic programming (Koza, 1992) to automatically breed sets of decision trees. A decision tree takes as input the past n actions of the other player (in our case, $n = 3$) and deterministically returns an action to be played in response. (Note that, since the tree is deterministic, it can infer what it played in the past given the other player's actions.) The breeding process is co-evolutional, meaning that two pools of trees are bred concurrently (one for each player). In each evolution, a random selection of the trees for player 1 is evaluated against a random selection of the trees for player 2. The fitness criterion includes the payoffs generated by a tree as well as its dissimilarity to other trees in the same pool. This was done to encourage a more diverse breeding of trees, as otherwise the trees tend to become very similar or identical (this happens, for example, when the breeding converges to a Nash equilibrium).

Co-Evolved Neural Networks (CNN)

We used a string-based genetic algorithm (Holland, 1975) to breed sets of artificial neural networks. The process is basically the same as the one used for decision trees. However, the difference is that artificial neural networks can learn to play stochastic strategies while decision trees always play deterministic strategies. Our networks consist of one input layer with 4 nodes (one for each of the two previous actions of both players), a hidden layer with 5 nodes, and an output layer with 1 node. The node in the output layer specifies the probability of choosing action 1 (and, since we play 2×2 games,

of action 2). All nodes use a sigmoidal threshold function and are fully connected to the nodes in the next layer. The evolution process places greater weight on mutation than crossover, since the latter has a tendency to destroy learned behaviours in neural networks rather than improve them.

6.1.5 Prior Beliefs

We specified a total of 10 different methods to automatically compute prior beliefs P_j for a given set of types Θ_j^*:

Uniform prior

The uniform prior sets $P_j(\theta_j^*) = |\Theta_j^*|^{-1}$ for all $\theta_j^* \in \Theta_j^*$. This is the baseline prior against which the other priors are compared.

Random prior

The random prior specifies $P_j(\theta_j^*) = .0001$ for a random half of the types in Θ_j^*. The remaining probability mass is uniformly spread over the other half. The random prior is used to check if the performance differences of the various priors may be purely due to the fact that they concentrate the probability mass on fewer types.

Value priors

Let $U_k^t(\theta_j^*)$ be the expected cumulative payoff to player k, from the start up until time t, if player j (i.e. the other player) is of type θ_j^* and player i (i.e. HBA) plays optimally against it. Each value prior is in the general form of $P_j(\theta_j^*) = \eta\,\psi(\theta_j^*)^b$, where η is a normalisation constant and b is a "booster" exponent used to magnify the differences between types θ_j^*. Based on this general form, we define four different value priors:

- Utility prior: $\psi_U(\theta_j^*) = U_i^t(\theta_j^*)$
- Stackelberg prior: $\psi_S(\theta_j^*) = U_j^t(\theta_j^*)$
- Welfare prior: $\psi_W(\theta_j^*) = U_i^t(\theta_j^*) + U_j^t(\theta_j^*)$
- Fairness prior: $\psi_F(\theta_j^*) = U_i^t(\theta_j^*) * U_j^t(\theta_j^*)$

Our choice of value priors is motivated by the variety of metrics they cover. As a result, these priors can produce substantially different probabilities for the same set of types. In this study, we set $t = 5$ and $b = 10$.

LP-priors

LP-priors are based on the idea that optimal priors can be formulated as the solution to a mathematical optimisation problem (in this case, a linear program). Each LP-prior generates a quadratic matrix A, where each element $A_{j,j'}$ contains the "loss" that HBA would incur if it planned its actions against the type $\theta^*_{j'}$ while the true type of player j is θ^*_j. Formally, let $U^t_k(\theta^*_j|\theta^*_{j'})$ be like $U^t_k(\theta^*_j)$ except that HBA believes that player j is of type $\theta^*_{j'}$ instead of θ^*_j. We define four different LP-priors:

- LP-Utility: $A_{j,j'} = \psi_U(\theta^*_j) - U^t_i(\theta^*_j|\theta^*_{j'})$
- LP-Stackelberg: $A_{j,j'} = \psi_S(\theta^*_j) - U^t_j(\theta^*_j|\theta^*_{j'})$
- LP-Welfare: $A_{j,j'} = \psi_W(\theta^*_j) - [U^t_i(\theta^*_j|\theta^*_{j'}) + U^t_j(\theta^*_j|\theta^*_{j'})]$
- LP-Fairness: $A_{j,j'} = \psi_F(\theta^*_j) - [U^t_i(\theta^*_j|\theta^*_{j'}) * U^t_j(\theta^*_j|\theta^*_{j'})]$

The matrix A can be fed into a linear program of the form $\min_c c^T x$ s.t. $[z,A]x \leq 0$, with $n = |\Theta^*_j|$, $c = (1,\{0\}^n)^T$, $z = (\{-1\}^n)^T$, to find a vector $x = (l, p_1, ..., p_n)$ in which l is the minimised expected loss to HBA when using the probabilities $p_1, ..., p_n$ (one for each type) as the prior belief P_j. In order to avoid premature elimination of types, we furthermore require that $p_v > 0$ for all $1 \leq v \leq n$. As before, we set $t = 5$ and $b = 10$.

While this is a mathematically rigorous formulation, it is important to note that it is a simplification of how HBA really works. HBA uses the prior in every recursion of its planning procedure, while the LP formulation implicitly assumes that HBA uses the prior to randomly sample one of the types against which it then plans optimally. Nonetheless, this is often a reasonable approximation.

6.1.6 Experimental Procedure

We performed identical experiments for every type generation method described in Section 6.1.4. Each of the 78 games was played 10 times with different random seeds, and each play was repeated against three opponents (30 plays in total):

1. (RT) A randomly generated type was used to control player 2 and the play lasted 100 rounds.

2. (FP) A fictitious player (Brown, 1951) was used to control player 2 and the play lasted 10,000 rounds.

3. (CFP) A conditioned fictitious player (which learns action distributions conditioned on the previous joint action) was used to control player 2 and the play lasted 10,000 rounds.

In each play, we randomly generated 9 unique types and provided them to HBA along with the true type of player 2, such that $|\Theta_2^*| = 10$. (In terms of our earlier definitions, each play had a static pure type distribution; cf. Section 3.1) Thus, the true type of player 2 was always included in the set of hypothesised types Θ_2^*. To avoid "end-game" effects, the players were unaware the number of rounds.

We included FP and CFP because they try to learn the behaviour of HBA. (While the generated types are adaptive, they do not create models of HBA's behaviour.) To facilitate the learning, we allowed for 10,000 rounds. Finally, since FP and CFP will always choose dominating actions if they exist (in which case there is no interaction), we filtered out all games in the FP and CFP plays that had a dominating action for player 2 (leaving 15 no-conflict and 33 conflict games for the C/FP plays).

6.2 Results

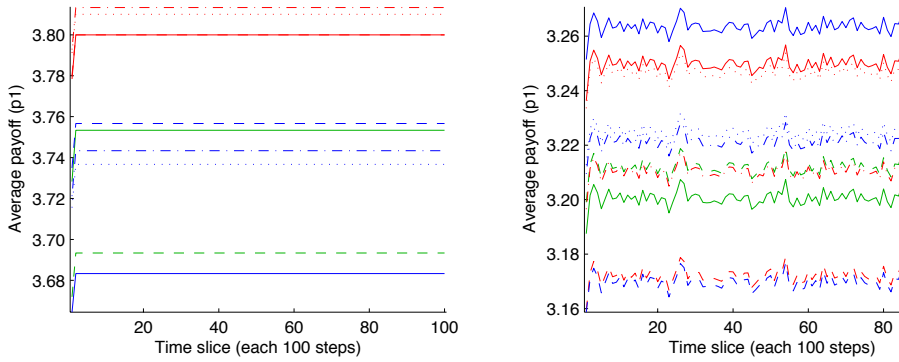
A complete listing of all experimental results can be found in an appendix document (Albrecht et al., 2015b). Based on this data, we make three main observations:

Observation 1. *Prior beliefs can have a significant impact on the long-term performance of HBA.*

This was observed in all classes of types, against all classes of opponents, and in all classes of games used in this study. Figure 6.1 provides three representative examples from a range of scenarios. Many of the relative differences due to prior beliefs were statistically significant, based on paired two-sided t-tests with a 5% significance level.

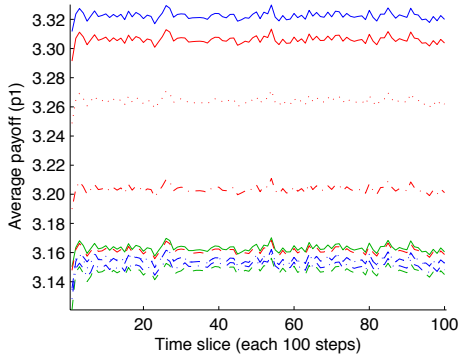
Our data explain this as follows: Different prior beliefs may cause HBA to take different actions at the beginning of the game. These actions will shape the beliefs of the other player (i.e. how it models and adapts to HBA's actions) which in turn will affect HBA's next actions. Thus, if different prior beliefs lead to different initial actions, they may lead to different play trajectories with different payoffs.

Given that there is a time after which HBA will know the true type of player 2 (since it is provided to HBA), it may seem surprising that this process would lead to differences in the long-term. In fact, in our experiments, HBA often learned the true type after only 3 to 5 rounds, and in most cases in under 20 rounds. After that point, if



(a) LFT (h=1) – CFP – No-Conflict

(b) CDT (h=3) – FP – Conflict



(c) CNN (h=5) – FP – Conflict

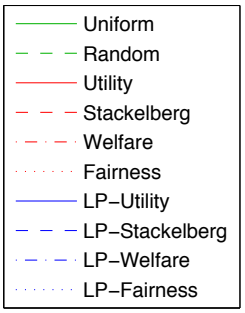


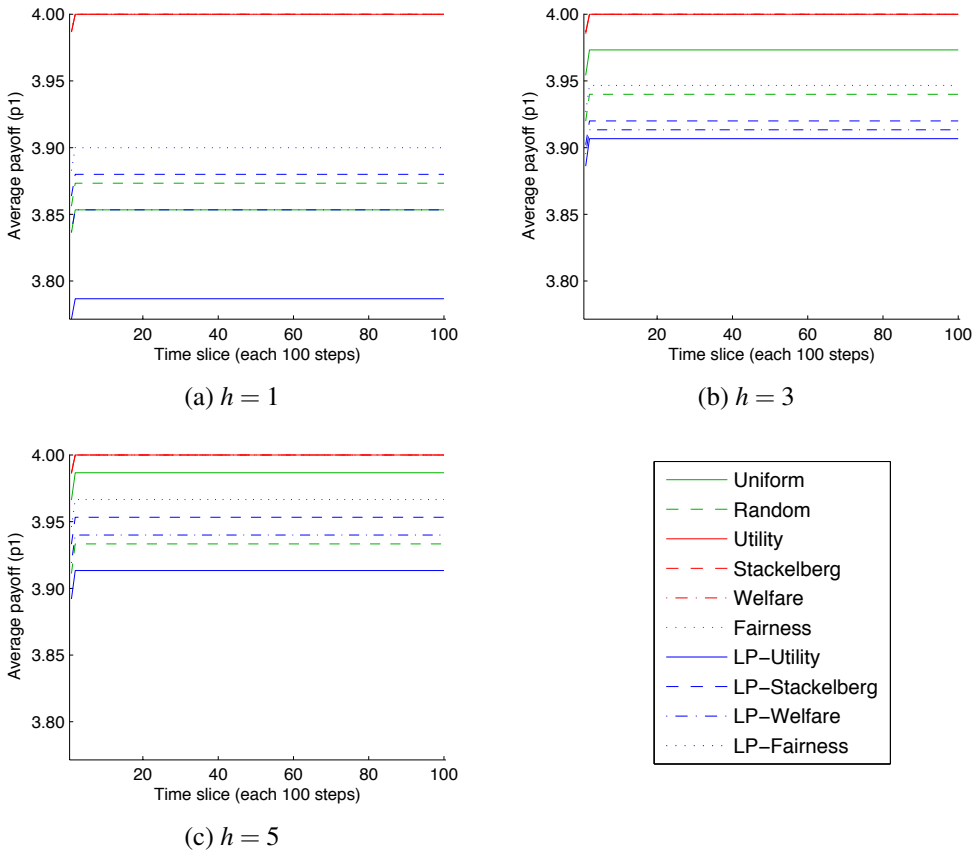
Figure 6.1: **Prior beliefs can have significant impact on long-term performance.** Plots show average payoffs of player 1 (HBA). X(h)–Y–Z format: HBA used X types and horizon h, player 2 was controlled by Y, and results are averaged over Z games.

the planning horizon of HBA is sufficiently deep, it will realise if its initial actions were sub-optimal and if it can manipulate the play trajectory to achieve higher payoffs in the long-term, thus diminishing the impact of prior beliefs.[1]

However, deep planning horizons can be problematic in practice since the time complexity of HBA is exponential in the depth of the planning horizon. Therefore, the planning horizon constitutes a trade-off between decision quality and computational tractability. Interestingly, our data show that if we increase the depth, but stay below a sufficient depth ("sufficient" as described above), it may also *amplify* the impact of prior beliefs:

Observation 2. *Deeper planning horizons can both diminish and amplify the impact of prior beliefs.*

[1] That is, provided that the other player's behaviour can still be changed. This is not the case, for instance, with trigger agents.



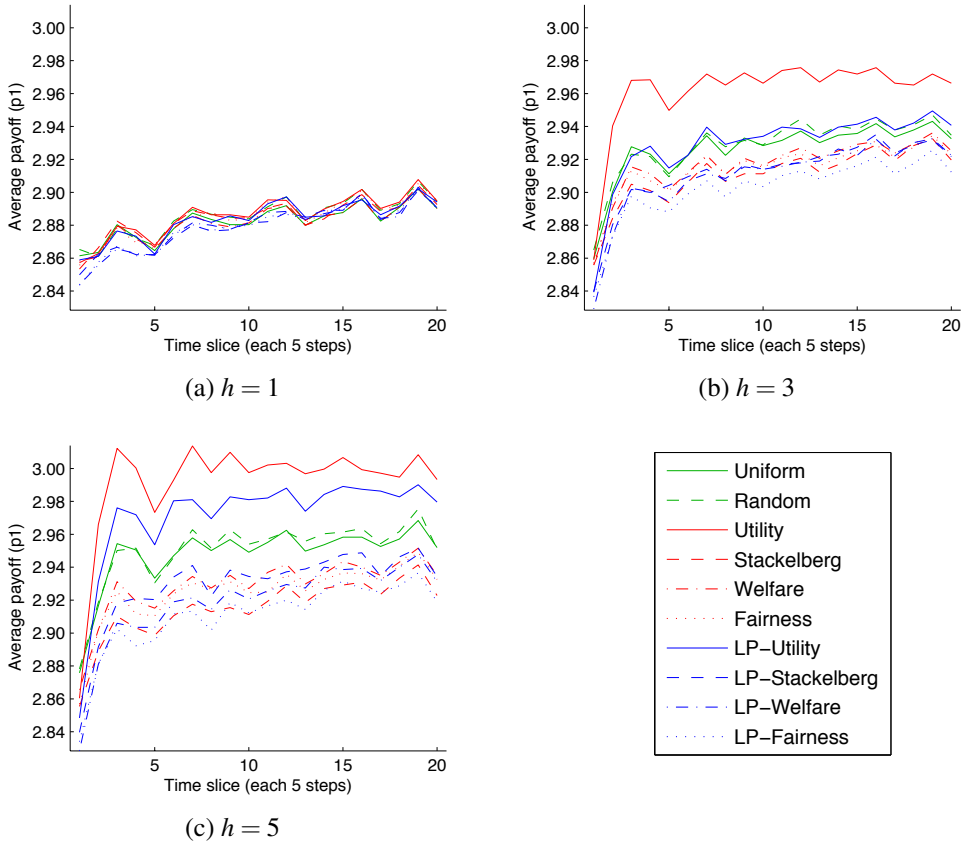
(a) $h = 1$ (b) $h = 3$

(c) $h = 5$

Figure 6.2: **Deeper planning horizons can diminish impact of prior beliefs.** Results shown for HBA with LFT types, player 2 controlled by FP, averaged over no-conflict games. h is depth of planning horizon (i.e. predicting h next actions of player 2).

Again, this was observed in all tested scenarios. Figures 6.2 and 6.3 show examples in which deeper planning horizons diminish and amplify the impact of prior beliefs, respectively.

How can deeper planning horizons amplify the impact of prior beliefs? Our data show that whether or not different prior beliefs cause HBA to take different initial actions depends not only on the prior beliefs and types, but also on the depth of the planning horizon. In some cases, differences between types (i.e. in their action choices) may be less visible in the near future and more visible in the distant future. In such cases, an HBA agent with a myopic planning horizon may choose the same (or similar) initial actions, despite different prior beliefs, because the differences in the types may not be visible within its planning horizon. On the other hand, an HBA agent with a deeper planning horizon may see the differences between the types and decide to choose different initial actions based on the prior beliefs.



(a) $h = 1$ (b) $h = 3$

(c) $h = 5$

Figure 6.3: **Deeper planning horizons can amplify impact of prior beliefs.** Results shown for HBA with CNN types, player 2 controlled by RT, averaged over conflict games. h is depth of planning horizon (i.e. predicting h next actions of player 2).

We now turn to a comparison between the different prior beliefs. Here, our data reveal an intriguing property:

Observation 3. *Automatic methods can compute prior beliefs with consistent performance effects.*

Figure 6.4 shows that the prior beliefs had consistent performance effects across a wide variety of scenarios. For example, the Utility prior produced consistently higher payoffs for player 1 (i.e. HBA) while the Stackelberg prior produced consistently higher payoffs for player 2 as well as higher welfare and fairness. The Welfare and Fairness priors were similar to the Stackelberg prior, but not quite as consistent. Similar results were observed for the LP variants of the priors, despite the fact that the LP formulation is a simplification of how HBA works (cf. Section 6.1.5).

We note that none of the prior beliefs, including the Uniform prior, produced high rates for the game solutions (i.e. Nash equilibrium, Pareto optimality, etc.). This is

because we measured *stage-game* solutions, which have no notion of time. These can be hard to attain in repeated games, especially if the other player does not actively seek a specific solution, as was often the case in our study.

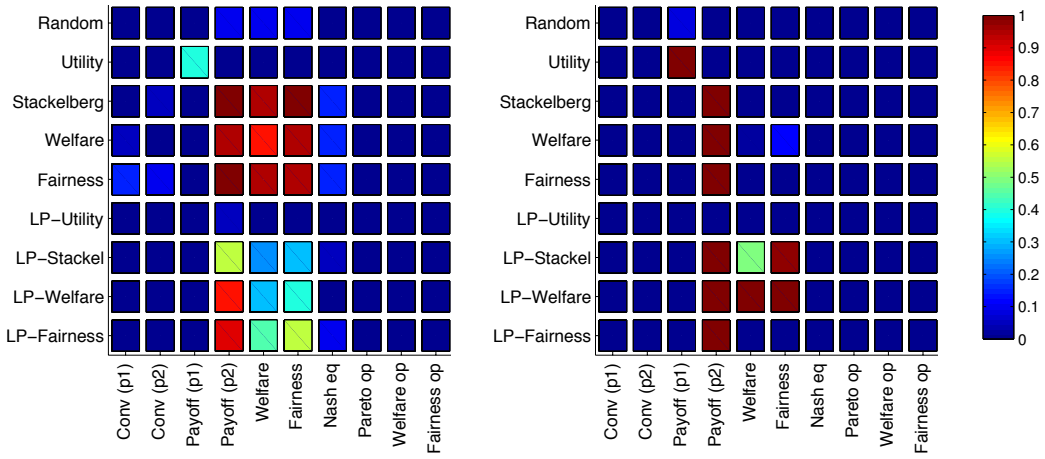
Observation 3 is intriguing because it indicates that prior beliefs could be eliminated as a manual parameter and instead be computed automatically, using methods such as the ones specified in Section 6.1.5. The fact that our methods produced consistent results means that prior beliefs can be constructed to optimise specific performance criteria. Note that this result is particularly interesting because the prior beliefs have no influence, whatsoever, on the true type of player 2.

This observation is further supported by the fact that the Random prior did not produce consistently different values (for any criterion) from the Uniform prior. This means that the differences in the prior beliefs are not merely due to the fact that they concentrate the probability mass on fewer types, but rather that the prior beliefs reflect the intrinsic metrics based on which they are computed (e.g. player 1's payoffs for Utility prior, player 2's payoffs for Stackelberg prior, etc.).

How is this phenomenon explained? We believe this may be an interesting analogy to the "optimism in uncertainty" principle (e.g. Brafman and Tennenholtz, 2003). The optimism lies in the fact that HBA commits to a specific class of types – those with high prior belief – while, in truth and without further evidence, there is no reason to believe that any one type is a priori more likely than others.

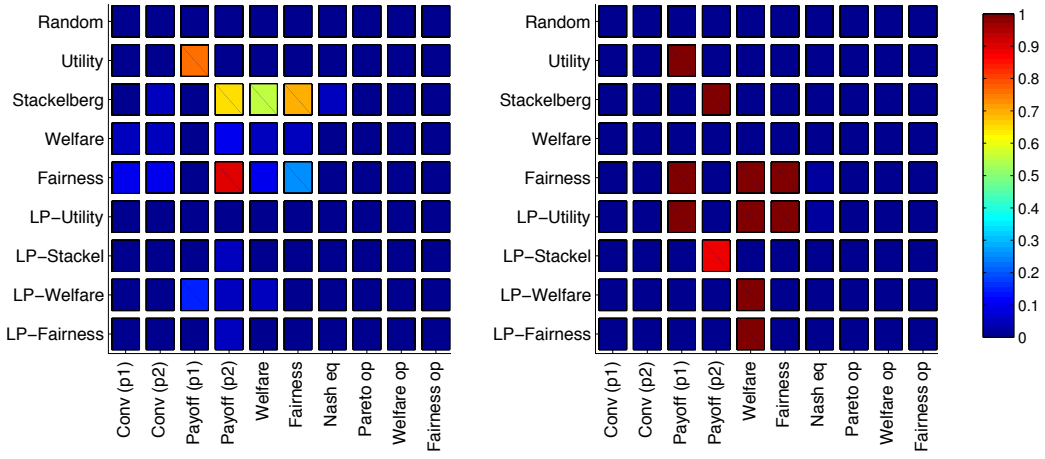
Each class of types is characterised by the intrinsic metric of the prior belief. For instance, the Utility prior assigns high probability to those types which would yield high payoffs to HBA if it played optimally against the types. By committing to such a characterisation, HBA can effectively utilise Observation 1 by choosing initial actions so as to shape the interaction to maximise the intrinsic metric. If the true type of player 2 is indeed in this class of types, then the interaction will proceed as planned by HBA and the intrinsic metric will be optimised. However, if the true type is not in this class, then HBA will quickly learn the correct type and adjust its play accordingly, albeit without necessarily maximising the intrinsic metric.

This is in contrast to the Uniform and Random priors, which have no intrinsic metric. Under these priors, HBA will plan its actions with respect to types which are not characterised by a common theme (i.e., all types under the Uniform prior, and a random half under the Random prior). Therefore, HBA cannot effectively utilise Observation 1.



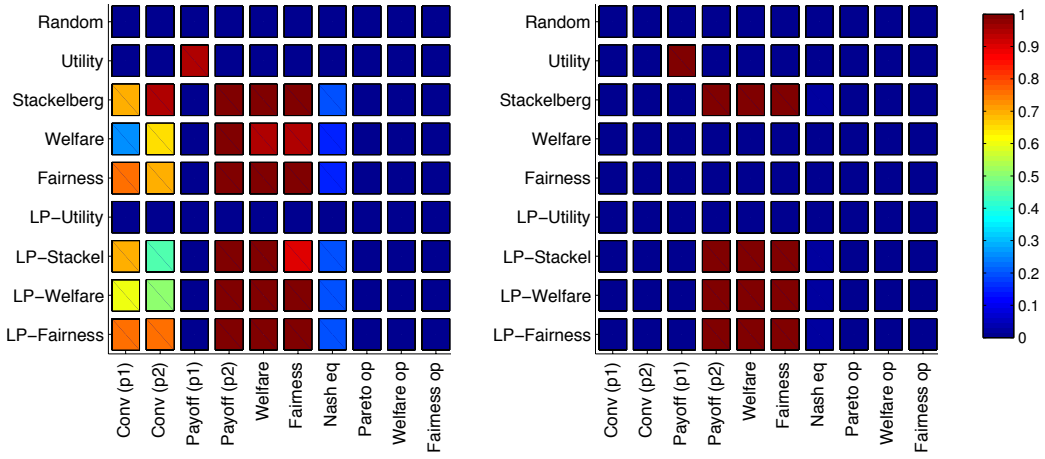
(a) LFT (h=5) – RT – Conflict

(b) LFT (h=5) – CFP – Conflict



(c) CDT (h=3) – RT – Conflict

(d) CDT (h=3) – FP – Conflict



(e) CNN (h=3) – RT – Conflict

(f) CNN (h=1) – FP – Conflict

Figure 6.4: **Automatic prior beliefs have consistent performance effects.** Rows show prior beliefs and columns show performance criteria. Each element (r, c) in the matrix corresponds to the percentage of time slices in which the prior belief r produced significantly higher values for the criterion c than the Uniform prior, averaged over all plays in all tested games. All significance statements are based on paired right-sided t-tests with a 5% significance level. See Figure 6.1 for X(h)–Y–Z format.

Chapter 7

Optimal Type Spaces

Perhaps the most obvious and crucial concern in the type-based methodology is the fact that the hypothesised types may be *incorrect*. This can range from slight deviations in predicted action probabilities, to predicting entirely different actions from what was observed. In either case, such inaccuracies may have a significant impact on our choice of actions: if the hypothesised types are incorrect, then our predictions of future interactions may be incorrect, which in turn may lead to suboptimal action choices. Therefore, an important question is what relation the hypothesised types must have to the true types in order for HBA to be able to solve its task? In particular, what does it mean for the hypothesised types to be *optimal*?

Given the complexity of behaviours agents may exhibit, this is an extremely difficult question. In addition, it is not generally sufficient to consider types alone, since actions are planned with respect to both types *and* beliefs over types. Rather, we have to consider a stochastic process in which our actions depend on the correctness of types as well as the evolution of our beliefs.

In this spirit, we describe a formal methodology whereby we compare two interactive processes: one in which the true types are known, and one in which this knowledge is approximated through beliefs over hypothesised types. Based on these processes, we use a probabilistic temporal logic to define a hierarchy of desirable termination guarantees, and analyse the theoretical conditions under which they are met. The main result of this analysis is a novel characterisation of optimality which is based on the concept of probabilistic bisimulation (Larsen and Skou, 1991). In addition to concisely defining what constitutes optimality of hypothesised types, this allows the user to apply efficient model checking algorithms to verify optimality in practice.

7.1 Inaccurate Type Spaces

Each hypothesised type θ_j^* in Θ_j^* is a hypothesis by the user regarding how player j might behave. Therefore, Θ_j^* may be *inaccurate* in the sense that none of the types therein accurately predict the observed behaviour of player j. This is demonstrated in the following example:

Example 6. Consider a SBG with two players and actions L and R. Player 1 is controlled by HBA while player 2 has a single type, θ_{LR}, which chooses L,R,L,R, etc. HBA is provided with hypothesised types $\Theta_j^* = \{\theta_R^*, \theta_{LRR}^*\}$, where θ_R^* always chooses R while θ_{LRR}^* chooses L,R,R,L,R,R etc. Both hypothesised types are inaccurate in the sense that they predict player 2's actions in only $\approx 50\%$ of the game.

Two important theoretical questions in this context are how closely the hypothesised type spaces Θ_j^* have to approximate the true type spaces Θ_j^+ in order for HBA to be able to *(1)* solve the task (i.e. bring the SBG into a terminal state), and *(2)* achieve maximum payoffs. These questions are closely related to the notions of *flexibility* and *efficiency* (cf. Section 3.3) which, respectively, correspond to the probability of termination and the average payoff per time step. In this chapter, we are primarily concerned with question 1, and we are concerned with question 2 only insofar as we want to solve the task in minimal time. (Since reducing the time until termination will increase the average payoff per time step, i.e. efficiency.) This focus is formally captured by the following assumption, which we make throughout this chapter:

Assumption 4. Let player i be controlled by HBA, then $u_i(s,a) = 1$ iff. $s \in \bar{S}$, else 0.

Assumption 4 specifies that we are only interested in reaching a terminal state, since this is the only way to obtain a none-zero payoff. In our analysis, we consider discount factors γ (cf. Algorithm 2) with $\gamma = 1$ and $\gamma < 1$. While all our results hold for both cases, there is an important distinction: If $\gamma = 1$, then the expected payoffs (3.10) correspond to the actual probability that the following state can lead to (or is) a terminal state (we call this the *success rate*), whereas this is not necessarily the case if $\gamma < 1$. This is since $\gamma < 1$ tends to prefer shorter paths, which means that actions with lower success rates may be preferred if they lead to faster termination. Therefore, if $\gamma = 1$ then HBA is solely interested in termination, and if $\gamma < 1$ then it is interested in *fast* termination, where lower γ prefers faster termination.

7.2 Methodology of Analysis

Given a SBG Γ, we define the *ideal process*, X, as the process induced by Γ in which player i is controlled by HBA and in which HBA always knows the current and all future types of all players. Then, given a posterior formulation Pr and hypothesised type spaces Θ_j^* for all $j \neq i$, we define the *user process*, Y, as the process induced by Γ in which player i is controlled by HBA (same as in X) and in which HBA uses Pr and Θ_j^* in the usual way. Thus, the only difference between X and Y is that X can always predict the player types whereas Y approximates this knowledge through Pr and Θ_j^*. We write $E_{s^t}^{a_i}(H^t|C)$ to denote the expected payoff (as defined by (3.10)) of action a_i in state s^t after history H^t, in process $C \in \{X,Y\}$.

The idea is that X constitutes the ideal solution in the sense that $E_{s^t}^{a_i}(H^t|X)$ corresponds to the *actual* expected payoff, which means that HBA chooses the truly best-possible actions in X. This is opposed to $E_{s^t}^{a_i}(H^t|Y)$, which is merely the *estimated* expected payoff based on Pr and Θ_j^*, so that HBA may choose suboptimal actions in Y. The methodology of our analysis is to specify what relation Y must have to X to satisfy certain guarantees for termination.

We specify such guarantees in PCTL[1] (Hansson and Jonsson, 1994), a probabilistic modal logic which also allows for the specification of time constraints. PCTL expressions are interpreted over infinite histories in labelled transition systems with atomic propositions (i.e. Kripke structures). In order to interpret PCTL expressions over X and Y, we make the following modifications without loss of generality: Firstly, any terminal state $\bar{s} \in \bar{S}$ is an *absorbing* state, meaning that if a process is in $\bar{s}$, then the next state will be $\bar{s}$ with probability 1 and all players receive a zero payoff. Secondly, we introduce the atomic proposition `term` and label each terminal state with it, so that `term` is true in s if and only if $s \in \bar{S}$.

We will use the following two PCTL expressions:

$$F_{\succ p}^{\leq t}\texttt{term},\ F_{\succ p}^{<\infty}\texttt{term} \tag{7.1}$$

where $t \in \mathbb{N}$, $p \in [0,1]$, and $\succ \in \{>, \geq\}$.

$F_{\succ p}^{\leq t}\texttt{term}$ specifies that, given a state s, with a probability of $\succ p$ a state s' will be reached from s within t time steps such that s' satisfies `term`. The semantics of $F_{\succ p}^{<\infty}\texttt{term}$ is similar except that s' will be reached in arbitrary but finite time. We write $s \models_C \phi$ to say that a state s satisfies the PCTL expression ϕ in process $C \in \{X,Y\}$.

[1] PCTL stands for "Probabilistic real-time Computation Tree Logic" (Hansson and Jonsson, 1994).

7.3 Critical Type Spaces

In the following section, we will sometimes assume that the hypothesised type spaces Θ_j^* are *uncritical*:

Definition 17. The hypothesised type spaces Θ_j^* are *critical* if there is a set $S^c \subseteq S \setminus \bar{S}$ which satisfies all of the following:

1. For each $H^t \in \mathbb{H}$ with $s^t \in S^c$, there is $a_i \in A_i$ such that $E_{s^t}^{a_i}(H^t|Y) > 0$ and $E_{s^t}^{a_i}(H^t|X) > 0$.

2. There is a positive probability that Y may eventually get into a state $s^c \in S^c$ from the initial state s^0.

3. If Y is in a state in S^c, then with probability 1 it will always be in a state in S^c (i.e. it will never leave S^c).

We say Θ_j^* are *uncritical* if they are not critical.

Intuitively, critical type spaces have the potential to lead HBA into a state space in which it *believes* it chooses the right actions to solve the task, while other actions are *actually* required to solve the task. The only effect that its actions have is to induce an infinite cycle, due to a critical inconsistency between the hypothesised and true type spaces. The following example demonstrates this:

Example 7. Recall Example 6 and let the task be to choose the same action as player j. Then, Θ_j^* is uncritical because HBA will always solve the task at $t = 1$, regardless of its posterior beliefs and despite the fact that Θ_j^* is inaccurate. Now, assume that $\Theta_j^* = \{\theta_{RL}^*\}$ where θ_{RL}^* chooses actions R,L,R,L etc. Then, Θ_j^* is critical since HBA will always choose the opposite action of player j, thinking that it would solve the task, when a different action would actually solve it.

A practical way to ensure that the type spaces Θ_j^* are (eventually) uncritical is to include methods for opponent modelling in each Θ_j^*, such as the "conceptual types" in Chapter 4 (cf. Section 4.2.2). If the opponent models are guaranteed to learn the correct behaviours, then the type spaces Θ_j^* are guaranteed to become uncritical. In Example 7, any standard modelling method would eventually learn that the true strategy of player j is θ_{LR}. As the model becomes more accurate, the posterior beliefs gradually shift towards it and eventually allow HBA to take the right action.

7.4 Termination Guarantees

We will now define a series of increasingly desirable termination guarantees using the definitions of the previous sections, and show under what conditions they hold.

7.4.1 Guarantee 1

Our first termination guarantee states that if X has a positive probability of solving the task, then so does Y:

Property 1. $s^0 \models_X F_{>0}^{<\infty}\texttt{term} \Rightarrow s^0 \models_Y F_{>0}^{<\infty}\texttt{term}$

We can show that Property 1 holds if the hypothesised type spaces Θ_j^* are uncritical and if Y only chooses actions for player i with positive expected payoff in X.

Let $\mathbb{A}(H^t|C)$ denote the set of actions that process C may choose from in state s^t after history H^t, i.e. $\mathbb{A}(H^t|C) = \arg\max_{a_i} E_{s^t}^{a_i}(H^t|C)$ (cf. step 3 in Algorithm 2).

Theorem 4. Property 1 holds if Θ_j^* are uncritical and

$$\forall H^t \in \mathbb{H}\ \forall a_i \in \mathbb{A}(H^t|Y) : E_{s^t}^{a_i}(H^t|X) > 0 \tag{7.2}$$

Proof. Assume $s^0 \models_X F_{>0}^{<\infty}\texttt{term}$. Then, we know that X chooses actions a_i which *may* lead into a state s' such that $s' \models_X F_{>0}^{<\infty}\texttt{term}$, and the same holds for all such states s'. Now, given (7.2) it is tempting to infer the same result for Y, since Y only chooses actions a_i which have positive expected payoff in X and, therefore, could truly lead into a terminal state. However, (7.2) alone is not sufficient to infer $s' \models_Y F_{>0}^{<\infty}\texttt{term}$ because of the special case in which Y chooses actions a_i such that $E_{s^t}^{a_i}(H^t|X) > 0$ but without ever reaching a terminal state. This is why we require that the hypothesised type spaces Θ_j^* are uncritical, which prevents this special case. Thus, we can infer that $s' \models_Y F_{>0}^{<\infty}\texttt{term}$, and, hence, Property 1 holds. □

7.4.2 Guarantee 2

The second guarantee states that if X always solves the task, then so does Y:

Property 2. $s^0 \models_X F_{\geq 1}^{<\infty}\texttt{term} \Rightarrow s^0 \models_Y F_{\geq 1}^{<\infty}\texttt{term}$

We can show that Property 2 holds if the type spaces Θ_j^* are uncritical and if Y only chooses actions for player i which lead to states into which X may get as well.

Let $\mu(H^t, s|C)$ be the probability that process C transitions into state s from state s^t after history H^t, i.e.

$$\mu(H^t, s|C) = \frac{1}{|\mathbb{A}|} \sum_{a_i \in \mathbb{A}} \sum_{a_{-i} \in A_{-i}} T(s^t, \langle a_i, a_{-i} \rangle, s) \prod_{j \neq i} \pi_j(H^t, a_j, \theta_j^t) \tag{7.3}$$

with $\mathbb{A} \equiv \mathbb{A}(H^t|C)$, and let $\mu(H^t, S'|C) = \sum_{s \in S'} \mu(H^t, s|C)$ for $S' \subset S$.

Theorem 5. Property 2 holds if Θ_j^* are uncritical and

$$\forall H^t \in \mathbb{H}\ \forall s \in S : \mu(H^t, s|Y) > 0 \Rightarrow \mu(H^t, s|X) > 0 \tag{7.4}$$

Proof. The fact that $s^0 \models_X F_{\geq 1}^{<\infty}\texttt{term}$ means that, throughout the process, X only transitions into states s with $s \models_X F_{\geq 1}^{<\infty}\texttt{term}$. As before, it is tempting to infer the same result for Y based on (7.4), since it only transitions into states which have maximum success rate in X. However, (7.4) alone is not sufficient since Y may choose actions such that (7.4) holds true but Y will never reach a terminal state. Nevertheless, since the hypothesised type spaces Θ_j^* are uncritical, we know that this special case will not occur, and, thus, Property 2 holds. □

We note that, in both Properties 1 and 2, the reverse direction holds true regardless of Theorems 4 and 5. Furthermore, we can combine the requirements of Theorems 4 and 5 to ensure that both properties hold.

7.4.3 Guarantee 3

The third guarantee subsumes the previous guarantees by stating that X and Y have the same minimum probability of solving the task:

Property 3. $s^0 \models_X F_{\geq p}^{<\infty}\texttt{term} \Rightarrow s^0 \models_Y F_{\geq p}^{<\infty}\texttt{term}$

We can show that Property 3 holds if the hypothesised type spaces Θ_j^* are uncritical and if Y only chooses actions for player i which X might have chosen as well.

Let $R(a_i, H^t|C)$ be the *success rate* of action a_i, formally $R(a_i, H^t|C) = E_{s^t}^{a_i}(H^t|C)$ with $\gamma = 1$ (so that it corresponds to the actual *probability* with which a_i may lead to termination in the future). Define $X_{\min}$ and $X_{\max}$ to be the processes which for each H^t choose actions $a_i \in \mathbb{A}(H^t|X)$ with, respectively, minimal and maximal success rate $R(a_i, H^t|X)$.

Theorem 6. If Θ_j^* are uncritical and

$$\forall H^t \in \mathbb{H} : \mathbb{A}(H^t|Y) \subseteq \mathbb{A}(H^t|X) \tag{7.5}$$

then

(i) for $\gamma = 1$: Proposition 3 holds in both directions

(ii) for $\gamma < 1$: $s^0 \models_X F^{<\infty}_{\geq p}\texttt{term} \Rightarrow s^0 \models_Y F^{<\infty}_{\geq p'}\texttt{term}$

with $p_{\min} \leq q \leq p_{\max}$ for $q \in \{p, p'\}$, where $p_{\min}$ and $p_{\max}$ are the highest probabilities such that $s^0 \models_{X_{\min}} F^{<\infty}_{\geq p_{\min}}\texttt{term}$ and $s^0 \models_{X_{\max}} F^{<\infty}_{\geq p_{\max}}\texttt{term}$.

Proof. (i): Since $\gamma = 1$, all actions $a_i \in \mathbb{A}(H^t|X)$ have the same success rate for a given H^t, and given (7.5) we know that Y's actions always have the same success rate as X's actions. Provided that the type spaces Θ^*_j are uncritical, we can conclude that Property 3 must hold, and for the same reasons the reverse direction must hold as well.

(ii): Since $\gamma < 1$, the actions $a_i \in \mathbb{A}(H^t|X)$ may have different success rates. The lowest and highest chances that X solves the task are precisely modelled by $X_{\min}$ and $X_{\max}$, and given (7.5) and the fact that Θ^*_j are uncritical, the same holds for Y. Therefore, we can infer the common bound $p_{\min} \leq \{p, p'\} \leq p_{\max}$ as defined in Theorem 6. □

7.4.4 Guarantee 4 (Strong Optimality)

Properties 1 to 3 are *indefinite* in the sense that they make no restrictions on time requirements. Our fourth and final guarantee subsumes all previous guarantees and states that if there is a probability p such that X terminates *within* t time steps, then so does Y for the same p and t:

Property 4. $s^0 \models_X F^{\leq t}_{\geq p}\texttt{term} \Rightarrow s^0 \models_Y F^{\leq t}_{\geq p}\texttt{term}$

We believe that Property 4 is an adequate criterion of *optimality* for hypothesised type spaces Θ^*_j since, if it holds, Θ^*_j must approximate the true type spaces Θ^+_j in a way which allows HBA to plan (almost) as accurately — in terms of solving the task — as the "ideal" HBA in X which always knows the true types.

What relation must Y have to X in order to satisfy Property 4? The fact that Y and X are processes over state transition systems means that we can draw on methods from the model checking literature to answer this question. Specifically, we will use the concept of *probabilistic bisimulation* (Larsen and Skou, 1991), which we here define within the context of our work:

Definition 18. A *probabilistic bisimulation* between X and Y is an equivalence relation $B \subseteq S \times S$ such that

(i) $(s^0, s^0) \in B$

(ii) $s_X \models_X \texttt{term} \Leftrightarrow s_Y \models_Y \texttt{term}$ for all $(s_X, s_Y) \in B$

(iii) $\mu(H_X^t, \hat{S}|X) = \mu(H_Y^t, \hat{S}|Y)$ for any histories H_X^t, H_Y^t with $(s_X^t, s_Y^t) \in B$ and all equivalence classes $\hat{S}$ under B.

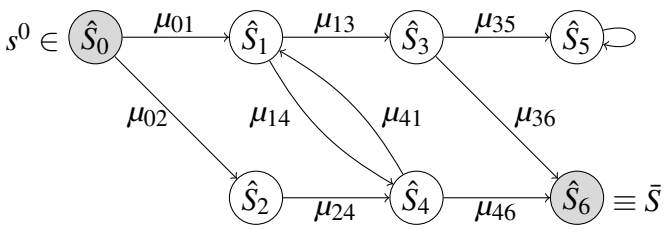
Intuitively, a probabilistic bisimulation states that X and Y do (on average) match each other's transitions. Our definition of probabilistic bisimulation is most general in that it does not require that transitions are matched by the same action or that related states satisfy the same atomic propositions other than termination. However, we do note that other definitions exist which make such additional requirements, and our results hold for each of these refinements.

The main contribution in this chapter is to show that the optimality criterion expressed by Property 4 holds in *both directions* if there exists a probabilistic bisimulation between X and Y. Thus, we offer an alternative formal characterisation of optimality for the hypothesised type spaces Θ_j^*:

Theorem 7. Property 4 holds in both directions if there is a probabilistic bisimulation between X and Y.

Proof. First of all, we note that, strictly speaking, the standard definitions of bisimulation (e.g. Baier, 1996; Larsen and Skou, 1991) assume the Markov property, which means that the next state of a process depends only on the current state of the process. In contrast, we consider the more general case in which the next state may depend on the history H^t of previous states and joint actions (since the player strategies π_j depend on H^t). However, one can always enforce the Markov property *by design*, i.e. by augmenting the state space S to account for the relevant factors of the past. In fact, we could postulate that the histories as a whole constitute the states of the system, i.e. $S = \mathbb{H}$. Therefore, to simplify the exposition, we assume the Markov property and we write $\mu(s, \hat{S}|C)$ to denote the cumulative probability that C transitions from state s into any state in $\hat{S}$.

Given the Markov property, the fact that B is an equivalence relation, and $\mu(s_X, \hat{S}|X) = \mu(s_Y, \hat{S}|Y)$ for $(s_X, s_Y) \in B$, we can represent the dynamics of X and Y in a common graph, such as the following one:



The nodes correspond to the equivalence classes under B. A directed edge from $\hat{S}_a$

to $\hat{S}_b$ specifies that there is a positive probability $\mu_{ab} = \mu(s_X, \hat{S}_b|X) = \mu(s_Y, \hat{S}_b|Y)$ that X and Y transition from states $s_X, s_Y \in \hat{S}_a$ to states $s'_X, s'_Y \in \hat{S}_b$. Note that s_X, s_Y and s'_X, s'_Y need not be equal but merely equivalent, i.e. $(s_X, s_Y) \in B$ and $(s'_X, s'_Y) \in B$. There is one node ($\hat{S}_0$) that contains the initial state s^0 and one node ($\hat{S}_6$) that contains all terminal states $\bar{S}$ and no other states. This is because once X and Y reach a terminal state they will always stay in it (i.e. $\mu(s, \bar{S}|X) = \mu(s, \bar{S}|Y) = 1$ for $s \in \bar{S}$) and since they are the only states that satisfy `term`. Thus, the graph starts in $\hat{S}_0$ and terminates (if at all) in $\hat{S}_6$.

Since the graph represents the dynamics of both X and Y, it is easy to see that Property 4 must hold in both directions. In particular, the probabilities that X and Y are in node $\hat{S}$ at time t are identical. One simply needs to add the probabilities of all directed paths of length t which end in $\hat{S}$ (provided that such paths exist), where the probability of a path is the product of the μ_{ab} along the path. Therefore, X and Y terminate with equal probability, and on average within the same number of time steps. □

Some remarks to clarify the usefulness of this result: First of all, in contrast to Theorems 4 to 6, Theorem 7 does not explicitly require Θ^*_j to be uncritical. In fact, this is implicit in the definition of probabilistic bisimulation. Moreover, while the other theorems relate Y and X for identical histories H^t, Theorem 7 relates Y and X for *related* histories H^t_Y and H^t_X, making it more generally applicable. Finally, Theorem 7 has an important practical implication: it tells us that we can use efficient methods for model checking (e.g. Baier, 1996; Larsen and Skou, 1991) to verify optimality of Θ^*_j. In fact, it can be shown that for Property 4 to hold (albeit not in the other direction) it suffices that Y be a *probabilistic simulation* (Baier, 1996) of X, which is a coarser preorder than probabilistic bisimulation. However, algorithms for checking probabilistic simulation (e.g. Baier, 1996) are computationally much more expensive (and fewer) than those for probabilistic bisimulation, hence their practical use is currently limited.

Chapter 8

Behavioural Hypothesis Testing

In the previous chapter, we considered the possibility of incorrect hypothesised types and analysed the conditions under which HBA is still able to solve its task, despite the incorrectness of types. While the analysis is rigorous and complete, it is performed *before* any interaction and with respect to the true types of other agents. How can we decide *during* the interaction whether to trust our hypothesised types?

There are several ways in which an answer to this question could be utilised. For example, if we persistently reject our hypothesised types, we may hypothesise an alternative set of types or resort to some default plan of action, such as a "maximin" strategy. Unfortunately, posterior beliefs do not provide an answer to this question because they quantify the *relative* likelihood of types (relative to a set of alternative types), but they are no measure of truth. That is, even if our beliefs point to one type, this does not tell us that the observed agent is indeed of that type. Instead, it only tells us that all other types have been discarded after the current interaction history.

In this chapter, we propose an automated statistical analysis to address this question. The analysis can be applied to individual types or to the combination of beliefs and types. By sampling "hypothetical" actions at each point in time, we can formulate the two-sample problem of whether the hypothetical and real actions were generated by the same type. We present an algorithm which decides this problem in the form of a frequentist hypothesis test. The algorithm can incorporate multiple statistical criteria into the test statistic and learns its distribution during the interaction, with asymptotic correctness guarantees. Analogous to standard frequentist testing, the hypothesis is rejected at a given point in time if the resulting p-value is below some "significance level". We present results from a comprehensive set of experiments, demonstrating that the algorithm achieves high accuracy and scalability at low computational costs.

8.1 The Difficulty of Behavioural Hypothesis Testing

Deciding whether a behavioural hypothesis is incorrect can be an extremely difficult problem. To illustrate the source of difficulty, consider an interaction process between two agents which can choose from three actions. Table 8.1 shows the first 5 time steps of the interaction. The columns show, respectively, the current time t of the interaction, the actions chosen by the agents at time t, and agent 1's hypothesised probabilities with which agent 2 would choose its actions at time t, based on the prior interaction history.

t	(a_1^t, a_2^t)	θ_2^*
1	$(1,2)$	$\langle .3, .1, .6 \rangle$
2	$(3,1)$	$\langle .2, .3, .5 \rangle$
3	$(2,3)$	$\langle .7, .1, .2 \rangle$
4	$(2,3)$	$\langle .0, .4, .6 \rangle$
5	$(1,2)$	$\langle .4, .2, .4 \rangle$

Table 8.1: Beginning of interaction process.

Assuming the process continues in this fashion, and without any restrictions on the behaviour of agent 2, how should agent 1 decide whether or not to reject its hypothesis about the behaviour of agent 2? Note that agent 1 cannot outright reject its hypothesis because all observed actions of agent 2 were supported by agent 1's hypothesis.

A natural way to address this question is to compute some kind of *score* from the information given in the above table, and to compare this score with some manually chosen rejecting threshold. A prominent example of such a score is the empirical frequency distribution (e.g. Conitzer and Sandholm, 2007; Foster and Young, 2003). However, while the simplicity of this method is appealing, there are two significant problems: (1) it is far from trivial to devise a scoring scheme that reliably quantifies "correctness" of hypotheses (for instance, an empirical frequency distribution taken over all past actions would be insufficient in the above example since the hypothesised action distributions are changing), and (2) it is unclear how one should choose the threshold parameter for any given scoring scheme.

In this chapter, we present an efficient algorithm which decides this question in the form of a frequentist hypothesis test. The algorithm addresses (1) by allowing for multiple scoring criteria in the construction of the test statistic, with the intent of obtaining an overall more reliable scoring scheme. The distribution of the test statistic is learned during the interaction process, and we show that the learning is asymptotically

correct. Analogous to standard frequentist testing, the hypothesis is rejected at a given point in time if the resulting p-value is below some "significance level". This eliminates (2) by providing a uniform semantics for rejection that is invariant to the employed scoring scheme.

Of course, there is a long-standing debate on the role of statistical hypothesis tests and quantities such as p-values (e.g. Gelman and Shalizi, 2013; Berger and Sellke, 1987; Cox, 1977). The usual consensus is that p-values should be combined with other forms of evidence to reach a final conclusion (Fisher, 1935), and this is the view we adopt as well. In this sense, our method may be used as part of a larger machinery to decide the truth of a hypothesis.

8.2 Individual Hypotheses and Beliefs

Recall from Chapter 3 that $\pi_j(H^t, a_j, \theta_j)$ denotes the probability with which player j chooses action $a_j \in A_j$ after the interaction history H^t, if player j is of type $\theta_j \in \Theta_j$. Furthermore, recall that we use a set $\Theta_j^* \subset \Theta_j$ of hypothesised types $\theta_j^* \in \Theta_j^*$ over which we maintain posterior beliefs (probabilities) $\Pr_j(\theta_j^* | H^t)$.

As noted in Chapter 7, it does not generally suffice to consider the correctness of individual types, since we plan our actions with respect to both types *and* our beliefs regarding the relative likelihood of types (cf. (3.10)). In this regard, we note that any combination of beliefs Pr and types Θ_j^* can be described as a single type $\hat{\theta}_j^*$ of the form

$$\pi_j(H^t, a_j, \hat{\theta}_j^*) = \sum_{\theta_j^* \in \Theta_j^*} \Pr(\theta_j^* | H^t)\, \pi_j(H^t, a_j, \theta_j^*). \tag{8.1}$$

This combination is equivalent to sampling a single type $\theta_j^* \in \Theta_j^*$ using probabilities $\Pr(\theta_j^* | H^t)$, and then using θ_j^* to choose actions $a_j \in A_j$ via $\pi_j(H^t, a_j, \theta_j^*)$ (Kuhn, 1953). Analogously, we may combine the true types $\Theta_j^+ \subset \Theta_j$ of player j, using the type distribution Υ, as a single type $\hat{\theta}_j^+$ such that

$$\pi_j(H^t, a_j, \hat{\theta}_j^+) = \sum_{\theta_j^+ \in \Theta_j^+} \Upsilon(t, \theta_j^+)\, \pi_j(H^t, a_j, \theta_j^+). \tag{8.2}$$

Therefore, to simplify the notation in this chapter, we will generally assume a single hypothesised type $\theta_j^* \in \Theta_j$ and a single true type $\theta_j^+ \in \Theta_j$. Note that this means that our method can be applied to the combination of beliefs and hypothesised types, as well as to individual types in Θ_j^*. Furthermore, we will write $\pi_j(H^t, \theta_j)$ to denote the probability distribution over actions A_j (rather than probabilities of individual actions).

8.3 A Method for Behavioural Hypothesis Testing

Let i denote our agent and let j denote another agent. Moreover, let $\theta_j^* \in \Theta_j$ denote our hypothesis for j's behaviour and let $\theta_j^+ \in \Theta_j$ denote j's true behaviour. The central question we ask is if $\theta_j^* = \theta_j^+$?

Unfortunately, since we do not know θ_j^+, we cannot directly answer this question. However, at each time t, we know j's past actions $\mathbf{a}_j^t = (a_j^0, ..., a_j^{t-1})$ which were generated by θ_j^+. If we use θ_j^* to generate a vector $\hat{\mathbf{a}}_j^t = (\hat{a}_j^0, ..., \hat{a}_j^{t-1})$, where $\hat{a}_j^\tau$ is sampled using $\pi_j(H^\tau, \theta_j^*)$, we can formulate the related two-sample problem of whether $\mathbf{a}_j^t$ and $\hat{\mathbf{a}}_j^t$ were generated from the same behaviour, namely θ_j^*.

In this section, we propose a general and efficient algorithm to decide this problem. At its core, the algorithm computes a frequentist p-value

$$p = P\left(|\mathrm{T}(\tilde{\mathbf{a}}_j^t, \hat{\mathbf{a}}_j^t)| \geq |\mathrm{T}(\mathbf{a}_j^t, \hat{\mathbf{a}}_j^t)|\right) \tag{8.3}$$

where $\tilde{\mathbf{a}}_j^t \sim \delta^t(\theta_j^*) = \left(\pi_j(H^0, \theta_j^*), ..., \pi_j(H^{t-1}, \theta_j^*)\right)$. The value of p corresponds to the probability with which we expect to observe a test statistic at least as extreme as $\mathrm{T}(\mathbf{a}_j^t, \hat{\mathbf{a}}_j^t)$, under the null-hypothesis that $\theta_j^* = \theta_j^+$. Thus, we reject θ_j^* if p is below some "significance level" α^*.

In the following subsections, we describe the test statistic T and its asymptotic properties, and how our algorithm learns the distribution of $\mathrm{T}(\tilde{\mathbf{a}}_j^t, \hat{\mathbf{a}}_j^t)$. A summary of the algorithm is given in Algorithm 5.

8.3.1 Test Statistic

We follow the general approach outlined in Section 8.1 by which we compute a *score* from a vector of actions and their hypothesised distributions. Formally, we define a *score function* as $z : (A_j)^t \times \Delta(A_j)^t \to \mathbb{R}$, where $\Delta(A_j)$ is the set of all probability distributions over A_j. Thus, $z(\mathbf{a}_j^t, \delta^t(\theta_j^*))$ is the score for observed actions $\mathbf{a}_j^t$ and hypothesised distributions $\delta^t(\theta_j^*)$, and we sometimes abbreviate this to $z(\mathbf{a}_j^t, \theta_j^*)$. We use Z to denote the space of all score functions.

Given a score function z, we define the test statistic T as

$$\mathrm{T}(\tilde{\mathbf{a}}_j^t, \hat{\mathbf{a}}_j^t) = \frac{1}{t}\sum_{\tau=1}^{t} \mathrm{T}_\tau(\tilde{\mathbf{a}}_j^\tau, \hat{\mathbf{a}}_j^\tau) \tag{8.4}$$

$$\mathrm{T}_\tau(\tilde{\mathbf{a}}_j^\tau, \hat{\mathbf{a}}_j^\tau) = z(\tilde{\mathbf{a}}_j^\tau, \theta_j^*) - z(\hat{\mathbf{a}}_j^\tau, \theta_j^*) \tag{8.5}$$

where $\tilde{\mathbf{a}}_j^\tau$ and $\hat{\mathbf{a}}_j^\tau$ are the τ-prefixes of $\tilde{\mathbf{a}}_j^t$ and $\hat{\mathbf{a}}_j^t$, respectively.

Algorithm 5 Automatic behavioural hypothesis testing

Input: history H^t (including observed action a_j^{t-1})

Output: p-value (reject θ_j^* if p below some threshold α^*)

Parameters: hypothesis θ_j^*; score functions $z_1, ..., z_K$; $N > 0$

// Expand action vectors

Set $\mathbf{a}_j^t \leftarrow \langle \mathbf{a}_j^{t-1}, a_j^{t-1} \rangle$

Sample $\hat{a}_j^{t-1} \sim \pi_j(H^{t-1}, \theta_j^*)$; set $\hat{\mathbf{a}}_j^t \leftarrow \langle \hat{\mathbf{a}}_j^{t-1}, \hat{a}_j^{t-1} \rangle$

for $n = 1, ..., N$ **do**

  Sample $\tilde{a}_j^{t-1} \sim \pi_j(H^{t-1}, \theta_j^*)$; set $\tilde{\mathbf{a}}_j^{t,n} \leftarrow \langle \tilde{\mathbf{a}}_j^{t-1,n}, \tilde{a}_j^{t-1} \rangle$

end for

// Fit skew-normal distribution f

if update parameters? **then**

  Compute $D \leftarrow \left\{ \mathrm{T}(\tilde{\mathbf{a}}_j^{t,n}, \hat{\mathbf{a}}_j^t) \mid n = 1, ..., N \right\}$

  Fit ξ, ω, β to D, e.g. using (8.14)

  Find mode μ from ξ, ω, β

end if

// Compute p-value

Compute $q \leftarrow \mathrm{T}(\mathbf{a}_j^t, \hat{\mathbf{a}}_j^t)$ using (8.4)/(8.7)

return $p \leftarrow f(q \mid \xi, \omega, \beta) \;/\; f(\mu \mid \xi, \omega, \beta)$

In this work, we assume that z is provided by the user. While formally unnecessary (in the sense that our analysis does not require it), we find it a useful design guideline to interpret a score as a kind of likelihood, such that higher scores suggest higher likelihood of θ_j^* being correct. Under this interpretation, a minimum requirement for z should be that it is *consistent*, such that, for any $t > 0$ and $\theta_j^* \in \Theta_j$,

$$\theta_j^* \in \Pi^z = \arg\max_{\theta'_j \in \Theta_j} \mathbb{E}_{\mathbf{a}'_j \sim \delta^t(\theta_j^*)} \left[z(\mathbf{a}'_j, \theta'_j) \right] \tag{8.6}$$

where $\mathbb{E}_\eta$ denotes the expectation under η. This ensures that if the null-hypothesis $\theta_j^* = \theta_j^+$ is true, then the score $z(\mathbf{a}_j^t, \theta_j^*)$ is maximised on expectation.

Ideally, we would like a score function z which is *perfect* in that it is consistent and $|\Pi^z| = 1$. This means that θ_j^* can maximise $z(\mathbf{a}_j^t, \theta_j^*)$ (where $\mathbf{a}_j^t \sim \delta^t(\theta_j^+)$) *only* if $\theta_j^* = \theta_j^+$. Unfortunately, it is unclear if such a score function exists for the general case and how it should look. Even if we restrict the behaviours agents may exhibit, it can still be difficult to find a perfect score function. On the other hand, it is a relatively simple task to specify a small set of score functions $z_1, ..., z_K$ which are consistent but imperfect.

(Examples are given in Section 8.4.) Given that these score functions are consistent, we know that the cardinality $|\cap_k \Pi^{z_k}|$ can only monotonically decrease. Therefore, it seems a reasonable approach to combine multiple imperfect score functions in an attempt to approximate a perfect score function.

Of course, we could simply define z as a linear (or otherwise) combination of $z_1, ..., z_K$. However, this approach is at risk of losing information from the individual scores, e.g. due to commutativity and other properties of the combination. Thus, we instead propose to compare the scores individually. Given score functions $z_1, ..., z_K \in Z$ which are all bounded by the same interval $[a,b] \subset \mathbb{R}$, we redefine T_τ to

$$\mathrm{T}_\tau(\tilde{\mathbf{a}}_j^\tau, \hat{\mathbf{a}}_j^\tau) = \sum_{k=1}^{K} w_k \left(z_k(\tilde{\mathbf{a}}_j^\tau, \theta_j^*) - z_k(\hat{\mathbf{a}}_j^\tau, \theta_j^*) \right) \tag{8.7}$$

where $w_k \in \mathbb{R}$ is a weight for score function z_k. In this work, we set $w_k = \frac{1}{K}$. (We also experiment with alternative weighting schemes in Section 8.4.) However, we believe that w_k may serve as an interface for useful modifications of our algorithm. For example, Yue et al. (2010) compute weights to increase the power of their hypothesis tests.

8.3.2 Asymptotic Properties

The vectors $\mathbf{a}_j^t$ and $\hat{\mathbf{a}}_j^t$ are constructed iteratively. That is, at time t, we observe agent j's past action a_j^{t-1}, which was generated from $\pi_j(H^{t-1}, \theta_j^+)$, and set $\mathbf{a}_j^t = \langle \mathbf{a}_j^{t-1}, a_j^{t-1} \rangle$. At the same time, we sample an action $\hat{a}_j^{t-1}$ using $\pi_j(H^{t-1}, \theta_j^*)$ and set $\hat{\mathbf{a}}_j^t = \langle \hat{\mathbf{a}}_j^{t-1}, \hat{a}_j^{t-1} \rangle$. Assuming the null-hypothesis $\theta_j^* = \theta_j^+$, will $\mathrm{T}(\mathbf{a}_j^t, \hat{\mathbf{a}}_j^t)$ converge in the process?

Unfortunately, T might not converge. This may seem surprising at first glance given that a_j^{t-1} and $\hat{a}_j^{t-1}$ have the same distribution $\pi_j(H^{t-1}, \theta_j^+) = \pi_j(H^{t-1}, \theta_j^*)$, since $\mathbb{E}_{x,y\sim\psi}[x-y] = 0$ for any distribution ψ. However, there is a subtle but important difference: while a_j^{t-1} and $\hat{a}_j^{t-1}$ have the same distribution, $z_k(\mathbf{a}_j^t, \theta_j^*)$ and $z_k(\hat{\mathbf{a}}_j^t, \theta_j^*)$ may have arbitrarily different distributions. This is because these scores may depend on the entire prefix vectors $\mathbf{a}_j^{t-1}$ and $\hat{\mathbf{a}}_j^{t-1}$, respectively, which means that their distributions may be different if $\mathbf{a}_j^{t-1} \neq \hat{\mathbf{a}}_j^{t-1}$. Fortunately, our algorithm does not require T to converge because it learns the distribution of T during the interaction process, as we will discuss in Section 8.3.3.

Interestingly, while T may not converge, it can be shown that the fluctuation of T is eventually normally distributed, for any set of score functions $z_1, ..., z_K$ with bound $[a,b]$. Formally, let $\mathbb{E}[\mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau)]$ and $\mathrm{Var}[\mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau)]$ denote the finite expectation and variance of $\mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau)$, where it is irrelevant if $\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau$ are sampled directly from $\delta^\tau(\theta_j^*)$

or generated iteratively as prescribed above. Furthermore, let $\sigma_t^2 = \sum_{\tau=1}^{t} \mathrm{Var}[\mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau)]$ denote the cumulative variance. Then, the standardised stochastic sum

$$\frac{1}{\sigma_t} \sum_{\tau=1}^{t} \mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau) - \mathbb{E}[\mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau)] \tag{8.8}$$

will converge in distribution to the standard normal distribution as $t \to \infty$. Thus, T is normally distributed as well.

To see this, first recall that the standard central limit theorem requires the random variables T_τ to be independent and identically distributed. In our case, T_τ are independent in that the random outcome of T_τ has no effect on the outcome of $T_{\tau'}$. However, T_τ and $T_{\tau'}$ depend on different action sequences, and may therefore have different distributions. Hence, we have to show an additional property, commonly known as *Lyapunov's condition* (e.g. Fischer, 2010), which states that there exists a positive integer d such that

$$\lim_{t\to\infty} \frac{\hat{\sigma}_t^{2+d}}{\sigma_t^{2+d}} = 0, \text{ with} \tag{8.9}$$

$$\hat{\sigma}_t^{2+d} = \sum_{\tau=1}^{t} \mathbb{E}\left[\left| \mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau) - \mathbb{E}[\mathrm{T}_\tau(\mathbf{a}_j^\tau, \hat{\mathbf{a}}_j^\tau)] \right|^{2+d} \right]. \tag{8.10}$$

Since z_k are bounded, we know that T_τ are bounded. Hence, the summands in (8.10) are uniformly bounded, say by U for brevity. Setting $d = 1$, we obtain

$$\lim_{t\to\infty} \frac{\hat{\sigma}_t^3}{\sigma_t^3} \leq \frac{U\hat{\sigma}_t^2}{\sigma_t^3} = \frac{U}{\sigma_t} \tag{8.11}$$

The last part goes to zero if $\sigma_t \to \infty$, and hence Lyapunov's condition holds. If, on the other hand, σ_t converges, then this means that the variance of T_τ is zero from some point onward (or that it has an appropriate convergence to zero). From this point, θ_j^* prescribes fully deterministic action choices for agent j (i.e. $\exists a_j : \pi_j(H^\tau, a_j, \theta_j^*) = 1$), and a statistical analysis is no longer necessary.

8.3.3 Learning the Test Distribution

Given that T is eventually normal, it may seem reasonable to compute (8.3) using a normal distribution whose parameters are fitted during the interaction. However, this fails to recognise that the distribution of T is shaped *gradually* over an extended time period, and that the fluctuation around T can be heavily skewed in either direction until convergence to a normal distribution emerges. Thus, a normal distribution may be a poor fit during this shaping period.

What is needed is a distribution which can represent any normal distribution, and which is flexible enough to faithfully represent the gradual shaping. One distribution which has these properties is the *skew-normal distribution* (Azzalini, 1985; O'Hagan and Leonard, 1976). Given the PDF ϕ and CDF Φ of the standard normal distribution, the skew-normal PDF is defined as

$$f(x \mid \xi, \omega, \beta) = \frac{2}{\omega}\phi\left(\frac{x-\xi}{\omega}\right)\Phi\left(\beta\left(\frac{x-\xi}{\omega}\right)\right) \tag{8.12}$$

where $\xi \in \mathbb{R}$ is the location parameter, $\omega \in \mathbb{R}^+$ is the scale parameter, and $\beta \in \mathbb{R}$ is the shape parameter. Note that this reduces to the normal PDF for $\beta = 0$, in which case ξ and ω correspond to the mean and standard deviation, respectively. Hence, the normal distribution is a sub-class of the skew-normal distribution.

Our algorithm learns the shifting parameters of f during the interaction process, using a simple but effective sampling procedure. Essentially, we use θ_j^* to iteratively generate N additional action vectors $\tilde{\mathbf{a}}_j^{t,1}, ..., \tilde{\mathbf{a}}_j^{t,N}$ in the exact same way as $\hat{\mathbf{a}}_j^t$. The vectors $\tilde{\mathbf{a}}_j^{t,n}$ are then mapped into data points

$$D = \left\{ \mathrm{T}(\tilde{\mathbf{a}}_j^{t,n}, \hat{\mathbf{a}}_j^t) \mid n = 1, ..., N \right\} \tag{8.13}$$

which are used to estimate the parameters ξ, ω, β by minimising the negative log-likelihood

$$N\log(\omega) - \sum_{x \in D} \log\phi\left(\frac{x-\xi}{\omega}\right) + \log\Phi\left(\beta\left(\frac{x-\xi}{\omega}\right)\right) \tag{8.14}$$

whilst ensuring that ω is positive. An alternative is the method-of-moments estimator, which can also be used to obtain initial values for (8.14). Note that it is usually unnecessary to estimate the parameters at every point in time; it seems reasonable to update the parameters less frequently as the amount of evidence (i.e. observed actions) grows.

Given the asymmetry of the skew-normal distribution, the semantics of "as extreme as" in (8.3) may no longer be obvious (e.g. is this with respect to the mean or mode?). In addition, the usual tail-area calculation of the p-value requires the CDF, but there is no closed form for the skew-normal CDF and approximating it is rather cumbersome. To circumvent these issues, we approximate the p-value as

$$p \;\approx\; \frac{f(\mathrm{T}(\mathbf{a}_j^t, \hat{\mathbf{a}}_j^t) \mid \xi, \omega, \beta)}{f(\mu \mid \xi, \omega, \beta)} \tag{8.15}$$

where μ is the mode of the fitted skew-normal distribution. This avoids the asymmetry issue and is easier to compute.

8.4 Experiments

We conducted a comprehensive set of experiments to investigate the accuracy (correct and incorrect rejection), scalability (with number of actions), and sampling complexity of our algorithm. The following three score functions and their combinations were used:

$$z_1(\mathbf{a}_j^t, \theta_j^*) = \frac{1}{t}\sum_{\tau=0}^{t-1} \frac{\pi_j(H^\tau, a_j^\tau, \theta_j^*)}{\max_{a_j \in A_j} \pi_j(H^\tau, a_j, \theta_j^*)} \tag{8.16}$$

$$z_2(\mathbf{a}_j^t, \theta_j^*) = \frac{1}{t}\sum_{\tau=0}^{t-1} 1 - \mathbb{E}_{a_j \sim \pi_j(H^\tau, \theta_j^*)} \left| \pi_j(H^\tau, a_j^\tau, \theta_j^*) - \pi_j(H^\tau, a_j, \theta_j^*) \right| \tag{8.17}$$

$$z_3(\mathbf{a}_j^t, \theta_j^*) = \sum_{a_j \in A_j} \min \left[\frac{1}{t}\sum_{\tau=0}^{t-1} [a_j^\tau = a_j]_1, \frac{1}{t}\sum_{\tau=0}^{t-1} \pi_j(H^\tau, a_j, \theta_j^*) \right] \tag{8.18}$$

where $[b]_1 = 1$ if b is true and 0 otherwise. Intuitively, z_1 measures the average ratio of probability mass assigned to observed actions over most likely actions under θ_j^*; z_2 measures the average expected (absolute) difference between probability mass assigned to observed actions and sampled actions under θ_j^* (subtracted from 1 to convert to likelihood); and z_3 measures the overlap between the empirical frequency distribution of observed actions (cf. Section 8.1) and the averaged predicted distributions of θ_j^*. All score functions follow the likelihood design principle outlined in Section 8.3.1 (i.e. higher scores suggest higher likelihood of θ_j^* being correct). Note that z_1 and z_3 are generally consistent (cf. Section 8.3.1), while z_2 is consistent for $|A_j| = 2$ but not necessarily for $|A_j| > 2$. Furthermore, z_1, z_2, z_3 are all imperfect.

The parameters of the test distribution (cf. Section 8.3.3) were estimated less frequently as t increased. The first estimation was performed at time $t = 1$ (i.e. after observing one action). After estimating the parameters at time t, we waited $\lfloor \sqrt{t} \rfloor - 1$ time steps until the parameters were re-fitted. Throughout our experiments, we used a significance level of $\alpha^* = 0.01$ (i.e. reject θ_j^* if the p-value is below 0.01).

8.4.1 Random Behaviours

In the first set of experiments, the behaviour (type) spaces Θ_i and Θ_j were restricted to "random" behaviours. Each random behaviour is defined by a sequence of random probability distributions over A_j. The distributions are created by drawing uniform random numbers from $(0, 1)$ for each action $a_j \in A_j$, and subsequent normalisation so that the values sum up to 1.

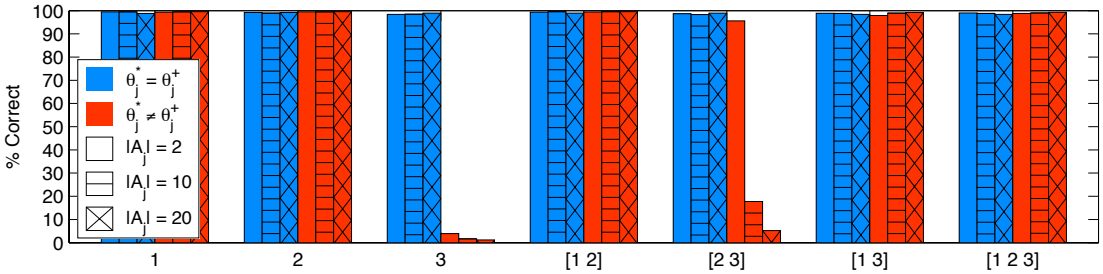


Figure 8.1: Average accuracy with random behaviours, for $N = 50$ and $|A_j| = 2, 10, 20$. Results averaged over 500 processes with 10000 time steps, for $\theta^*_j = \theta^+_j$ and $\theta^*_j \neq \theta^+_j$ each. The x-axis shows score functions z_k used in test statistic.

Random behaviours are a good baseline for our experiments because they are usually hard to distinguish. This is due to the fact that the entire set A_j is always in the support of the behaviours, and since they do not react to any past actions. These properties mean that there is little structure in the interaction that can be used to distinguish behaviours.

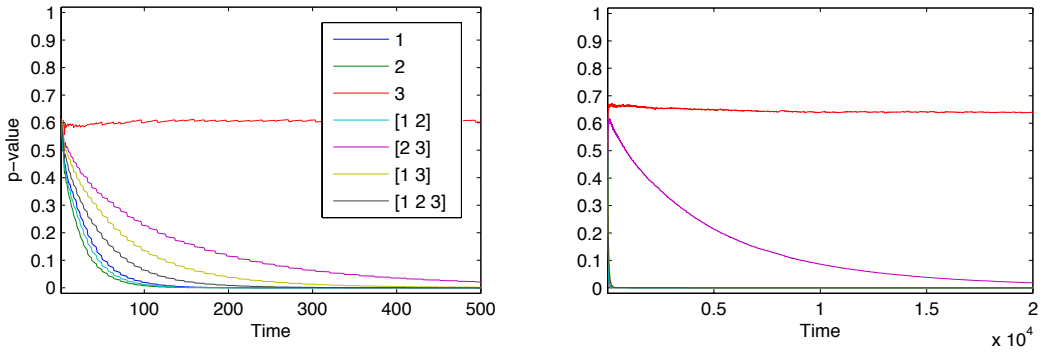
We simulated 1000 interaction processes, each lasting 10000 time steps. In each process, we randomly sampled behaviours $\theta_i \in \Theta_i, \theta^+_j \in \Theta_j$ to control agents i and j, respectively. In half of these processes, we used a correct hypothesis $\theta^*_j = \theta^+_j$. In the other half, we sampled a random hypothesis $\theta^*_j \in \Theta_j$ with $\theta^*_j \neq \theta^+_j$. We repeated each set of simulations for $|A_j| = 2, 10, 20$ (with $|A_i| = |A_j|$) and $N = 10, 50, 100$ (cf. Section 8.3.3).

8.4.1.1 Accuracy & Scalability

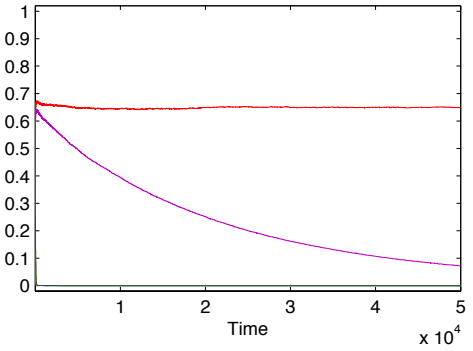
Figure 8.1 shows the average accuracy of our algorithm (for $N = 50$), by which we mean the average percentage of time steps in which the algorithm made correct decisions (i.e. no reject if $\theta^*_j = \theta^+_j$; reject if $\theta^*_j \neq \theta^+_j$). The x-axis shows the combination of score functions used to compute the test statistic (e.g. [1 2] means that we combined z_1, z_2).

The results show that our algorithm achieved excellent accuracy, often bordering the 100% mark. They also show that the algorithm scaled well with the number of actions, with no degradation in accuracy. However, there were two exceptions to these observations: Firstly, using z_3 resulted in very poor accuracy for $\theta^*_j \neq \theta^+_j$. Secondly, the combination of z_2, z_3 scaled badly for $\theta^*_j \neq \theta^+_j$.

The reason for both of these exceptions is that z_3 is not a good scoring scheme for random behaviours. The function z_3 quantifies a similarity between the empirical frequency distribution and the averaged hypothesised distributions. For random behaviours (as defined in this work), both of these distributions will converge to the uniform distri-



(a) $|A_j| = 2$ (b) $|A_j| = 10$



(c) $|A_j| = 20$

Figure 8.2: Average p-values with random behaviours, for $N = 50$ and $\theta_j^* \neq \theta_j^+$ (i.e. hypothesis wrong). Results averaged over 500 processes. The legend shows the score functions z_k used in test statistic.

bution. Thus, under z_3, any two random behaviours will eventually be the same, which explains the low accuracy for $\theta_j^* \neq \theta_j^+$.

As can be seen in Figure 8.1, the inadequacy of z_3 is solved when adding any of the other score functions z_1, z_2. These functions add discriminative information to the test statistic, which technically means that the cardinality $|\Pi^z|$ in (8.6) is reduced. However, in the case of $[z_2, z_3]$, the converge is substantially slower for higher $|A_j|$, meaning that more evidence is needed until θ_j^* can be rejected. Figure 8.2 shows how a higher number of actions affects the average convergence rate of p-values computed with z_2, z_3.

In addition to the score functions z_k, a central aspect for the convergence of p-values are the corresponding weights w_k (cf. (8.7)). As mentioned in Section 8.3.1, we use uniform weights $w_k = \frac{1}{K}$. However, to show that the weighting is no trivial matter, we repeated our experiments with four alternative weighting schemes: Let $z_k^\tau = z_k(\tilde{\mathbf{a}}_j^\tau, \theta_j^*) - z_k(\hat{\mathbf{a}}_j^\tau, \theta_j^*)$ denote the summands in (8.7). The weighting schemes `truemax` / `truemin` assign $w_k = 1$ for the first k that maximises / minimises $|z_k^\tau|$, and 0

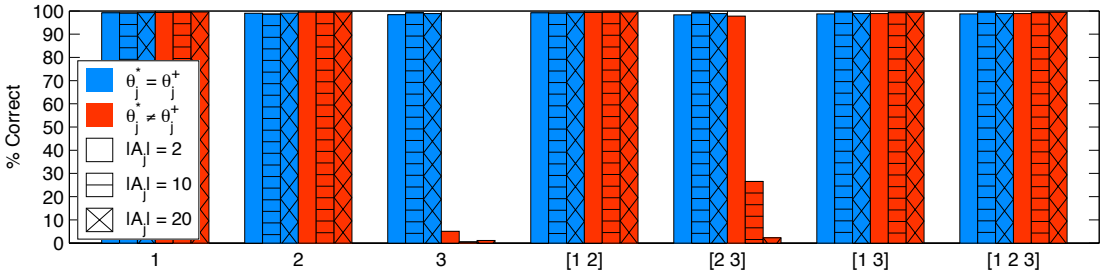


Figure 8.3: Average accuracy with random behaviours, for $N = 50$ and $|A_j| = 2, 10, 20$. Results averaged over 500 processes with 10000 time steps, for $\theta^*_j = \theta^+_j$ and $\theta^*_j \neq \theta^+_j$ each. The x-axis shows score functions z_k used in test statistic. Weights w_k computed using `truemax` weighting.

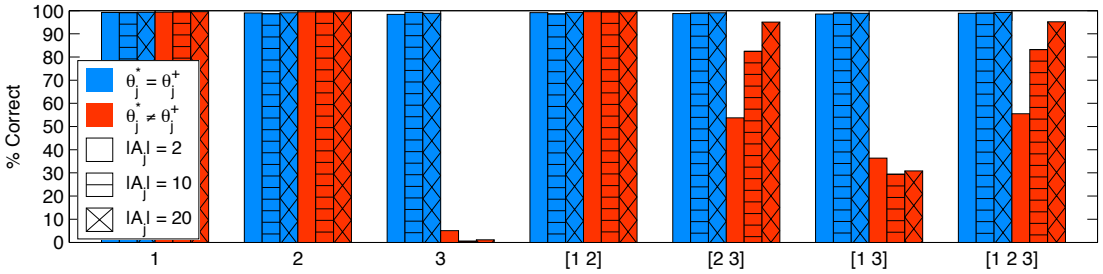


Figure 8.4: Average accuracy with random behaviours, for $N = 50$ and $|A_j| = 2, 10, 20$. Results averaged over 500 processes with 10000 time steps, for $\theta^*_j = \theta^+_j$ and $\theta^*_j \neq \theta^+_j$ each. The x-axis shows score functions z_k used in test statistic. Weights w_k computed using `truemin` weighting.

otherwise. Similarly, the weighting schemes `max` / `min` assign $w_k = 1$ for the first k that maximises / minimises z^τ_k, and 0 otherwise.

Figures 8.3 and 8.4 show the results for `truemax` and `truemin`. As can be seen in the figures, `truemax` is very similar to uniform weights while `truemin` improves the convergence for $[z_2, z_3]$ but compromises elsewhere. The results for `max` and `min` are very similar to those of `truemin` and `truemax`, respectively, hence we omit them.

Finally, we recomputed all accuracies using a more lenient significance level of $\alpha^* = 0.05$. As could be expected, this marginally decreased and increased (i.e. by a few percentage points) the accuracy for $\theta^*_j = \theta^+_j$ and $\theta^*_j \neq \theta^+_j$, respectively. This was primarily observed in the early stages of the interaction. Overall, however, the results were very similar to those obtained with $\alpha^* = 0.01$.

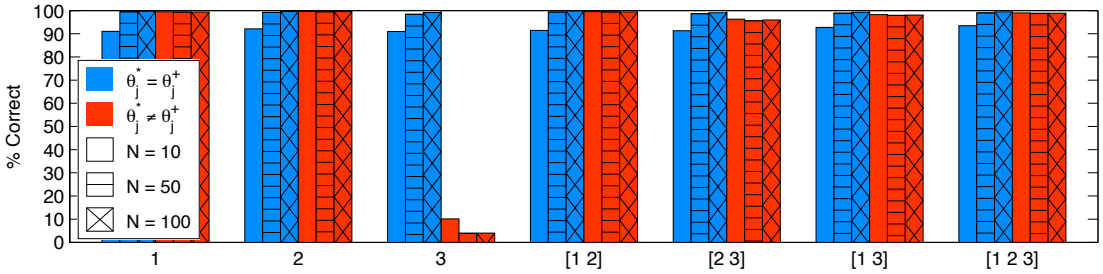


Figure 8.5: Average accuracy with random behaviours, for $|A_j| = 2$ and $N = 10, 50, 100$. Results averaged over 500 processes with 10000 time steps, for $\theta_j^* = \theta_j^+$ and $\theta_j^* \neq \theta_j^+$ each. The x-axis shows score functions z_k used in test statistic.

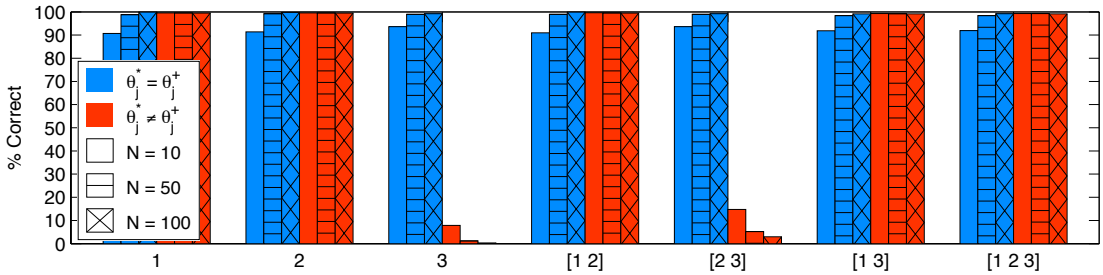


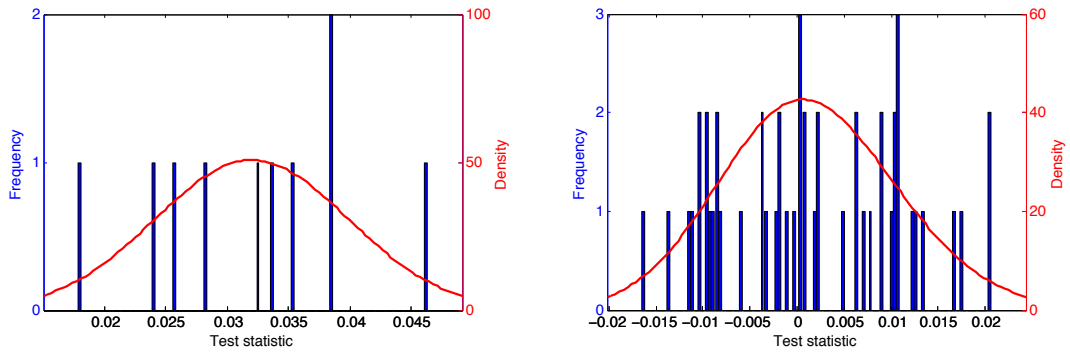
Figure 8.6: Average accuracy with random behaviours, for $|A_j| = 20$ and $N = 10, 50, 100$. Results averaged over 500 processes with 10000 time steps, for $\theta_j^* = \theta_j^+$ and $\theta_j^* \neq \theta_j^+$ each. The x-axis shows score functions z_k used in test statistic.

8.4.1.2 Sampling Complexity

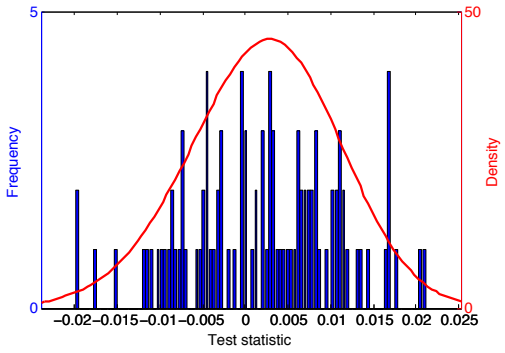
Recall that N specifies the number of sampled action vectors $\tilde{\mathbf{a}}_j^{t,n}$ used to learn the distribution of the test statistic (cf. Section 8.3.3). In the previous section, we reported results for $N = 50$. In this section, we investigate differences in accuracy for $N = 10, 50, 100$.

Figures 8.5 and 8.6 show the differences for $|A_j| = 2, 20$, respectively. (The figure for $|A_j| = 10$ was virtually the same as the one for $|A_j| = 20$, except with minor improvements in accuracy for the $[z_2, z_3]$ cluster. Hence, we omit it here.) As can be seen, there were improvements of up to 10% from $N = 10$ to $N = 50$, and no (or very marginal) improvements from $N = 50$ to $N = 100$. This was observed for all $|A_j| = 2, 10, 20$, and all constellations of score functions. The fact that $N = 50$ was sufficient even for $|A_j| = 20$ is remarkable, since, under random behaviours, there are 20^t possible action vectors to sample at any time t.

We also compared the learned skew-normal distributions and found that they fitted the data very well. Figures 8.7 and 8.8 show the histograms and fitted skew-normal



(a) $N = 10$ (b) $N = 50$

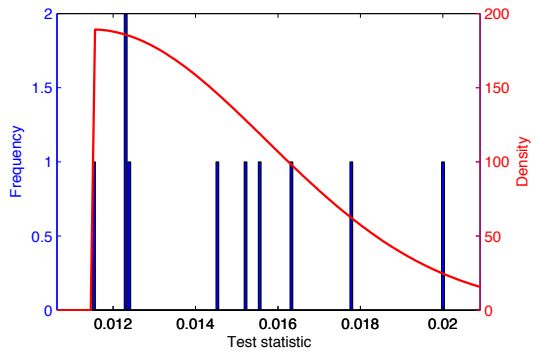


(c) $N = 100$

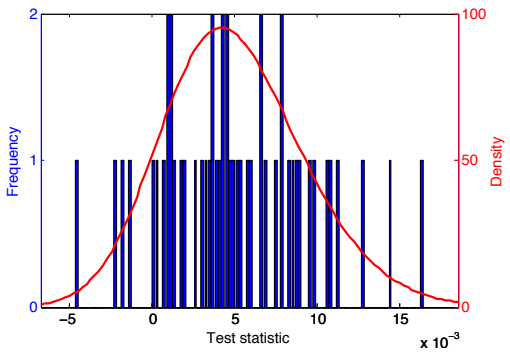
Figure 8.7: Example histograms and fitted skew-normal distributions (shown in red curve) after 1000 time steps, for random behaviours with $|A_j| = 10$ and $N = 10, 50, 100$. Using score function z_1 in test statistic.

distributions for two example processes after 1000 time steps. In Figure 8.8, we deliberately chose an example in which the learned distribution was maximally skewed for $N = 10$, which is a sign that N was too small. Nonetheless, in the majority of the processes, the learned distribution was only moderately skewed and our algorithm achieved an average accuracy of 90% even for $N = 10$. Moreover, if one wants to avoid maximally skewed distributions, one can simply restrict the parameter space when fitting the skew-normal (specifically, the shape parameter β; cf. Section 8.3.3).

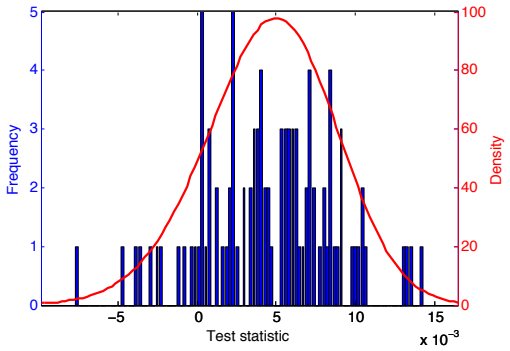
The flexibility of the skew-normal distribution was particularly useful in the early stages of the interaction, in which the test statistic typically does not follow a normal distribution. Figure 8.9 shows the test distribution for an example process after 10 time steps, using z_2 for the test statistic and $N = 100$ (the histogram was created using $N = 10000$). The learned skew-normal approximated the true test distribution very closely. Note that, in such examples, the normal and Student distributions do not produce good fits.



(a) $N = 10$



(b) $N = 50$



(c) $N = 100$

Figure 8.8: Example histograms and fitted skew-normal distributions (shown in red curve) after 1000 time steps, for random behaviours with $|A_j| = 10$ and $N = 10, 50, 100$. Using score functions z_1, z_2, z_3 in test statistic.

Our implementation of the algorithm performed all calculations as iterative updates (except for the skew-normal fitting). Hence, it used little (fixed) memory and had very low computation times. For example, using all three score functions and $|A_j| = 20, N = 100$, one cycle in the algorithm (cf. Algorithm 5) took on average less than 1 millisecond without fitting the skew-normal parameters, and less than 10 milliseconds when fitting the skew-normal parameters (using an off-the-shelf Simplex-optimiser with default parameters). The times were measured using Matlab R2014a on a Unix machine with a 2.6 GHz Intel Core i5 processor.

8.4.2 Adaptive Behaviours

We complemented the "structure-free" interaction of random behaviours by conducting analogous experiments with three additional classes of behaviours. Specifically, we used the benchmark framework specified in Chapter 6, which consists of 78 distinct 2×2

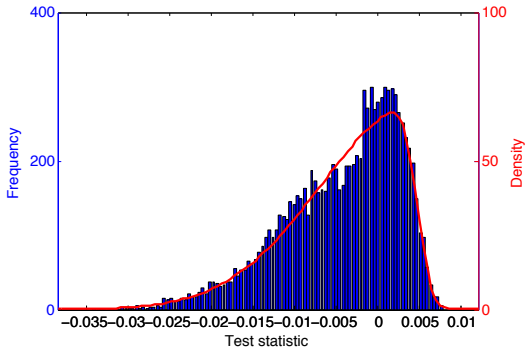


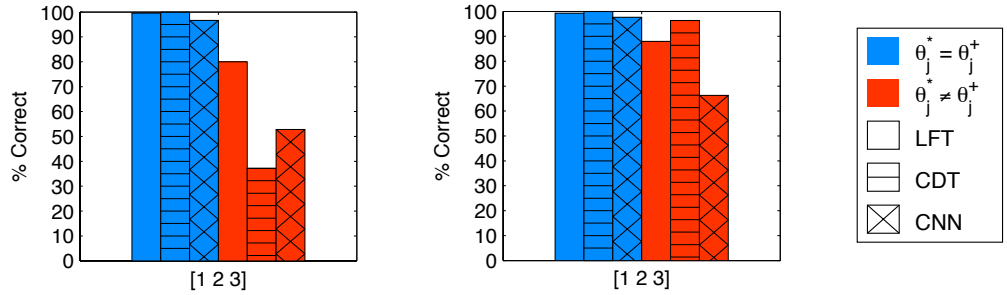
Figure 8.9: Example of true test distribution for z_2 and learned skew-normal distribution (shown in red curve) after 10 time steps, with $|A_j| = 10$ and $N = 100$.

matrix games and three methods to automatically generate sets of behaviours for any given game. The three behaviour classes are Leader-Follower-Trigger Agents (LFT), Co-Evolved Decision Trees (CDT), and Co-Evolved Neural Networks (CNN). These classes cover a broad spectrum of possible behaviours, including fully deterministic (CDT), fully stochastic (CNN), and hybrid (LFT) behaviours. Furthermore, all generated behaviours are *adaptive* to varying degrees (i.e. they adapt their action choices based on the other player's choices). Detailed descriptions of the games and behaviour classes can be found in Appendices A and B, respectively.

The following experiments were performed for each behaviour class, using identical randomisation: For each of the 78 games, we simulated 10 interaction processes, each lasting 10000 time steps. For each process, we randomly sampled behaviours $\theta_i \in \Theta_i, \theta_j^+ \in \Theta_j$ to control agents i and j, respectively, where Θ_i, Θ_j were restricted to the same behaviour class. In half of these processes, we used a correct hypothesis $\theta_j^* = \theta_j^+$, and in the other half, we sampled a random hypothesis $\theta_j^* \in \Theta_j$ with $\theta_j^* \neq \theta_j^+$. As before, we repeated each simulation for $N = 10, 50, 100$ and all constellations of score functions, but found that there were virtually no differences. Hence, in the following, we report results for $N = 50$ and the $[z_1, z_2, z_3]$ cluster.

Figure 8.10a shows the average accuracy achieved by our algorithm for all three behaviour classes. While the accuracy for $\theta_j^* = \theta_j^+$ was generally good, the accuracy for $\theta_j^* \neq \theta_j^+$ was mixed. Note that this was not merely due to the fact that the score functions were imperfect (cf. Section 8.3.1), since we obtained the same results for all combinations. Rather, this reveals an inherent limitation of our approach, which is that *we do not actively probe aspects of the hypothesis* θ_j^*. In other words, our algorithm performs statistical hypothesis tests based only on evidence that was generated by θ_i.

To illustrate this, it is useful to consider the tree structure of behaviours in the CDT



(a) Θ_i, Θ_j restricted to same class (b) Θ_i set to random behaviours

Figure 8.10: Average accuracy with behaviour classes LFT, CDT, CNN, for $N = 50$. Results averaged over 500 processes with 10000 time steps, for $\theta_j^* = \theta_j^+$ and $\theta_j^* \neq \theta_j^+$ each. Bars shown for $[z_1, z_2, z_3]$ test statistic.

class. Each node in a tree θ_j^+ corresponds to a past action taken by θ_i (cf. Figure B.1 in Appendix B). Depending on how θ_i chooses actions, we may only ever see a subset of the entire tree that defines θ_j^+. However, if our hypothesis θ_j^* differs from θ_j^+ only in the unseen aspects of θ_j^+, then there is no way for our algorithm to differentiate the two. Hence the asymmetry in accuracy for $\theta_j^* = \theta_j^+$ and $\theta_j^* \neq \theta_j^+$. Note that this problem did not occur in random behaviours because, there, all aspects are eventually visible.

Following this observation, we repeated the same experiments but restricted Θ_i to random behaviours (cf. Section 8.4.1), with the goal of exploring θ_j^* more thoroughly. As shown in Figure 8.10b, this led to significant improvements in accuracy, especially for the CDT class. Nonetheless, choosing actions purely randomly may not be a sufficient probing strategy, hence the accuracy for CNN was still relatively low. For CNN, this was further complicated by the fact that two neural networks θ_j, θ'_j may formally be different ($\theta_j \neq \theta'_j$) but have essentially the same action probabilities (with extremely small differences). Hence, in such cases, we would require much more evidence to distinguish the behaviours.

Chapter 9

Conclusion

This thesis is a comprehensive study of the type-based methodology in the context of ad hoc coordination, in which an autonomous agent seeks to achieve flexible and efficient interaction with other agents whose behaviours are a priori unknown. The idea in the type-based method is to compare the observed actions of the other agents with a set of hypothesised behaviours, called types, and to plan our own actions with respect to those types which we believe are most likely. We formulated a canonical description of this idea, called Harsanyi-Bellman Ad Hoc Coordination (HBA), and demonstrated its potential to address ad hoc coordination problems in empirical case studies.

Two central aspects of this method are the beliefs over types and the types themselves. In this regard, we identified and addressed a spectrum of important questions, pertaining to the evolution and impact of beliefs as well as the implications and detection of incorrect types. Specifically, how can evidence be incorporated into beliefs and under what conditions will the beliefs be correct? What practical impact do prior beliefs have and can they be computed automatically? Moreover, what relation must the hypothesised types have to the true types in order for HBA to solve its task? And is it possible to ascertain during the interaction whether the hypothesised types are incorrect?

Ad hoc coordination in multiagent systems constitutes an extremely complex and difficult problem, with relevance to many important applications. The type-based method has the potential to be a key component in solutions to ad hoc coordination problems, and this thesis provides both a formal introduction as well as useful insights regarding the application and theoretical underpinning of this method. In the remaining sections, we will highlight the key results of this thesis, discuss some of the limitations of HBA, and present possible directions for future research.

9.1 Selection of Key Results

This thesis provides a number of contributions to the theory and practice of the type-based method. A summary of the contributions can be found in Section 1.1 as well as at beginning of corresponding chapters. Some of the results stand out from the others, and the following is a selection of key results in this thesis:

- A remarkable result is the fact that we were able to show that prior beliefs over hypothesised types can be computed automatically with consistent effects on the long-term performance of HBA (cf. Observation 3 in Chapter 6). This is intriguing because it demonstrates that prior beliefs have the potential to manipulate the interaction trajectory to optimise certain metrics (such as our own payoffs), while in truth and without further evidence, there is no reason to believe that any one type is a priori more likely than others (see Section 6.2 for explanations). Practically, this result indicates that prior beliefs could be eliminated as a manual parameter in the type-based method and instead be computed automatically from the hypothesised types and problem description. However, the relevance of this result goes beyond the type-based method studied in this thesis; it also relates to the theory of learning in Bayesian games (cf. Section 2.4) and to a substantial body of research on prior beliefs (e.g. Bernardo, 1979; Jaynes, 1968).

- A potential concern associated with the type-based method is the fact that the hypothesised types may be incorrect. Hence, an important question is what relation the hypothesised types must have to the true types in order for HBA to be able to solve its task optimally? In this regard, we were able to draw a connection to the theory of formal model verification (cf. Section 2.7). Specifically, in Chapter 7, we proved the notable result that if there exists a probabilistic bisimulation between the interaction processes induced by the true and hypothesised types, then HBA will solve its task with the same probability and in the same average time *as if it knew the true types*, regardless of the specific nature and magnitude of inaccuracies in the hypothesised types. Practically, this means that we can use efficient methods from the model checking literature (e.g. Baier, 1996; Larsen and Skou, 1991) to verify optimality of types in practice.

- The bisimulation analysis for optimal types is performed *before* any interaction and with respect to the true types of other agents. How can we ascertain *during* the interaction, and with no prior knowledge of the true types, whether our hy-

pothesised types may be incorrect? Interestingly, it can be shown that established methods for statistical hypothesis testing can be employed to answer this question. Specifically, in Chapter 8, we formulated a highly-efficient automated analysis in the form of a frequentist hypothesis test (i.e. p-values) and showed that the test distribution can be learned during the interaction process. Since our method can be applied to the combination of beliefs and types as well as to individual types (cf. Section 8.2), it is of relevance for any intelligent agent that uses a hypothesis regarding the behaviour of other agents, including opponent modelling methods such as those discussed in Section 2.3.

9.2 Discussion

As discussed in Chapter 1 and Section 2.3, the type-based method (and, specifically, its canonical description HBA) has several features which make it a suitable approach to ad hoc coordination problems. In particular, the fact that we may hypothesise any types of behaviours means that we can potentially deal with a great variety of different agents. Furthermore, since types specify complete behaviours, we can plan actions far into the future, including in previously unseen situations. Nonetheless, there are some potential limitations that should be taken into account.

One potential limitation is the computational complexity of the planning step in HBA. In its canonical form (Algorithm 2), HBA unfolds a complete tree of all future trajectories which in practice have to be pruned after some time or depth (e.g. as in Algorithm 3). The time complexity of this process is exponential in factors such as the number of agents, actions, and states in the system. Therefore, the planning step can be a very costly operation in complex interactive domains. Nonetheless, as shown in Section 4.2, the planning step can be made more efficient by using stochastic sampling, albeit possibly at the cost of reduced planning accuracy.

Another potential limitation is the fact that the number of types one may wish to hypothesise can grow substantially with the size of the interaction problem (e.g. number of agents, actions, states, etc.). For example, it is relatively simple to hypothesise a small set of reasonable types with good coverage of possible behaviours for simple matrix games such as those used in Section 4.1. However, it is less trivial to hypothesise a set of reasonable types with good coverage in more complex domains such as the foraging domain used in Section 4.2. From a computational perspective, it is advisable to minimise the number of hypothesised types since each of their predictions have to

be computed at each point in time. Moreover, if types are specified manually, this can become a very cumbersome task in complex domains.

Since we define types as black-box functions, they may implement any kind of logic and, therefore, be arbitrarily complex. In most of our experiments, we used relatively simple types which can be executed very efficiently (e.g. see Tables 4.2 and 4.3 as well as Appendix B). However, in the simulated experiments discussed in Section 4.2, we also used complex types which use reinforcement learning and plan their actions with respect to the system dynamics and learned agent models (specifically, C/JAL implemented using Algorithm 4). This significantly increased the computational demands of HBA compared to the simpler types used in the other experiments.

Section 1.2 discussed the informational assumptions underlying HBA. One such assumption is that states and actions can be observed directly. This assumption may be problematic in robotic domains, in which states and actions are often not directly observed but instead inferred from possibly noisy and incomplete sensor data. If the degree of stochasticity (i.e. noisiness) is too large to handle with heuristic solutions (e.g. maximum probability heuristic), then it may be necessary to explicitly model such uncertainties, e.g. as in I-POMDPs (cf. Section 2.5).

Another assumption is that the system dynamics (i.e. the rules by which the system transitions into different states) are known beforehand and correct. In many applications, it may be difficult or impossible to specify a complete and correct model of the system. Rather, the model may be incomplete and inaccurate, or the model may have to be learned during the interaction using machine learning methods. This bears an interesting resemblance to the possibility of incorrect hypothesised types, which we treated extensively in this thesis. It would be interesting to explore to what extent our observations regarding incorrect types also hold for incorrect system models.

9.3 Directions for Future Work

This thesis provides a rich ground for future research, and below we discuss a selection of possible research directions. (Additional inspiration may be taken from the discussion on scope and limitations in Section 1.2, and the discussion in Section 9.2.)

Posterior and Prior Beliefs

In Chapter 5, we formulated three basic ways to incorporate evidence into posterior beliefs. Each of these formulations is based on assumptions regarding properties of the type distribution, and limited by the conditions under which they are guaranteed to converge to the true distribution of types. One direction for future research would be to investigate novel formulations of posterior beliefs which are more flexible regarding properties of the type distribution, and which converge to the true type distribution under less restrictive conditions. Furthermore, since our correctness analysis was asymptotic, it would be interesting to know if there exist useful finite-time error bounds.

In Chapter 6, we specified a number of methods to automatically compute prior beliefs and showed empirically that they can produce consistent performance effects. An interesting direction for future research would be to develop a theoretical explanation of these results. Moreover, it would be interesting to know if prior beliefs can be computed efficiently with useful performance *guarantees*. The LP-priors studied in this work are a first attempt in this direction, since solving a LP-prior also provides a bound on the expected loss of whichever intrinsic metric is being optimised. However, as discussed in Chapter 6, the LP formulation is a simplification of the true reasoning of HBA and so the bound may be incorrect.

Optimality of Hypothesised Types

In Chapter 7, we analysed under what relation between hypothesised and true types HBA is able to solve its task, despite inaccuracies in the hypothesised types. Our analysis was general in that it was invariant to the specific posterior formulation used in HBA. This analysis could be refined by committing to a specific posterior formulation (such as "product posteriors"; cf. Section 5.2). Furthermore, our analysis focused on flexibility, which is the average probability with which HBA is expected to solve its task. However, it does not directly relate to the notion of efficiency, which is the average payoff received in solved tasks (cf. Section 3.3). An interesting direction for future work would be to extend our analysis to account for both flexibility and efficiency.

Properties of Behavioural Hypothesis Testing

The statistical hypothesis test developed in Chapter 8 is based on the notion of score functions, which specify a relation between observed actions and hypothesised action

probabilities. To bring some structure into the space of score functions, we introduced the concepts of consistency and perfection as minimal and ideal properties, respectively. However, more research is required to understand precisely what properties a useful score function should satisfy, and whether the concept of perfection is feasible or even necessary in the general case.

The overall test statistic is computed as a linear combination of the differences between scores, where each coefficient in the sum can be interpreted as a weight that quantifies the importance of differences. While our asymptotic analysis of the test statistic was agnostic of the specific weighting, we experimented with alternative weighting schemes to show that the weighting can have a substantial effect on convergence rates. An interesting direction for future research would be to investigate the effect of score weights on the convergence of p-values and overall decision accuracy.

Automatic Hypothesis Generation

There are several methods to hypothesise possible behaviours. The default method is to specify types manually based on past experience and expectations regarding the behaviour of agents. We chose this approach in the case studies presented in Chapter 4. Another method is to generate types automatically from the problem description. For example, in Chapters 6 and 8, we used three different methods to automatically generate sets of behaviours for any given matrix game. Yet another method would be to extract possible behaviours from a corpus of historical data, for example by using machine learning methods. An important direction for future work is to develop new methods which can automatically hypothesise reasonable sets of behaviours for complex domains, using information from the problem description and past experience.

Structured Types

As discussed in Section 9.2, a potential problem with the type-based method is the fact that the number of types one may want to hypothesise can grow dramatically with the size of the interaction problem. This is problematic from a computational perspective because the prediction of each type has to be computed at each point in time. Furthermore, if types are specified manually, this can be a cumbersome task for the user. Therefore, it seems a rather important research direction to develop methods to effectively reduce the number of hypothesised types.

One way to achieve this would be to allow for parameters in the type specifications.

That is, rather than hypothesising several instances of essentially the same behaviour with different parameter settings, it would be useful to specify a single parameterised type to cover the entire spectrum of instances. This would require an ability to automatically infer the correct parameter setting from the interaction history. Another possible method would be to allow for a hierarchical description of types. That is, instead of specifying a single complete behaviour, we might specify a library of incomplete behaviours (for example, limited to subsets of the state space) and learn how to combine them to describe the true behaviour of an agent.

Complex Applications

Much research in artificial intelligence is focused on innovative applications such as adaptive user interfaces, robotic elderly care, and automated trading agents. Ad hoc coordination is a key technological challenge in such applications (cf. Chapter 1), and the type-based method has the potential to be an important component in their realisation. The experiments conducted in this thesis show how a method such as HBA can be applied successfully in practice, albeit in simplified domains and under laboratory conditions. Therefore, the next step is to deploy HBA in complex real-world applications such as the ones mentioned above. We expect that such a transition would bring many issues to the fore, including those discussed in Section 9.2 and this section.

Appendix A

Games

This appendix contains a listing of all structurally distinct strictly ordinal 2×2 matrix games, based on (Rapoport and Guyer, 1966). The games are distinct in that no game can be reproduced by any transformation of any other game. Possible transformations include interchanging the rows, columns, players, and any combinations thereof. The games are strictly ordinal, meaning that each player ranks the 4 possible outcomes from 1 (least preferred) to 4 (most preferred), and no two outcomes can have the same rank.

The games are presented in the following format:

N	(X)
$\underline{a_{1,1}, b_{1,1}}$	$a_{1,2}, b_{1,2}$
$a_{2,1}, b_{2,1}$	$a_{2,2}, b_{2,2}$

N is the number of the game and X is the corresponding number of the game in the original listing (Rapoport and Guyer, 1966). The variables $a_{i,j}$ and $b_{i,j}$, where $i, j \in \{1,2\}$, contain the payoffs to player 1 (row player) and player 2 (column player), respectively, if player 1 chooses action i and player 2 chooses action j. A payoff pair is underlined if the corresponding actions constitute a pure Nash equilibrium.

The listing is divided into one listing for all no-conflict games and one listing for all conflict games. In a no-conflict game, the players have the same set of most preferred outcomes. In a conflict game, the players disagree on the most preferred outcomes.

A.1 No-Conflict Games

1	*(1)*
4, 4	3, 3
2, 2	1, 1

2	*(2)*
4, 4	3, 3
1, 2	2, 1

3	*(3)*
4, 4	3, 2
2, 3	1, 1

4	*(4)*
4, 4	3, 2
1, 3	2, 1

5	*(5)*
4, 4	3, 1
1, 3	2, 2

6	*(6)*
4, 4	2, 3
3, 2	1, 1

7	*(22)*
4, 4	3, 3
2, 1	1, 2

8	*(23)*[1]
4, 4	3, 3
1, 1	2, 2

9	*(24)*
4, 4	3, 2
2, 1	1, 3

10	*(25)*
4, 4	3, 2
1, 1	2, 3

11	*(26)*
4, 4	2, 3
3, 1	1, 2

12	*(27)*
4, 4	2, 2
3, 1	1, 3

13	*(28)*
4, 4	3, 1
2, 2	1, 3

14	*(29)*
4, 4	3, 1
1, 2	2, 3

15	*(30)*
4, 4	2, 1
3, 2	1, 3

16	*(58)*
4, 4	2, 3
1, 1	**3, 2**

17	*(59)*
4, 4	2, 2
1, 1	**3, 3**

18	*(60)*
4, 4	2, 1
1, 2	**3, 3**

19	*(61)*
4, 4	1, 3
3, 1	**2, 2**

20	*(62)*
4, 4	1, 2
3, 1	**2, 3**

21	*(63)*
4, 4	1, 2
2, 1	**3, 3**

A.2 Conflict Games

1	*(7)*
3, 3	4, 2
2, 4	1, 1

2	*(8)*
3, 3	4, 2
1, 4	2, 1

3	*(9)*
3, 3	4, 1
1, 4	2, 2

4	*(10)*
2, 3	4, 2
1, 4	3, 1

5	*(11)*
2, 3	4, 1
1, 4	3, 2

6	*(12)*
2, 2	4, 1
1, 4	3, 3

7	*(13)*
3, 4	4, 2
2, 3	1, 1

8	*(14)*
3, 4	4, 2
1, 3	2, 1

9	*(15)*
3, 4	4, 1
2, 3	1, 2

10	*(16)*
3, 4	4, 1
1, 3	2, 2

11	*(17)*
2, 4	4, 2
1, 3	3, 1

12	*(18)*
2, 4	4, 1
1, 3	3, 2

13	*(19)*
3, 4	4, 3
1, 2	2, 1

14	*(20)*
3, 4	4, 3
2, 2	1, 1

15	*(21)*
2, 4	4, 3
1, 2	3, 1

[1] Game no. 23 in the original listing (Rapoport and Guyer, 1966) has a typo: payoff $a_{2,1}$ must be 1.

16 *(31)*	
3, 4	2, 2
1, 3	4, 1

17 *(32)*	
3, 4	2, 1
1, 3	4, 2

18 *(33)*	
3, 4	1, 2
2, 3	4, 1

19 *(34)*	
3, 4	1, 1
2, 3	4, 2

20 *(35)*	
2, 4	3, 2
1, 3	4, 1

21 *(36)*	
2, 4	3, 1
1, 3	4, 2

22 *(37)*	
3, 4	2, 3
1, 2	4, 1

23 *(38)*	
3, 4	1, 3
2, 2	4, 1

24 *(39)*	
2, 4	3, 3
1, 2	4, 1

25 *(40)*	
3, 4	4, 1
2, 2	1, 3

26 *(41)*	
3, 4	4, 1
1, 2	2, 3

27 *(42)*	
3, 3	4, 1
2, 2	1, 4

28 *(43)*	
3, 3	4, 1
1, 2	2, 4

29 *(44)*	
2, 4	4, 1
1, 2	3, 3

30 *(45)*	
3, 2	4, 1
2, 3	1, 4

31 *(46)*	
3, 2	4, 1
1, 3	2, 4

32 *(47)*	
2, 3	4, 1
1, 2	3, 4

33 *(48)*	
2, 2	4, 1
1, 3	3, 4

34 *(49)*	
3, 4	4, 3
2, 1	1, 2

35 *(50)*	
3, 4	4, 3
1, 1	2, 2

36 *(51)*	
3, 4	4, 2
2, 1	1, 3

37 *(52)*	
3, 4	4, 2
1, 1	2, 3

38 *(53)*	
3, 3	4, 2
2, 1	1, 4

39 *(54)*	
3, 3	4, 2
1, 1	2, 4

40 *(55)*	
2, 4	4, 3
1, 1	3, 2

41 *(56)*	
2, 4	4, 2
1, 1	3, 3

42 *(57)*	
2, 3	4, 2
1, 1	3, 4

43 *(64)*	
3, 4	2, 1
1, 2	4, 3

44 *(65)*	
2, 4	3, 1
1, 2	4, 3

45 *(66)*	
3, 3	2, 4
4, 2	1, 1

46 *(67)*	
2, 3	3, 4
4, 2	1, 1

47 *(68)*	
2, 2	3, 4
4, 3	1, 1

48 *(69)*	
2, 2	4, 3
3, 4	1, 1

49 *(70)*	
3, 4	2, 1
4, 2	1, 3

50 *(71)*	
3, 3	2, 1
4, 2	1, 4

51 *(72)*	
3, 2	2, 1
4, 3	1, 4

52 *(73)*	
2, 4	4, 1
3, 2	1, 3

53 *(74)*	
2, 4	3, 1
4, 2	1, 3

54 *(75)*	
2, 3	4, 1
3, 2	1, 4

55 *(76)*	
2, 3	3, 1
4, 2	1, 4

56 *(77)*	
2, 2	4, 1
3, 3	1, 4

57 *(78)*	
2, 2	3, 1
4, 3	1, 4

Appendix B

Agents

This appendix contains algorithmic descriptions and parameter settings for the type generation methods used in Chapters 6 and 8. Informal descriptions of these methods can be found in Section 6.1.4.

We assume 2×2 matrix games in which $U_k(a_i, a_j)$, $k = i, j$, denotes the payoff to player k if player i takes action a_i and player j takes action a_j. Furthermore, we use π_k to denote a stage-game strategy for player k, which is defined as a probability distribution $\pi_k \in \Delta(A_k)$ over player k's actions. We write $U_k(\pi_i, \pi_j)$ to denote the expected payoff to player k under strategies (π_i, π_j). Finally, we use i to denote the player controlled by the agents described in this appendix, and j or $-i$ to denote the other player.

B.1 Leader-Follower-Trigger Agents

Given a matrix game with payoffs $U_{i,j}$, we use Algorithm 6 to generate the set $\hat{\Omega}$ of all "target solutions" of maximum length l. Each target solution $\omega \in \hat{\Omega}$ consists of 1 to l consecutive actions pairs $(\hat{a}_i^t, \hat{a}_j^t)$ to be played by the players. The set contains only those solutions in which each player has an average payoff at least as has high as the maximin (safety) value for that player. (Target solutions of length 2 or greater in which all action pairs are identical are removed.) We note that Crandall (2014) also specifies that target solutions must be "enforceable", such that following the solution constitutes a best-response for each player (see Section 3.1 in Crandall, 2014). However, as this may significantly reduce the number of target solutions, we omitted this additional requirement in our work. Finally, for each target solution ω in the set $\hat{\Omega}$, we use Algorithms 7, 8, and 9 to create a leader, follower, and trigger agent, respectively.

Parameter setting: maximum solution length $l = 2$

Algorithm 6 Procedure to generate target solutions for given game

Input: matrix game $U_{i,j}$

Output: target solutions $\hat{\Omega}$

Parameters: maximum solution length l

// Generate all solution sequences

$$\Omega \leftarrow \left\{\omega = \langle(\hat{a}_i^1, \hat{a}_j^1), ..., (\hat{a}_i^t, \hat{a}_j^t)\rangle \mid 1 \leq t \leq l,\ \hat{a}_i \in A_i,\ \hat{a}_j \in A_j\right\}$$

// Remove solutions in which either player does not obtain its maximin value

$$\hat{\Omega} \leftarrow \left\{\bar{\omega}(i) \geq U_i^* \wedge \bar{\omega}(j) \geq U_j^* \mid \omega \in \Omega\right\}$$

where

$$\bar{\omega}(k) = \frac{1}{|\omega|} \sum_{\tau=1}^{|\omega|} U_k(\hat{a}_i^\tau, \hat{a}_j^\tau)$$

$$U_k^* = \max_{\pi_k \in \Delta(A_k)} \min_{\pi_{-k} \in \Delta(A_{-k})} U_k(\pi_k, \pi_{-k})$$

Algorithm 7 Leader agent

Parameters: target solution ω

Initialise: "guilt" $\gamma \leftarrow 0$

loop

Observe current time t

$(\hat{a}_i^t, \hat{a}_j^t) \leftarrow$ next action pair in ω

if $\gamma = 0$ **then**

Play action $\hat{a}_i^t$

else

Play minimax strategy $\pi_i \in \arg\min_{\pi_i \in \Delta(A_i)} \max_{\pi_j \in \Delta(A_j)} U_j(\pi_i, \pi_j)$

end if

Observe action a_j^t taken by player j

if $a_j^t \neq \hat{a}_j^t$ or $\gamma > 0$ **then**

$\gamma \leftarrow \max\left[0, \gamma + U_j(\hat{a}_i^t, a_j^t) - \bar{\omega}(j) + [\gamma = 0]_1 * 0.1\right]$

where $\bar{\omega}(k)$ defined in Algorithm 6

end if

end loop

Algorithm 8 Follower agent

Parameters: target solution ω
Initialise: $d \leftarrow$ false
loop
  Observe current time t
  if d **then**
    $(\hat{a}_i^t, \hat{a}_j^t) \leftarrow$ random action pair in ω
  else
    $(\hat{a}_i^t, \hat{a}_j^t) \leftarrow$ next action pair in ω
  end if
  Play action $\hat{a}_i^t$
  Observe action a_j^t taken by player j
  if $a_j^t \neq \hat{a}_j^t$ **then**
    $d \leftarrow$ true
  else
    $d \leftarrow$ false
  end if
end loop

Algorithm 9 Trigger agent

Parameters: target solution ω
Initialise: $d \leftarrow$ false
loop
  Observe current time t
  if d **then**
    Play maximin strategy $\pi_i \in \arg\max_{\pi_i \in \Delta(A_i)} \min_{\pi_j \in \Delta(A_j)} U_i(\pi_i, \pi_j)$
  else
    $(\hat{a}_i^t, \hat{a}_j^t) \leftarrow$ next action pair in ω
    Play action $\hat{a}_i^t$
  end if
  Observe action a_j^t taken by player j
  if $a_j^t \neq \hat{a}_j^t$ **then**
    $d \leftarrow$ true
  end if
end loop

B.2 Co-Evolved Decision Trees / Neural Networks

Given a matrix game with payoffs $U_{i,j}$, we used the generic co-evolution algorithm given in Algorithm 10 to generate sets of decision tree agents and neural network agents for both players. The precise structure of these agents is explained in Figures B.1 and B.2, respectively. The parameter listing, below, specifies further implementation details for the co-evolution algorithm.

Parameter setting:

- Agents per population: 50
- Each agent evaluated against random 40% of other population
- Fitness = average payoff after 20 rounds ($\in [1,4]$) – average similarity ($\in [0,1]$)
- Resampling method: linear ranking
- Decision trees:
  - Tree depth: 3 (up to three previous actions of other player)
  - Evolutions: 300 (population with highest average fitness returned)
  - Similarity of two trees: percentage of nodes with same action choice
  - Random mutation of single node (flipping action): 5% of population
  - Random crossing of sub-trees (preserving tree depth): 30% of population
- Neural networks:
  - Input layer: 4 nodes (up to two previous joint actions)
  - Hidden layer: 5 nodes
  - Output layer: 1 node (probability of action 1)
  - Each node fully-connected with nodes of next layer
  - Standard sigmoidal threshold function
  - Evolutions: 1000 (population with highest average fitness returned)
  - Similarity of two networks: 1 – average difference of output for each input
  - Random mutation of single input weight (standard normal shift): 20% of population
  - Random crossing of nodes (preserving network structure): 10% of population

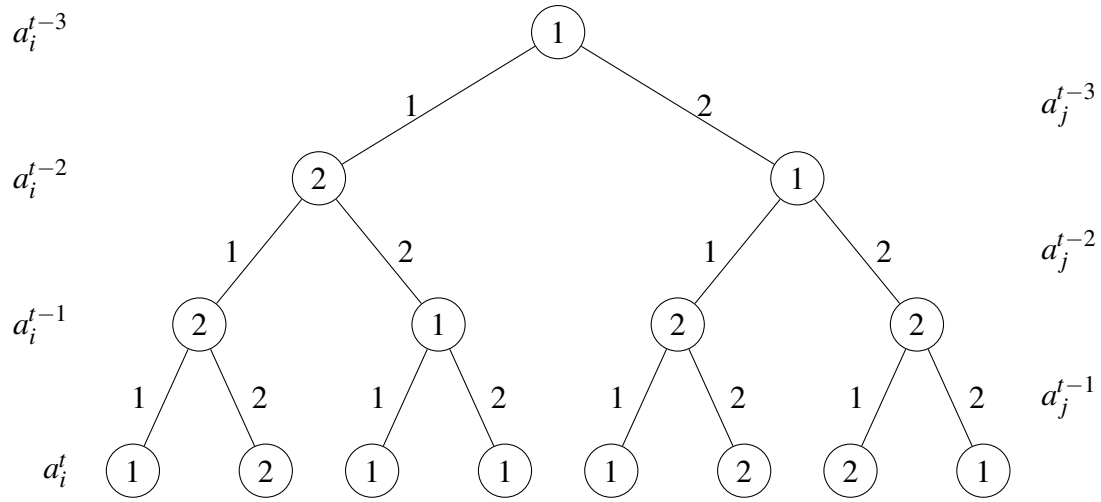


Figure B.1: Instance of a decision tree agent (controlling player i). Each node contains the action taken by player i in response to the previous actions of player j, represented by the edges. All trees in this work have depth 3, which means that they account for up to 3 previous actions of player j. For example, if the previous actions of player j were $\langle 2,1,1\rangle$, then player i chooses action 1; if only two rounds have been played and player j's actions were $\langle 1,1\rangle$, then player i chooses action 2.

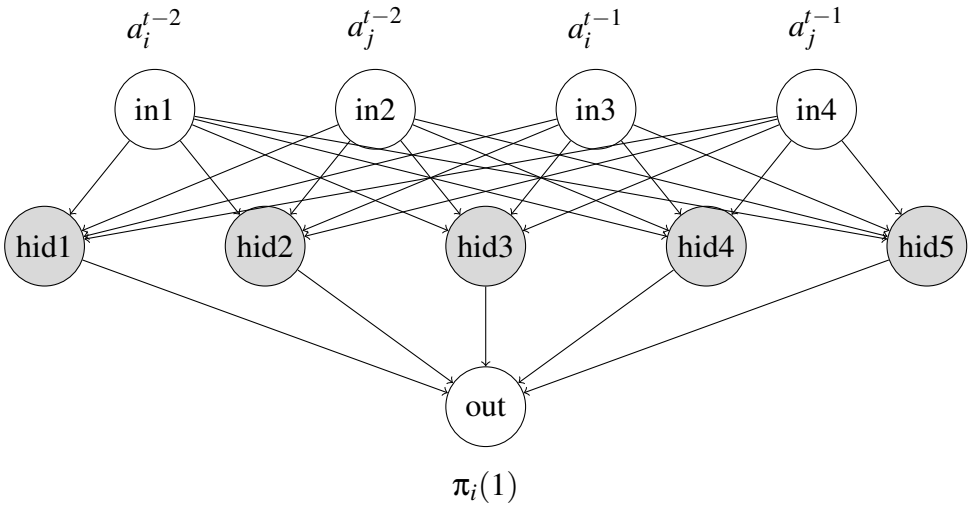


Figure B.2: Structure of neural network agent. The agent controls player i, the other player is denoted by j. Each network consists of 4 input nodes, 5 hidden nodes, and 1 output node. The input nodes take the previous two joint actions as inputs, or -1 if not available. Each node is fully-connected to the nodes in the next layer. Nodes have individual weights w_k for each incoming edge x_k, plus one "threshold" weight w^*. Using the standard sigmoid function, a node's output value is computed as $\frac{1}{1+e^{-(w^T x - w^*)}}$. The output node "out" returns the probability of taking action 1.

Algorithm 10 Generic co-evolution algorithm

Input: matrix game $U_{i,j}$

Output: sets Φ_i^* and Φ_j^* of co-evolved agents

Parameters: see text

Initialise random populations Φ_i and Φ_j

Set $\Phi_i^* \leftarrow \Phi_i$ and $\Phi_j^* \leftarrow \Phi_j$

Evaluate initial populations:

Each agent $\phi_i \in \Phi_i$ plays $U_{i,j}$ against random sample of Φ_j, and vice versa

repeat

Rank agents based on their fitness and resample proportionately

Modify agents:

Random mutation of random sample in Φ_i and Φ_j, respectively

Random crossing of random sample in Φ_i and Φ_j, respectively

Evaluate current populations:

Each agent $\phi_i \in \Phi_i$ plays $U_{i,j}$ against random sample of Φ_j, and vice versa

if average fitness of (Φ_i, Φ_j) better than (Φ_i^*, Φ_j^*) **then**

Set $\Phi_i^* \leftarrow \Phi_i$ and $\Phi_j^* \leftarrow \Phi_j$

end if

until maximum number of evolutions reached

Bibliography

N. Agmon and P. Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *Proceedings of 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 341–348, 2012.

N. Agmon, S. Barrett, and P. Stone. Modeling uncertainty in leading ad hoc teams. In *Proceedings of 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 397–404, 2014.

D.W. Albrecht, I. Zukerman, and A.E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8: 5–47, 1998.

S.V. Albrecht and S. Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 349–356, 2012.

S.V. Albrecht and S. Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems (extended abstract). In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 1155–1156, 2013a.

S.V. Albrecht and S. Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. Technical report, School of Informatics, The University of Edinburgh, 2013b.

S.V. Albrecht and S. Ramamoorthy. On convergence and optimality of best-response learning with policy types in multiagent systems. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pages 12–21, 2014.

S.V. Albrecht and S. Ramamoorthy. Are you doing what I think you are doing? Criticis-

ing uncertain agent models. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, 2015.

S.V. Albrecht, J.W. Crandall, and S. Ramamoorthy. An empirical study on the practical impact of prior beliefs over policy types. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 1988–1994, 2015a.

S.V. Albrecht, J.W. Crandall, and S. Ramamoorthy. An empirical study on the practical impact of prior beliefs over policy types – Appendix, 2015b. *http://rad.inf.ed.ac.uk/data/publications/2015/aaai15app.pdf*.

D. Avrahami-Zilberbrand and G.A. Kaminka. Incorporating observer biases in keyhole plan recognition (efficiently!). In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 944–949, 2007.

A. Azzalini. A class of distributions which includes the normal ones. *Scandinavian Journal of Statistics*, 12:171–178, 1985.

C. Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In *Proceedings of the 8th International Conference on Computer Aided Verification, Lecture Notes in Computer Science*, volume 1102, pages 38–49. Springer, 1996.

D. Banerjee and S. Sen. Reaching Pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multiagent Systems*, 15(1): 91–108, 2007.

S. Barrett. *Making Friends on the Fly: Advances in Ad Hoc Teamwork*. PhD thesis, The University of Texas at Austin, December 2014.

S. Barrett and P. Stone. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2010–2016, 2015.

S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 567–574, 2011.

S. Barrett, P. Stone, S. Kraus, and A. Rosenfeld. Teamwork with limited knowledge of teammates. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 102–108, 2013.

I.V. Basawa and D.J. Scott. Efficient tests for stochastic processes. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 21–31, 1977.

M.J. Bayarri and J.O. Berger. P values for composite null models. *Journal of the American Statistical Association*, 95(452):1127–1142, 2000.

R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

J.O. Berger and T. Sellke. Testing a point null hypothesis: the irreconcilability of p values and evidence (with discussion). *Journal of the American Statistical Association*, 82:112–122, 1987.

J.M. Bernardo. Reference posterior distributions for Bayesian inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):113–147, 1979.

E. Bonchek-Dokow, G.A. Kaminka, and C. Domshlak. Distinguishing between intentional and unintentional sequences of actions. In *Proceedings of the 9th International Conference on Cognitive Modeling*, pages 170–175, 2009.

M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 53–58, 2005.

M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

G.E.P. Box. Sampling and Bayes' inference in scientific modelling and robustness. *Journal of the Royal Statistical Society. Series A (General)*, pages 383–430, 1980.

R.I. Brafman and M. Tennenholtz. R-max – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3: 213–231, 2003.

S.J. Brams. *Theory of Moves*. Cambridge University Press, 1993.

G.W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376, 1951.

S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.

D. Carmel and S. Markovitch. Learning models of opponent's strategy in game playing. In *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning*, pages 140–147, 1993.

D. Chakraborty and P. Stone. Cooperating with a Markovian ad hoc teammate. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 1085–1092, 2013.

G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems*, pages 709–716, 2003.

E. Charniak and R.P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.

E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 1999.

C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 746–752, 1998.

V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

D.R. Cox. The role of significance tests (with discussion). *Scandinavian Journal of Statistics*, 4:49–70, 1977.

J.W. Crandall. Towards minimizing disappointment in repeated games. *Journal of Artificial Intelligence Research*, 49:111–142, 2014.

E. Dekel, D. Fudenberg, and D.K. Levine. Learning to play Bayesian games. *Games and Economic Behavior*, 46(2):282–303, 2004.

M.B. Dias, T.K. Harris, B. Browning, E.G. Jones, B. Argall, M. Veloso, A. Stentz, and A. Rudnicky. Dynamically formed human-robot teams performing coordinated tasks. In *AAAI Spring Symposium "To Boldly Go Where No Human-Robot Team Has Gone Before"*, pages 30–38, 2006.

P. Doshi and P.J. Gmytrasiewicz. On the difficulty of achieving equilibrium in interactive POMDPs. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1131–1136, 2006.

P. Doshi and P.J. Gmytrasiewicz. Monte carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, pages 297–337, 2009.

P. Doshi and D. Perez. Generalized point based value iteration for interactive POMDPs. In *Proceedings of the 23rd AAAI Conference on Artificial intelligence*, pages 63–68, 2008.

P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive POMDPs: representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18(3):376–416, 2009.

P. Doshi, X. Qu, A. Goodie, and D. Young. Modeling recursive reasoning by humans using empirically informed interactive POMDPs. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1223–1230, 2010.

H. Fischer. *A History of the Central Limit Theorem: From Classical to Modern Probability Theory*. Springer Science & Business Media, 2010.

R.A. Fisher. *The Design of Experiments*. Oliver & Boyd, 1935.

D.P. Foster and H.P. Young. Learning, hypothesis testing, and Nash equilibrium. *Games and Economic Behavior*, 45(1):73–96, 2003.

D. Fudenberg and J. Tirole. Perfect Bayesian equilibrium and sequential equilibrium. *Journal of Economic Theory*, 53(2):236–260, 1991a.

D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, 1991b.

Y. Gal, A. Pfeffer, F. Marzo, and B.J. Grosz. Learning social preferences in games. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 226–231, 2004.

A. Gelman and C.R. Shalizi. Philosophy and the practice of Bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 66(1):8–38, 2013.

K. Genter and P. Stone. Influencing a flock via ad hoc teamwork. In *Proceedings of the 9th International Conference on Swarm Intelligence*, pages 110–121, 2014.

K. Genter, N. Agmon, and P. Stone. Role-based ad hoc teamwork. In *Plan, Activity, and Intent Recognition: Theory and Practice*, pages 251–272. Elsevier, 2013.

I. Gilboa and D. Schmeidler. *A Theory of Case-Based Decisions*. Cambridge University Press, 2001.

P.J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24(1):49–79, 2005.

P.J. Gmytrasiewicz and E.H. Durfee. Rational coordination in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 3(4):319–350, 2000.

B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

B.J. Grosz and S. Kraus. The evolution of SharedPlans. In *Foundations of Rational Agency*, pages 227–262. Springer, 1999.

H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

J.C. Harsanyi. Games with incomplete information played by "Bayesian" players. Part I. The basic model. *Management Science*, 14(3):159–182, 1967.

J.C. Harsanyi. Games with incomplete information played by "Bayesian" players. Part II. Bayesian equilibrium points. *Management Science*, 14(5):320–334, 1968a.

J.C. Harsanyi. Games with incomplete information played by "Bayesian" players. Part III. The basic probability distribution of the game. *Management Science*, 14(7): 486–502, 1968b.

S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. *Economic Essays: A Festschrift for Werner Hildenbrand*, pages 181–200, 2001.

K. Hindriks and D. Tykhonov. Opponent modelling in automated multi-issue negotiation using Bayesian learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 331–338, 2008.

J.H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. The MIT Press, 1975.

J. Hu and M.P. Wellman. Nash q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003. ISSN 1532-4435.

E.T. Jaynes. Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics*, 4(3):227–241, 1968.

J.S. Jordan. Bayesian learning in normal form games. *Games and Economic Behavior*, 3(1):60–81, 1991.

L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101:99–134, 1998.

D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47:263–292, 1979.

E. Kalai and E. Lehrer. Rational learning leads to Nash equilibrium. *Econometrica*, 61 (5):1019–1045, 1993.

G.A. Kaminka and I. Frenkel. Integration of coordination mechanisms in the BITE multi-robot architecture. In *Proceedings of the International Conference on Robotics and Automation*, pages 2859–2866, 2007.

M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.

J.R. Koza. *Genetic programming: On the programming of computers by means of natural selection*. The MIT Press, 1992.

H.W. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, 2(28):193–216, 1953.

K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

X.-L. Meng. Posterior predictive p-values. *The Annals of Statistics*, pages 1142–1160, 1994.

John H. Nachbar. Beliefs in repeated games. *Econometrica*, 73(2):459–480, 2005.

J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

B. Ng, C. Meyers, K. Boakye, and J. Nitao. Towards applying interactive POMDPs to real-world adversary modeling. In *Proceedings of the 22nd Innovative Applications of Artificial Intelligence Conference*, pages 1814–1820, 2010.

A. O'Hagan and T. Leonard. Bayes estimation subject to uncertainty about parameter constraints. *Biometrika*, 63(1):201–203, 1976.

A. Rapoport and M. Guyer. A taxonomy of 2×2 games. *General Systems: Yearbook of the Society for General Systems Research*, 11:203–214, 1966.

B. Rathnasabapathy, P. Doshi, and P. Gmytrasiewicz. Exact solutions of interactive POMDPs using behavioral equivalence. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1025–1032, 2006.

D.B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

D. Ryabko and B. Ryabko. On hypotheses testing for ergodic processes. In *Proceedings of IEEE Information Theory Workshop*, pages 281–283, 2008.

S. Sen, S. Airiau, and R. Mukherjee. Towards a Pareto-optimal solution in general-sum games. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 153–160, 2003.

L.S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095, 1953.

F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 550–558, 2005.

T. Steffens. Feature-based declarative opponent-modelling. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 125–136. Springer, 2004.

P. Stone and S. Kraus. To teach or not to teach? decision making under uncertainty in ad hoc teams. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 117–124, 2010.

P. Stone, G.A. Kaminka, S. Kraus, and J.S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1504–1509, 2010.

G. Sukthankar, R.P. Goldman, C. Geib, D.V. Pynadath, and H.H. Bui. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann, 2014.

R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. The MIT Press, 1998.

M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7: 83–124, 1997.

H.J. van den Herik, H.H.L.M. Donkers, and P.H.M. Spronck. Opponent modelling and commercial games. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, pages 15–25, 2005.

A. Vehtari and J. Ojanen. A survey of Bayesian predictive methods for model assessment, selection and comparison. *Statistics Surveys*, 6:142–228, 2012.

W.A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. *Psychological Bulletin*, 77(1):65–72, 1972.

C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

J. Wendler and J. Bach. Recognizing and predicting agent behavior with case based reasoning. *RoboCup 2003: Robot Soccer World Cup VII*, pages 729–738, 2004.

M. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2nd edition, 2009.

H.P. Young. *Strategic Learning and Its Limits*. Oxford University Press, 2004.

Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 507–514, 2010.