REPRESENTING AND REASONING ABOUT BAYESIAN GAMES WITH EPISTEMIC LOGIC

Oldooz Dianat

Master of Computer Science, University Technology of Malaysia (UTM), 2010

Bachelor of Science in Applied Mathematics , University of Tehran, 2004

This dissertation is presented for the degree of

Doctor of Philosophy

at

Department of Computing

To the memory of my father.

Declaration

I certify that the work in this thesis entitled REPRESENTING AND REASONING ABOUT BAYESIAN GAMES WITH EPISTEMIC LOGIC has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree to any other university or institution other than Macquarie University. I also certify that the thesis is an original piece of research and it has been written by me. Any help and assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signed: Oldooz Dianat

Date:

Abstract

Multi-agent systems are systems which include more than one autonomous agent with either varying information or varying interests, or both. An agent in a multi-agent system should behave rationally, which informally is defined as choosing actions that improve its chance of success. This definition of rationality is adopted to game theory which is the science of studying interactions between agents in multi-agent systems. However, these descriptions do not consider the reasoning abilities of agents. One way to tackle this issue is to use logical declarative languages, as these languages enable reasoning about the best strategy in games by considering other players' rationality and reasoning abilities. Furthermore, logical languages are used to represent game models explicitly and these languages can formulate certain specific situations, such as game theory solutions. Agents are then able to verify the correctness of these formulae in the model, thus these languages equip agents with decision making capability based on reasoning.

In this thesis, we study normal form games, in which a set of agents make their decisions simultaneously, without the knowledge about the decisions of other agents, and Bayesian games that let agents face uncertainty and hold private information. We first provide an epistemic language which can model the knowledge of an agent for reasoning about games without uncertainty for reasoning about normal form games. We then extend it for representing and reasoning about Bayesian games. The extended language is used to describe explicit models that assist agents in decision making. In addition, this language is used as an expressive, general, semantically well-defined query language for model checkers. To show that our language is a succinct and expressive language and our approach is practical for a reasonable class of applications, several representative game scenarios are investigated, such as detection of attackers in wireless networks and recognition of the benefits of using cloud computing.

Acknowledgment

I would like to take this opportunity to thank my principal supervisor Professor Mehmet A. Orgun for his encouragement, and his belief in me. He gave me the freedom to pursue my own directions, while providing me with advice whenever I am in need. His emphasis on presenting the simplest explanation possible has helped me to understand problems in ways deeper that I thought possible. I am most indebted to his support and guidance.

To my associate supervisor Dr Lee Flax, thanks for his guidance. Thanks are also due to the members of the Department of Computing at Macquarie University for their help and encouragement. I am especially thankful of Associate Professor Lenore D. Zuck from the University of Illinois at Chicago for her valuable comments.

I am grateful of Yasaman Motazedi, a fellow PhD student in computing department for working together. I would like to thank Dr Christoph Krisp for his time and his comments on an earlier draft of this thesis.

I would like to thank my colleagues at the Commonwealth Scientific and Industrial Research Organisation (CSIRO) for a most enjoyable and fruitful year. I would also like to thank my master degree supervisor Professor Habibollah Haron who has first instilled in me a love for research.

From a more personal side, I am most grateful to my mother, who has over the years provided with unconditional love and care. Her willingness to let me pursue my dreams despite being far away, has given me the opportunity to complete this academic journey. I would also like to thank Sepideh and Pooya for their unwavering support.

I am also grateful for having a great circle of friends who share my joys and disappointments during PhD program, Atefeh, Emma, Francesca, Joshua, Kayla, Mauro, Melanie, Nieke, Nora and Tommaso. I am also grateful for family members and many friends, both near and far away.

List of Related Publications

This thesis has resulted in the following publications; my contribution to those publications is 80%.

Conference papers:

- O. Dianat and M. A. Orgun; Representing and Reasoning about Utilization of Cloud Computing as Bayesian games with Epistemic Logic. In Proceedings of The 4th International Conference on Ambient Systems, Networks and Technologies (ANT-2013), June 2528, 2013, Halifax, Nova Scotia, Canada, Procedia Computer Science, Volume 19, 2013, Pages 4047, ISSN 1877-0509.

- O. Dianat and M. A. Orgun; Modelling Bayesian Attacker Detection Game in Wireless Networks with Epistemic Logic. In Proceedings of The 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, October 1417, 2012 Pittsburgh, Pennsylvania, USA, IEEE Conference Publications, pages 210215.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Synergy means behaviour of whole systems unpredicted by the behaviour of their parts. (R. Buckminster Fulller, What I have learned).

1.1 Motivation

Agents can be considered as successors to the traditional knowledge-based or expert systems which could reason with the symbolic representation of some universe of discourse (or knowledge domain), and could provide expert-level advise on problems related to that domain. When several agents interact with each other in a computerised system, a multi-agent system is formed. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent to solve. Furthermore, these systems can reduce operation time by providing methods for distributed computation.

In multi-agent systems, an optimal action to be chosen by one agent depends on the actions the other agents choose, therefore game theory provides a suitable basis for the analysis of these interactions [31]. This is motivated by the fact that a game is a setting in which a player is an agent who operates in an environment determined by the other agents' behaviour and who takes the behaviour of the other agents as an input to its decision making process [96]. Under this interpretation, a solution's strategy entails the behaviour of a player who knows the other players' strategies but reasons about the outcome from the model's primitive, i.e., the preferences and the informational structures. In other words, it deals with the question of rational decision making by individual agents at situations of strategic interaction. To answer this question, traditional game theory offers a number of solution concepts. In abstract terms, a solution concept is a foundation that relates to each

game a set of strategies. Each strategy of this set is viewed as one that a rational player would choose. The Nash equilibrium probably is the best known solution concept for normal form games, that is, those as games in which agents choose an action simultaneously [92].

Most works in game theory have made two crucial assumptions. First, the payoff to each agent (player) is given by a fixed, deterministic value. Second, these values are common knowledge among all agents. However, both assumptions often fail to hold for real world problems. Here, players do not necessarily know which game they are playing, or who the other players are [112]. This leads to uncertainty in game theory, which is represented by a probability distribution over a set of possible games [67]. Classical Bayesianism is still a favourite means of dealing with uncertainty [66]. Therefore, games with incomplete information or Bayesian games provide a natural and compelling model that enables understanding the actions of agents in a multi-agent system under uncertainty. Bayesian games can simply be defined as incorporation of procedural rationality consideration into models of uncertain interactive situations, such as a set of normal form games where players do not know which game is about to be played.

Game theory is a well-studied area, it is, however, not based on logic. It is important to try to base it on a logic or to incorporate it in a logic, because game theory is often presented as a theory that models reasoning. In the theoretical description of a game, an informal notation of rationality is given in the situation that rational agents interact with each other [104]. Therefore, a logical language is required which is intended to make this informal rationality precise. A formal language for games is not only about formalising solution concepts but also about precisely and intuitively specifying the condition under which rational players should be expected to act to achieve a solution concept. Nevertheless, any action which game theory suggests, should be proved that, under appropriate setting, it is a rational move [104]. Therefore, it is an adequate practice to develop a formal language with a clear distinction between syntax and semantics and various notions of validity. Modal logic offers formalisation for games with the required syntax and semantics.

To analyse implicitly the concept of modal operators of modal logic, one should use the axiom systems of modal logic. To analyse explicitly a formal language, a model and rules have to be defined and the rules have to assign

values to the sentences of the language related to the model. Saul Kripke proposed semantics for modal logics, which allowed for a more systematic evaluation of modal logics. If a sentence is true in some set of models, then the sentence is valid relative to those models. This is the reason of flexibility of modal logic. As a result, based on demand, a logic semantics can be justified by finding a set of models. The models characterise the logic with respect to the fact that the set of theorems of the logic is exactly the set of sentences that are valid relative to that set of models. Therefore, game-theoretic settings can have models which offer an interpretation for solution concepts, such as a model of a game is a representation of the game that is being played, or a set of models of a game should determine a set of strategies. Therefore, one of the main theoretical tasks is to define a model in which one particular situation as playing a game, is presented. The model should represent players' choice, players' preferences and players' utility functions.

So far only the semantics of a formal logic is expressed. However, the syntax of a formal language is important [104]. Although we can reason about a fixed situation by working with sets (possible worlds) rather than formulae, we can formalise certain notations in a semantics-independent way with the syntax of the language. For example, based on one model, K_i mean "agent i has knowledge" and in another model, it means "agent i is rational". Another advantage of developing syntax is to differentiate logically equivalent formulae. For example, both formulae $K_i true$ and $K_i(p \vee \neg p)$ are logically equivalent, since $(p \vee \neg p)$ is a tautology. However, a computationally bounded agent may not identify $(p \vee \neg p)$ as a tautology and therefore, might not know it. In addition, syntax allows us to reason and to carry out proofs, which are usually achieved by induction on the structure of formulae. Moreover, formulae provide comparison tools for the same basic phenomena. In other words, formulae relate structures in a way that cannot be done using sets. For example, considering two epistemic structures M_1, M_2, both of which have 2 worlds (states). Agent 1 and agent 2 in M_1 know that p is true. Nevertheless, in M_2 at one of its worlds p is false and agent 2 does not know whether p is true or not. Consequently, $K_1 p \vee K_1 \neg p$ holds in every state of both M_1 and M_2, while $K_2 p \vee K_2 \neg p$. Therefore, a formula can compare different worlds, as $K_2 p \vee K_2 \neg p$ differentiates between worlds in M_1 and M_2.

Besides abstracting and specifying the behaviour of complex systems by

means of logic, in recent years researchers have been concerned about the problem of verifying the specifications of those complex systems. One of the most successful verification techniques among traditional approaches, such as simulation, testing, and deductive reasoning, is model checking [28]. The procedure is as follows: a real system, such as S, is first modelled to M_S by a logical language that encodes the computational traces of the system, and then it is formally verified that the system complies with certain desired properties such as P, which is expressed via a logical formula φ_P. Verification with model checking technique is defined as the problem of demonstrating whether or not $M_S \models \varphi_P$.

Several tools have been developed to perform this task automatically for temporal and epistemic models. However, traditional model checking tools do not allow for the representation of social interaction and autonomous behaviour of the agents under the lack of information.

To show that our language is a succinct and expressive language and our approach is practical for a reasonable class of applications, different representative scenarios will be investigated. Another motivation to model applications with this formal language is to show those properties that may be expressed using this language. Since extended epistemic logic is a suitable language to cope with uncertainty, it is also appropriate to analyse real life situations, such as detection of attackers in wireless networks and recognition of the benefits of using cloud computing. The reason to consider these two scenarios is that, they both demonstrate Bayesian game characteristics. Simply, both scenarios form different normal form game settings in which players are uncertain about the game that is being played.

1.2 Related Work

Modal logic has already been applied in conjunction with game theory to model interaction between agents in multi-agent systems. Moreover, some game theorists make the point that the formalism of an agent's information by means of logic should be included in game models [92]. Epistemic and temporal logic are used widely for modelling game settings and analysing them. Epistemic logic as the logic of knowledge and belief presents the knowledge

of agents. Temporal logic provides the ability for formalising the concept of time and it is used extensively for representing games with sequential actions over time [29], [3], [57], [25], [23], [42]. These logics do not provide enough flexibility to represent the interaction between agents in normal form games and Bayesian games.

A variety of logics have been proposed for reasoning about normal form games. We discuss those logics that are closely related to this thesis. van Benthem proposed dynamic epistemic logic [108], [110], [36] for representing the change of agents' knowledge. As this logic is the mixture of epistemic logic and pubic announcement logic, the lack of announcement in games makes it inapplicable for normal form games. By the help of Dynamic logic [91] two player normal from games are presented. However, by dynamic logic, games can be modelled while the outcome of the current game is dependent upon the previous game. Because this is not the desired property for reasoning about only one normal form games.

To the best of our knowledge, Bayesian games as concurrent games in which agents suffer from lack of information about the game is being played, have never been formalised by means of logic. However, concurrent games have attracted a good deal of attention in the logic community as they offer a similar structure to concurrent systems such as operating systems. Alternating-time temporal logic [3] is a suitable formalism for representing two player concurrent games in which players move simultaneously and the combination of two moves determines the next state. A variety of extensions of this logic have been proposed to increase the expressiveness of this logic such as alternating-time μ-calculus logic [49].

All the mentioned temporal logics represent perfect information games. Nevertheless, there are some studies that consider the uncertainty of players about the strategy such as alternating-time temporal observational logic [58]. In another study [99], incomplete information games are studied with the assumption that agents have access to only their system state, known as information. Most of similar work to these studies are suitable for agents with bounded recall of the past.

Another logic for concurrent games is concurrent dynamic game logic [111], which has two different sets of rules, one for propositions and another for games. This language provides the ability to represent games as propositions.

However this logic is not suitable for representing Bayesian games as it does not offer any means to model the private knowledge of agents in Bayesian games.

A combined logic of dynamic epistemic logic and probabilistic epistemic logic [39] is proposed as probabilistic dynamic epistemic logic [61]. Probabilistic epistemic logic semantically represents probability on the worlds that agents consider possible. The mentioned combination provides the dynamic power of dynamic epistemic logic for probabilistic epistemic logic. Therefore, the resulting logic has two sets of rules for propositions and games. However, representing the rationality of an agent is not offered in this language.

Beliefs in games with incomplete information is studied in [27]. A probabilistic Kripke structure is used and a notion of rationality is proposed. However, representing probabilistic beliefs is not considered in this work. In [10] also the notion of dynamic rationality is studied by conditional doxastic logic. This logic represents plausibility situations that are common situations in the belief revision structures. Backward induction is used as the method to detect solutions of games and therefore to define rational behaviour. As backward induction is based on reasoning backwards in time, it only applies to extensive form games.

Harsanyi [47] analysed the uncertainty about the structure of a game-specifically, about the players' payoff functions. To this end, he introduced the concept of a player's type, a fundamental concept which can be used to encode what the player believes the payoff functions might be, what the player believes other players believe the payoff functions might be, and so on indefinitely. Interactive epistemology deals with the beliefs and the knowledge of game players. It comes in two versions. The semantic approach represents knowledge by means of possible world structures, identifying the knowledge of a player i with the set of propositions that are true at all worlds which i cannot distinguish from the actual world [15], [48], [105]. The syntactic approach represents knowledge by sentences that are provable in extensions of various epistemic logics.

On a related front, syntactic investigations have found their way into the economic and game theoretic literature [9], [107], [17], [30]. Most of these applications are concerned with extensive games.

The syntactic approach with axiomatic systems for playing games has

recently been proposed by B. de Bruin [32]. Bonanno proposed a formula for representing Nash equilibrium strategies in games [16]. He also argued that epistemic logic is not only a useful tool to describe the rational behaviour of players, but also effective when it comes to recommending players how to act. The syntax of the language can be used as a query language to verify a game's properties with model checking techniques.

A variety of model checking tools exist to automatically verify properties in different systems. PRISM [65], Mocha [4] and MCK [44] are among the set of model checkers that are in some aspects related to our work. PRISM is a powerful model checker for probabilistic systems and it can be used for checking properties using different temporal logics. PRISM however is not applicable to model Bayesian games as it still does not support real and rational numbers and as a consequence unable to represent probabilities. For representing the type of players, a system is needed to support a larger set of numbers than the natural numbers. Mocha is another model checker that can be used to represent games and verify game properties [94], but its current stage does not support probabilistic systems. MCK is a model checker suitable for models that deal with knowledge of agents. Therefore, it supports epistemic logic, and it also supports probabilistic reasoning [52]. This system, however, does not provide enough flexibility for representing the type of Bayesian games.

Different applications are modelled as Bayesian games, such as cryptographic protocols [113], mechanism design, such as auctions [27], detection of attackers in wireless network [115] and cloud computing [64]. We provide formal descriptions for the two latter applications as they have not been analysed with epistemic logic and model checking techniques yet.

1.3 The Approach

The standard approach to develop a logic in the philosophy and computer science communities is, to build the language syntax first, and then assign formulas in the language truth values in a semantic structure. Epistemic logic is shown to be a precise language for formalising games [26], [40], [96]. We can represent a basic notion of individual rationality, often the idea of

maximisation of expected utility, and obtain what the payoff is given the informational context of a game. This means that, rather than saying that a set of strategies is a rational solution for a game, with epistemic logic, we try to understand what would be a rational decision for each agent, based on the agent's expectations on other agents' behaviour. In other words, in a game where agents interact with other rational decision makers, each agent's expectations about what it will gain based on its decision depend on what the agent expects the other decision-makers decide to do. We restrict ourselves to agents' attitudes that are fully self-examining, truthful and not revisable as proposed in [109]. In the literature of epistemic logic as well as game theory, it is commonly called knowledge [93].

This thesis has benefited from the studies of B. de Bruin [32] and G. Bonanno [19] but our approach is substantially different from them. Semantics that were proposed by [19] are only applicable to two person games, while we allow any number of players in the general case. Furthermore, the syntax in [32] focuses only on mixed strategies, while we consider pure strategies. More importantly, B. de Bruin's approach does not provide links between syntax and semantics by means of frame characterisation results. Semantics of our work is influenced by G. Bonanno [19] but it is different in several ways. G. Bonanno's approach is based on ordinal payoffs, while we assume von Neumann Morgenstern payoffs [100]. The significant difference to [19] is, that it only focuses on complete information games, while our approach covers both, complete information games and incomplete information games.

Following early work on epistemic game theory, we specify players' payoff, their beliefs about other players' types and their beliefs about beliefs by using type frames [7]. To get a succinct representation, a type frame is specified using Kripke structures. Roughly speaking, a type frame specifies a set Ω of possible worlds (or states), and for each player i, each world ω specifies a payoff for i, as well as what worlds player i considers possible at ω- called i's belief at ω.

In a Bayesian game, as the players should consider all games in one move, each player has to play a set of strategies simultaneously. Semantically, this mixed set of strategies is not interpreted as player's choices, but as its beliefs. Therefore, type frames in combination with epistemic logic offer a suitable formal representation of Bayesian games.

1.4 Aims and Contributions

We aim to develop an epistemic logic in an incomplete information game setting. In such a setting, an agent i is certain about its own payoff, but may have uncertainty about the other agents' payoffs, and about the other agents' beliefs about the whole payoff. Typically, the literature on epistemic logic in games focuses on extensive form games, in which players take turns to play. These games can represent many multi-agent system applications. However, simultaneous single moves are inevitable in many situations, for example auctions with private values which is, the players are assumed to how much the good is worth to them, but different players may value it differently. In order to analyse these situations, we may want to represent them in a formal manner. As just explained, this involves more than just studying these settings with game theory; one also needs tools to reason about incentives of agents. The central tool we will use throughout the thesis, is a formal language, called epistemic logic. Initially, in this logic, one could model the knowledge of a single agent only. We introduce an epistemic logic for analysing normal form games and Bayesian games. This epistemic logic with uncertainty obtains a more all-encompassing picture of practical reasoning for agents. This formal language enables us to define the desirability of outcomes not only based on players' actual payoff, but also based on their beliefs about other players' types. In this thesis, we use rationality to analyse which outcomes are consistent with the definition of solution concepts such as the Nash equilibrium.

Another restriction in the literature is to analyse games mostly semantically. The axioms for games have not been studied until recently [112]. Both, syntax and semantics, are considered as closely connected throughout this thesis, as they provide support for each other.

This formal language will be supported by a model checker. The term model checking refers to a collection of techniques for the automatic analysis of reactive systems that are assumed to maintain an ongoing interaction with their environment. Subtle errors in the design of these systems that often elude conventional simulation and testing techniques can be identified using this method. Model checking has been proven cost-effective and integrates well with conventional design methods. Therefore, it is being adopted as a

standard procedure for quality assurance [71].

In model checking, a logic formula φ specifies some system property to be checked. One approach in model checking is to have another logic formula, Γ, that precisely specifies the system. To check that a property holds, we have to prove that we can infer φ from Γ [20]. The model checker either confirms that the properties hold or reports that they are violated. Later, it provides a counter example: a run that violates the property. It can provide valuable feedback and points to design errors.

Our language is a concise and expressive language for describing complex properties of multi-agent systems. We showcase this logic in the context of two example games from wireless networks and cloud computing, presenting uncertainty over set of similar situations with different outcomes in both scenarios.

The main contributions of this thesis are as follows:

- We provide an extension of the epistemic logic for normal form games by considering player's preferences over a set of outcomes. The extension makes the formulation of rationality by epistemic logic more tangible. Parallel to syntax, semantics is extended to explicitly analyse the rationality and thus the solution concepts in these games.

- We provide a formal approach for establishing Bayesian games for multi-agent systems, which include a method for modelling agents' strategies, payoffs and preferences, and a technique to express agent's uncertainty over a set of normal form games. The approach can be used to specify agents' knowledge and uncertainty beliefs. Such an approach provides a foundation for reasoning about playing games given the lack of information about the game that is being played. It can also be used to describe the interaction and preferences of agents under uncertainty. It provides a basis to enable formal verification techniques to check whether a given agent behaves rationally. Therefore, an irrational behaviour can be revised before any major loss happens for agents.

- We provide a system that implements model checking techniques. The system can be used to verify game playing rules and rationality in normal form games and Bayesian games. It offers the modelling of these

games by the proposed formal language. The system demonstrates the advantages of the model checking verification technique and its associated reasoning technique, such as analysing a game property. The system can be used to prove the correctness act of agents in games with respect to the specification language. It also provides rational moves for an agent based on the other agents' actions which determines a solution concept. This contribution is presented in [33].

- As the first application of our system, we provide a method for the detection of attackers in wireless networks, by means of epistemic logic. The approach formalises the interaction between attacker nodes and regular nodes in the network, which facilitates the determination of attacking behaviour. It provides recommendation for having a secure network by formal definition of rational movement of regular nodes. A scenario can be modelled as a Bayesian game in epistemic logic and checked automatically by the model checker system with respect to the semantics of formal language. A certain situation, an attack to the wireless network, can be automatically detected by the system. This contribution is presented in [34].

- As the second application of our system, we provide an analysis for an ongoing interaction between cloud providers and cloud clients. The formal language enables us to formulate the utilisation of the cloud for clients' processing, covering dynamics of pricing offered by cloud providers. The approach can suggest advisable behaviour by contrasting their possible choices under uncertainty. The model checker system models the scenario and it helps to better understand of the financial aspects of the scenario. It recommends clients the potential balance between using clouds or private data centres by reasoning about rational reaction to different cloud price schemata with respect to formal specification language. This contribution is presented in [35].

These methods and techniques will be developed for representing and reasoning about Bayesian game settings. They can be used in any multi-agent systems where agents face variety of similar events with different outcomes in the way that the agent is not certain which event might be happening.

1.5 Outline of the Thesis

The thesis is organised as follows:

We start in Chapter 2 by showing the connection between multi-agent systems and game-theoretical setting. The definition of rational behaviour of agents justifies the important position of game theory in multi-agent systems. Game theory covers different ways of agent interaction, and we study the behaviour of self-interested agents in the context of normal form games and Bayesian games. We introduce different methods for detecting solution concepts in these games and provide examples. Modal logic provides an adequate foundation for representing and reasoning about games. We explain different formal languages which satisfy the requirement under different assumptions to formalise games. We provide a brief review of different logics which justifies the need for proposing an extended epistemic logic for these classes of games. As a tool to apply formal languages to model and check desired properties in a scenario, model checker techniques are explained. We compare different model checker tools to show their power and weaknesses.

Normal form games form the basic foundation of Bayesian games. Therefore, in Chapter 3 we first extend the already proposed epistemic logic for normal form games by adding axioms and evaluation rules. We show the rationality axiom and its truth value justification. We apply this formal language to model and reason about a simple two player normal form game (the prisoner's dilemma game). We propose a formal language for modelling and reasoning about Bayesian games. The language is the epistemic logic for normal form games with the power to express the type of players as probability beliefs. The semantics and syntax of the language, are presented. We analyse the rational behaviour of players in a Bayesian game. The chapter also presents a case study of reasoning about rational behaviour in a Bayesian game.

In Chapter 4, we turn to model checking tool as an automatic verification mechanism about Bayesian games. Here, we develop a model checker that receives epistemic logic specifications to model normal form games and Bayesian games. We show the input language and specification language and present different algorithms that are used to implement the tool.

Chapter 5 presents two applications for the formal language of Bayesian games. The applications are modelled as Bayesian games and then represented by epistemic logic. The first application is the detection of attackers in a wireless network. The detection action is formalised and the rational behaviour is explicitly determined. The rational behaviour of participants as a specification is checked by the model checker. Cloud computing is modelled to recognise the balance between using a cloud or an own data centre. The modelled scenario is formalised as a Bayesian game by the epistemic logic. The rational behaviour of the cloud provider and the client is formalised and the model checker used to verify the suitable choices under different circumstances for both players.

This thesis concludes in Chapter 6 which presents a summary of the contributions. We discuss future work that we are interested in pursuing as a result of this work.

Chapter 2

Game Theory and Modal Logic

In this chapter, we introduce the concept of agent interaction in game settings with respect of detecting the stable points with the best possible outcome. Since formalism by means of logic proves to be a strong tool for specifying and analysing games, we briefly summarise the formalisms to model normal form games and Bayesian game and reason about the rational behaviour of agents. After this, we summarise the approaches to model checking with different tools.

2.1 Multi-agent Systems and Game Theory

In the recent decades, it is expected from agents to decide for themselves and satisfy their design objectives by doing what they need to do autonomously [116]. The agents may be humans, individuals, groups, companies or artificial systems. They are broadly defined as actors in a system [97]. The term multi-agent systems obviously implies a system with more than one actor. These systems spread widely and play important roles in our society, ranging from wireless networks to cloud computing. In other words, multi-agent systems are systems which include multiple autonomous agents with either varying information or varying interests, or both.

However, the agents in multi-agent systems should behave rationally, which informally is defined as choosing actions that improve their success [53]. Nevertheless, there are different formal definitions for a rational agent. One of them is, that a rational agent acts as well as tries to opt its resources. That is, the profit for a rational agent is at least as high as the profit of any other agent running on the same system [40].

A rational agent is of real, practical interest because its behaviour is the best that can be obtained [13]. This is the definition of rationality which is adopted to game theory. In addition, game theory offers rules for analysing decision problems in multi-agent systems. In these problems, the utility of a given action depends on the actions of other agents. The classical game scenario involves a set of agents who make their decisions simultaneously, without the knowledge about the decisions of other agents. In a formal definition based on game theory, a rational agent embodies preferences, knowledge about the environment, and moreover knowledge about other agents with which it will interact.

Two common approaches in game theory are non-cooperative game theory and cooperative game theory. In the latter approach, the basic modelling unit, is a group of agents rather than an individual agent [100]. In this thesis, we are exclusively interested in non-cooperative games and we do not study cooperative games.

The non-cooperative games are referred to as the dominant approach in game theory [100]. In this branch, self-interested agents have a degree of preferences across a set of available alternatives [83]. To model an agent's interest based on that definition of preferences is referred to as utility theory. Throughout this thesis, the assumption is, that the agent has desires about how to act which is consistent with utility theory. We use words such as, "utility functions", "payoffs" or "outcomes" to implicitly refer to this theory.

Non-cooperative games cover a wide spectrum of games, ranging from normal form games to games with sequential actions, also known as extensive form games. The extensive form games, unlike normal form games, represent the sequence or time of the action a player takes in a game. However, both games are expected to be finite games. In contrast to infinite games, finite games provide the fundamental game-theoretic setting, but to represent realistic situations they are insufficient. Hence, infinite games are suitable to model complex and real-life scenarios.

Infinite games cover different games, such as repeated games, stochastic games, Bayesian games, and congestion games. Naturally, for repeated games and stochastic games, time is the most important factor, while Bayesian games have different structures and involve uncertainty.

In this thesis, we study normal form games as the basic structure of game

theory and Bayesian games that let agents face uncertainty and hold private information.

2.1.1 Normal Form Games

The most familiar representation of strategic interactions in game theory are normal form games. These games are also known as the strategic form. A natural way to present these games is via an n-dimensional matrix. An example is given in table 2.1 for the well known game of the prisoner's dilemma.

		player 2	
		cooperates	defects
player 1	cooperates	(2,2)	(0,3)
	defects	(3,0)	(1,1)

Table 2.1: Prisoner's dilemma

There are different methods to play normal form games. Agents should follow some rules to achieve solutions of games. These rules should identify some outcomes that are interesting in one sense or another [75]. We briefly review three main solutions.

- Iterated elimination of strictly dominated strategies: a strictly dominated strategy is the strategy that pays less than other available strategies, regardless of what the other players play. Therefore, in a matrix presentation of a game, we delete all cells that relate to this strategy. For example, in the game shown in table 2.1 we delete the rows and columns that relate to the strategy "cooperate". Therefore, the only outcome is "defect" for both players.

- Pareto optimality: it tries to identify an outcome that is better than another from the point of view of an outsider. Formally, it says that there should be a set of strategies that some agents cannot be made better off by making other agents worse off.

- Nash equilibrium: this is a stable point in a game, at which no agent would want to change its strategy given the strategies of other agents.

In this thesis, as the solution for those games we try to detect the pure Nash equilibrium. The reason is that the complexity of finding a sample (pure or mixed) Nash equilibrium in a finite game with two or more agents is a polynomial parity argument directed graph (PPAD) [100] which is a less known complexity class. PPAD is a subset of the total function nondeterministic polynomial (TFNP) class. A binary relation $P(x, y)$ belongs to TFNP class if and only if there is a deterministic polynomial time algorithm that can determine whether $P(x, y)$ is true given both x and y, and for every x, there exists a y such that $P(x, y)$ is true.

To detect the solution in a game, we go through each step shown below [77].

1. detect all pure strategies for player i.
2. for each pure strategy of player i, determine the pure strategies of other players.
3. determine the best utility among a set of combination of the pure strategy of player i and other players' strategies.

Now we find the solution for the prison dilemma game that is shown in table 3.1. One pure strategy for player 1 is "cooperates" and another one is "defects". The utilities for player 1, if it cooperates, are 2 and 0. If it defects, the utilities are 3 and 1. Similarly, the pure strategies for player 2 are either "cooperates" or "defects". The utilities for player 2 when it cooperates are 2 and 3 and if it defects are 0 and 1. The best sets of utilities are (2,2) and (1,1). The sets of utilities (3,0) and (0,3) are unstable. In the first case, player 2 will deviate from it to increase its outcome. Likewise, the utility (0,3) will not be chosen by player 1, this player is able to gain a better outcome than this utility.

In the next section, we review different models of Bayesian games which consist of different normal form games.

2.1.2 Bayesian Games

Game theory studies decision problems in which the utility of a given action depends not only on the actions of other agents but also on chance of events

in the environment. If we want to model these events, the state of the world as a game depends on randomness in the environment. These settings are known as Bayesian games or games of incomplete information, because agents are uncertain about the very game being played [120]. Agents' uncertainties are presented as a probability distribution over a set of possible games.

There are several ways for representing Bayesian games, as information set [7], extensive form with chance moves [100], epistemic type [47] or interactive epistemology [12]. The difference between the epistemic type and interactive epistemology is that the latter one restricts agents' beliefs about their own knowledge and their opponents' strategies.

In this thesis, we choose the epistemic type, because it offers a presentation that can be represented by epistemic logic.

Two accepted assumptions about these games are :

- All normal form games that form a Bayesian game have the same number of agents and the same number of strategies for each agent; these games only differ in their utility functions.
- The type (probability belief) of each agent is a posterior, obtained by conditioning a common prior on individual private signal.

The second assumption is important, since Bayesian games define, besides the uncertainties of agents about the game being played, the agent's beliefs about the beliefs of other agents about the game that is being played. Therefore, an entire infinite hierarchy of nested beliefs with this assumptions is defined. John Harsanyi [47] suggested a solution that avoided the difficulty of having to deal with infinite hierarchies of beliefs, by providing a much more workable implicit, encapsulated model. The key notion in Harsanyi's model is the type (second assumption). Each agent can be of several types, where a type is to be thought of as a full description of the agents beliefs about the state of nature (the data of the game), beliefs about the beliefs of other agents about the state of nature and about its own beliefs, etc. This assumption is necessary to formulate the main ideas in Bayesian games[100]. Therefore, in Bayesian games we have expected utility which is:

$$ExpU_i(a_i, a_{-i}) = \sum_{j=1}^{m} P(\theta_{ij}, \theta_{-i})(u_i(a_i, a_{-i}|\theta_{ij}, \theta_{-i})) \tag{2.1}$$

We assume that the number of players is n and each player has m types. $\theta_{-i} = \{\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_n\}$ is the set of types other players have. $P(\theta_{ij}, \theta_{-i})$ is the probability of having type j of player i with a set of types of other players. $u_i(a_i, a_{-i}|\theta_{ij}, \theta_{-i})$ is the utility of player i playing strategy a_i in combination with other players' strategy sets $a_{-i} = \{a_1, ..., a_{i-1}, a_{i+1}, ..., a_n\}$ given $(\theta_{ij}, \theta_{-i})$.

The best response and consequently Bayes-Nash equilibrium is defined based on the equation 2.1. The best response of agent i to the set of other players' strategies a_{-i} is a set of strategies that:

$$BestRes_i(a_{-i}) = \arg\max_{a'_i \in a_i} ExpU_i(a'_i, a_{-i}) \tag{2.2}$$

Equation 2.2 intuitively means that a set of strategies of all available strategies for player i ($a'_i \in a_i$) for which given $ExpU$ function attains its maximum value.

A Bayes-Nash equilibrium is a set of strategies that satisfies the following statement

$$\forall i \; a_i \in BestRes_i(a_{-i}).$$

Despite its similarity to the Nash equilibrium, it seems conceptually more complicated. However, the solution is to model Bayesian games as normal form games. Formally known as induced normal form for Bayesian games, the induced normal form presentation of Bayesian games has expected utilities for each player. Furthermore, the Bayes-Nash equilibria of a Bayesian game are the Nash equilibria of its induced normal form.

As an example, we define an arbitrary Bayesian game, and present its induced normal form and Bayes-Nash equilibria. Consider the Bayesian game in table 2.2, which is constructed from 16 normal form games. The strategies for player 1 in each normal form game of the Bayesian game is U and D,

player 2																
player1		Θ_{2_1}			Θ_{2_2}			Θ_{2_3}			Θ_{2_4}		
	Θ_{1_1}	game1			game2			game3			game4		
			L	R		L	R		L	R		L	R
		U	3,2	1,1	U	4,4	1,7	U	5,5	1,1	U	3,0	0,3
		D	1,1	2,3	D	7,1	2,2	D	1,1	3,3	D	0,3	3,0
		p=0.1			p=0.09			p=0.12			p=0.13		
	Θ_{1_2}	game5			game6			game7			game8		
			L	R		L	R		L	R		L	R
		U	4,1	1,4	U	3,3	0,0	U	5,5	0,6	U	6,4	0,0
		D	1,4	4,1	D	0,0	1,1	D	6,0	4,4	D	0,0	4,6
		p=0.05			p=0.1			p=0.02			p=0.11		
	Θ_{1_3}	game9			game10			game11			game12		
			L	R		L	R		L	R		L	R
		U	4,4	1,6	U	3,2	2,3	U	5,5	2,2	U	6,5	2,2
		D	6,1	3,3	D	2,3	3,2	D	2,2	4,4	D	2,2	5,6
		p=0.01			p=0.08			p=0.02			p=0.02		
	Θ_{1_4}	game13			game14			game15			game16		
			L	R		L	R		L	R		L	R
		U	4,4	0,0	U	6,5	2,2	U	2,0	0,2	U	6,6	0,7
		D	0,0	1,1	D	2,2	5,6	D	0,2	2,0	D	7,0	3,3
		p=0.05			p=0.09			p=0.02			p=0.08		

Table 2.2: A Bayesian game with 16 normal form games

which could be substituted with any desired action. Similarly for player 2 is L and R.

In the induced normal form game each player has sixteen possible strategies. Each player has four types and two actions, thus for every type it has two actions therefore, $2^4 = 16$. Then player 1's sixteen strategies are labelled as UUUU, UUUD, UUDU, UDUU, DUUU, UUDD,UDUD, DUUD, DUDU, DDUU, UDDU, UDDD, DUDD, DDUD, DDDU, DDDD. Note that UUUU means that player 1 chooses U regardless its type, UUUD means that it chooses U when it has type θ_{1_1} and U in its other types. Similarly, we can denote the strategies of player 2 in the Bayesian game by LLLL, LLLR, LLRL, LRLL, RLLL, LLRR, LRLR, RLLR, RLRL, RRLL, LRRL, LRRR, RLRR, RRLR, RRRL, RRRR.

We define a 16×16 normal form game in which these are the sixteen strategies of the two players, and the utilities are the expected utilities in the

individual games, given the players' beliefs. For example, player 1's expected utility under the set of strategies (DUDU, LLRR) is calculated by:

$$
\begin{aligned}
u_1(DUDU, LLRR) = \sum_{j=1}^{4} P(\theta_{1_j}, \theta_2)(u_1(a_1, a_2|\theta_{1_j}, \theta_2)) = \\
p(\theta_{1_1}, \theta_{2_1})u_1(D, L, \theta_{1_1}, \theta_{2_1}) + p(\theta_{1_1}, \theta_{2_2})u_1(D, L, \theta_{1_1}, \theta_{2_2}) + \\
p(\theta_{1_1}, \theta_{2_3})u_1(D, R, \theta_{1_1}, \theta_{2_3}) + p(\theta_{1_1}, \theta_{2_4})u_1(D, R, \theta_{1_1}, \theta_{2_4}) + \\
p(\theta_{1_2}, \theta_{2_1})u_1(U, L, \theta_{1_2}, \theta_{2_1}) + p(\theta_{1_2}, \theta_{2_2})u_1(U, L, \theta_{1_2}, \theta_{2_2}) + \\
p(\theta_{1_2}, \theta_{2_3})u_1(U, R, \theta_{1_2}, \theta_{2_3}) + p(\theta_{1_2}, \theta_{2_4})u_1(U, R, \theta_{1_2}, \theta_{2_4}) + \\
p(\theta_{1_3}, \theta_{2_3})u_1(D, L, \theta_{1_3}, \theta_{2_3}) + p(\theta_{1_3}, \theta_{2_2})u_1(D, L, \theta_{1_3}, \theta_{2_2}) + \\
p(\theta_{1_3}, \theta_{2_3})u_1(D, R, \theta_{1_3}, \theta_{2_3}) + p(\theta_{1_3}, \theta_{2_4})u_1(D, R, \theta_{1_3}, \theta_{2_4}) + \\
p(\theta_{1_4}, \theta_{2_1})u_1(U, L, \theta_{1_4}, \theta_{2_1}) + p(\theta_{1_4}, \theta_{2_2})u_1(U, L, \theta_{1_4}, \theta_{2_2}) + \\
p(\theta_{1_4}, \theta_{2_3})u_1(U, R, \theta_{1_4}, \theta_{2_3}) + p(\theta_{1_4}, \theta_{2_4})u_1(U, R, \theta_{1_4}, \theta_{2_4}) = \\
0.1 * 1 + 0.09 * 7 + 0.12 * 3 + 0.13 * 3 + 0.05 * 4 + \\
0.01 * 3 + 0.02 * 0 + 0.11 * 0 + 0.01 * 6 + 0.08 * 2 + \\
0.02 * 4 + 0.02 * 5 + 0.05 * 4 + 0.09 * 6 + 0.02 * 0 + \\
0.08 * 0 = 2.85
\end{aligned}
$$

We can construct the complete utility matrix by computing all the possible combination of strategies at given types. The induced normal form that is achieved by this computation is shown in table 2.3. Now, the game can be analysed in a straightforward fashion. For example, we can determine player 2's best response to DUUD is RLLR with an utility of 3.4.

When we have an application based on game theoretic settings, agents autonomously perform some actions. The agents need to reason about what they know and what they believe precisely to make a decision. One way is to represent games with formal languages. In the next section, we provide a brief survey about the connection between game theory and modal logic.

Δ Table 2.3: The induced normal form of the Bayesian game shown in figure 2.2

	RLRL	RRLL	LRRL	LRRR	RLRR	RRLR	RRRL	RRRR
UUUU	2.81,3.37	2.12,3.09	3.21,2.99	2.13,2.64	1.73,3.02	1.04,2.74	1.87,3.07	0.79,2.72
UUUD	2.29,2.46	1.6,2.18	2.69,2.08	2.73,2.69	2.33,3.07	1.64,2.79	1.35,2.16	1.39,2.77
UUDU	3.01,3.38	1.92,3.02	3.41,3	2.33,2.65	1.93,3.03	0.84,2.67	2.07,3.08	0.99,2.73
UDUU	1.99,2.95	2.8,3.57	3.89,3.47	2.81,3.12	0.91,2.6	1.72,3.22	2.55,3.55	1.47,3.2
DUUU	1.99,2.95	2.94,2.69	2.41,2.53	1.33,2.18	2.55,2.62	1.86,2.34	2.69,2.67	1.61,2.32
UUDD	3.01,3.38	1.4,2.11	3.41,3	2.93,2.7	2.53,3.08	1.44,2.72	2.07,3.08	0.99,2.73
UDUD	1.47,2.04	2.28,2.66	3.37,2.56	3.41,3.17	1.51,2.65	2.32,3.27	2.03,2.64	2.07,3.25
DUUD	3.11,2.06	2.42,1.78	1.89,1.62	1.93,2.23	3.15,2.67	2.46,2.39	2.17,1.76	2.21,2.37
DUDU	3.83,2.98	2.74,2.62	2.61,2.54	1.53,2.19	2.75,2.63	1.66,2.27	2.89,2.68	1.81,2.33
DDUU	2.81,2.55	3.62,3.17	3.09,3.01	2.01,2.66	1.73,2.2	2.54,2.82	3.37,3.15	2.29,2.8
UDDU	2.19,2.96	2.6,3.5	4.09,3.48	3.01,3.13	1.11,2.61	1.52,3.15	2.75,3.56	1.67,3.21
UDDD	1.67,2.05	2.08,2.59	3.57,2.57	3.61,3.18	1.71,2.66	2.12,3.2	2.23,2.65	2.27,3.26
DUDD	3.11,2.06	2.22,1.71	1.89,1.62	1.93,2.23	3.15,2.67	2.46,2.39	2.17,1.76	2.21,2.37
DDUD	2.29,1.64	3.1,2.26	2.57,2.1	2.61,2.71	2.33,2.25	3.14,2.87	2.85,2.24	2.89,2.85
DDDU	3.01,2.56	3.42,3.1	3.29,3.02	2.21,2.67	1.93,2.21	2.34,2.75	3.57,3.16	2.49,2.81
DDDD	2.49,1.65	2.9,2.19	2.77,2.11	2.81,2.72	2.53,2.26	2.94,2.8	3.05,2.25	3.09,2.86

Table 2.3: Second part of the induced normal form of the Bayesian game shown in figure 2.2

	LLLL	LLLR	LLRL	LRLL	RLLL	LLRR	LRLR	RLLR
UUUU	4.4,3.31	3.32,2.96	4.15,3.29	3.46,3.01	3.06,3.39	3.07,2.94	2.38,2.66	1.98,3.04
UUUD	3.88,2.4	3.92,3.01	3.63,2.38	2.94,2.1	2.54,2.48	3.67,2.99	2.98,2.71	2.58,3.09
UUDU	4.2,3.24	3.12,2.89	4.35,3.3	3.26,2.94	2.86,3.32	3.27,2.95	2.18,2.59	1.78,2.97
UDUU	3.58,2.89	2.5,2.54	3.33,2.87	4.14,3.49	2.24,2.97	2.25,2.52	3.06,3.14	1.16,2.62
DUUU	3.6,2.85	2.52,2.5	3.35,2.83	2.66,2.55	3.88,2.99	2.27,2.48	1.58,2.2	2.8,2.64
UUDD	3.68,2.33	3.72,2.94	4.35,3.3	3.26,2.94	2.34,2.41	3.87,3	2.78,2.64	2.38,3.02
UDUD	3.06,1.98	3.1,2.59	2.81,1.96	3.62,2.58	1.72,2.06	2.85,2.57	3.66,3.19	1.76,2.67
DUUD	3.08,1.94	3.12,2.55	2.83,1.92	2.14,1.64	3.36,2.08	2.87,2.53	2.18,2.25	3.4,2.69
DUDU	3.4,2.78	2.32,2.43	3.55,2.84	2.46,2.48	3.68,2.92	2.47,2.49	1.38,2.13	2.6,2.57
DDUU	2.78,2.43	1.7,2.08	2.53,2.41	1.7,2.08	3.06,2.57	1.45,2.06	2.26,2.68	1.98,2.22
UDDU	3.38,2.82	2.3,2.47	3.53,2.88	3.94,3.42	2.04,2.9	2.45,2.53	2.86,3.07	0.96,2.55
UDDD	2.86,1.91	2.9,2.52	3.01,1.97	3.42,2.51	1.52,1.99	3.05,2.58	3.46,3.12	1.56,2.6
DUDD	2.88,1.87	3.12,2.55	2.83,1.92	1.94,1.57	3.16,2.01	2.87,2.53	2.18,2.25	3.4,2.69
DDUD	2.26,1.52	2.3,2.13	2.01,1.5	2.14,1.64	2.54,1.66	2.05,2.11	2.86,2.73	2.58,2.27
DDDU	2.58,2.36	1.5,2.01	2.73,2.42	3.14,2.96	2.86,2.5	1.65,2.07	2.06,2.61	1.78,2.15
DDDD	2.06,1.45	2.1,2.06	2.21,1.51	2.62,2.05	2.34,1.59	2.25,2.12	2.66,2.66	2.38,2.2

2.2 Game Theory and Modal Logic

The connection between game theory and modal logic has been developed recently [112]. Most of the research that has been done by the logic community has restricted their attention to mainly two player extensive games (games with sequential actions over time) of prefect information which are just competitive in which players either lose or win [118], [16], [46]. In spite of the restriction on games, presenting and reasoning about these games turn out to be complicated. Nevertheless, more recent work on logic has extended the game theoretic toolbox by introducing cooperative game theory, imperfect information and multi-player games of more than two players [112].

As games can be modelled as tree structures, Kripke models [63] provide a natural way to represent them. A Kripke model can be considered as a directed graph so that nodes can represent different concepts such as states, times, situations, worlds and other properties. Each edge represents a binary relation between nodes. The formal definition of Kripke model $\mathfrak{M}$ is:

$$\mathfrak{M} = \{\Omega, R, V\}$$

Where Ω is a non-empty set and its elements are nodes. R is a binary relation on Ω. V is the function (evaluation of formulae) that assigns to each atomic proposition p a subset $V(p)$ of Ω. $V(p)$ is the set of nodes in $\mathfrak{M}$ at which p is true.

A Kripke model can be considered as the basic model in a modal language. In modal logic, a formula is evaluated inside models at particular points. This is the definition of semantics for modal logic. The syntax of modal logics is the same language of propositional logic augmented with additional modal operators.

Different categories of modal logics are used for representing and reasoning about games such as dynamic game logic [78] and alternating-time μ-calculus [111]. However, two prominent logics are epistemic logic and temporal logic. Epistemic logic is the logic of knowledge and belief. It was introduced by Jaako Hintikka [50] and flourished by Fagin, Halpern, Moses, and Vardi [40]. In epistemic logic the accessibility relation R of Kripke model is for interpreting the modal operator K. In this logic, the set of states that are accessible from a state are epistemic alternatives for this state. Therefore, an

agent in this state is not able to distinguish between these accessible states due to its lack of knowledge.

Temporal logic deals with time and the modal operates in this logic represent different concepts of time. It has different versions such as linear temporal logic, computation tree logic (CTL) [29] (branching time logic) and alternating-time temporal logic (ATL)[3]. Temporal logic is extensively used in connection with game theory [57], [25], [23], [42]. However, the nature of games in this thesis, in which agents play simultaneously single moves, temporal logic renders unsuitable to be applied.

2.2.1 Representing and Reasoning about Normal Form Games

Normal form games can be defined as the basic presentation in game-theoretical settings. However, reasoning about this basic structure appears to be complicated. There are several different studies about this field and we provide a brief survey about the ones related to this thesis.

2.2.1.1 Dynamic Epistemic Logic

In [108], [110] and [36] van Benthem investigated normal form games (strategic games) with dynamic epistemic logic. With this logic, we can express how knowledge changes based on decision making in games. Thus, we can describe acts of information flow, such as public announcements or observations. However, this logic is the combination of epistemic logic and public announcement logic. If in a game either no announcement is provided or no information about the announcement is available, the logic is not applicable.

The epistemic dynamic language without common knowledge has the following formation rules:

$$\Phi := p \in P \mid \neg\phi \mid \phi \ \vee \ \psi \mid K_i\phi \mid [\chi]\psi$$

where P is a set of propositions, a modal operator K_i for every agent i with $K_i\phi$ means "agent i knows ϕ". The modal operator $[\chi]\psi$ means that "after the truthful public announcement of χ , ψ is true".

In addition to Kripke model $\mathfrak{M} = \{\Omega, R, V\}$, submodel $\mathfrak{M}$, $\mathfrak{M}_{|\chi}$ is defined

as all states in which χ is true. Formally, we have $\mathfrak{M}_{|\chi} = \{\Omega', R', V'\}$ where $\Omega' = \{\omega \in \Omega | (\Omega, \omega) \models \chi\}$. The valuation of this modal operator is defined as:

$$\mathfrak{M}, s \models [\chi]\psi \text{ iff } (\mathfrak{M}, s \models \chi \Rightarrow \mathfrak{M}_{|\chi}, s \models \psi)$$

The axioms of this logic are the following:

- $([\chi]\varphi \wedge [\chi](\varphi \rightarrow \psi)) \rightarrow [\chi]\psi$
- $[\chi]p \leftrightarrow (\chi \rightarrow p)$
- $[\chi]\neg\varphi \leftrightarrow (\chi \rightarrow \neg[\chi]\varphi)$
- $[\chi]K_i\varphi \leftrightarrow (\chi \rightarrow K_i[\chi]\varphi)$

with the following inference rule:

- $\vdash \varphi \Rightarrow [\chi]\varphi$

The rationality of agents and solutions of games are not studied by this logic.

2.2.1.2 Dynamic Logic

In [91], two player normal from games are presented by dynamic logic. The basic concept in that work is to represent "composition of game play pairs". For reasoning about normal form games, it is considered that numbers of these games are played in a manner that the outcome of the current game is dependent upon the previous game. Although they start by modelling one game as the basic structure, the reasoning performed in a single stage is outcome based. We briefly present the syntax and the semantics of that work.

The proposed language has the following grammar:

$$\Phi := p \in P \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle g, \eta \rangle^{\forall}\alpha$$

where P is a countable set of propositions, g is a normal form game, $\eta \subseteq \Sigma^g$ and Σ^g is a set of action symbols which represent moves of players in game g. Thus, η represent a set of strategies in g. Intuitively, $\langle g, \eta \rangle^{\forall}\alpha$ states that formula α holds at states where strategies of g are specified by η.

From the semantic point of view, normal form games are modelled as a tree of depth one, at which edges are labelled by pairs of actions, one for each player. The game g is a set $\{S, \rightarrow, s_0, \lambda\}$ where S is the set of states, s_0 is the root of the tree. The transition function $\rightarrow$ assigns to $s_0 \times \Sigma$ a state and finally a utility function λ that assigns to each state a pair of players' payoffs. The game model $M = (g, V)$ where $g = \{S, \rightarrow, s_0, \lambda\}$ and a valuation function is defined as $V : S \rightarrow 2^P$.

A set of special propositions $\Theta_i = \{\theta_i^1, ..., \theta_i^l\}$ is defined for player i when the number of actions for player i is l. Each θ_i^j is the jth action of player i. The payoffs of the players are important and the logic should provide reasoning about this concept. The only truth valuation of this work that is not obvious is:

- $M, s \models \langle g, \eta \rangle^{\forall} \alpha$ iff s is not a leaf node and $\forall s' \in tail(g, \eta), M, s' \models \alpha$

where $tail(g, \eta) = \{s' \,| s_0 \xrightarrow{a} s' \text{ and } a \in \eta\}$

The strategy b is better that b' for player i, given that a strategy x for the other player $\bar{i}$ is:

$$Better_x^i(b, b') \equiv \bigwedge_{\theta_i \in \Theta_i} (\langle g, (b', x) \rangle^{\forall} \theta_i \supset \langle g, (b, x) \rangle^{\forall} \theta_i)$$

The best response is expressed by this logic as b is the best response of player i for x as $BR_x^i(b) \equiv \bigwedge_{b' \in \Sigma_i^g} Better_x^i(b, b')$.

2.2.1.3 Epistemic Logic

Bonanno [16] models normal form games semantically. Kripke model is used with m modal operators $\Box_i$ where $i = \{1, ..., m\}$. Therefore, $M = \{\Omega, R_1, ..., R_m, V\}$ where R_i is a binary relation on Ω and we have $\alpha R_i \beta$ when state β is R_i-accessible from α. A function $\sigma : \Omega \rightarrow S$ that assigns to each state a set of strategies is added to M in order to obtain a model of a particular game G. Important notation, which is introduced in that work, is:

- r_i: means player i is rational.

Based on the defined model M, we can say $\alpha \in V(r_i)$ if and only if :

1. player i is not uncertain about the strategy it is playing because:

$$\alpha R_i \beta \quad then \quad \sigma_i(\alpha) = \sigma_i(\beta)$$

2. $\sigma_i(\alpha)$ maximises i's utility given its belief.

However, the solution concept which is studied in this work is based on the iterative deletion of strictly dominated strategies.

Another important contribution of that work is the following formula:

$$(2.3) \qquad {}_{\wedge}(u_i = p_i) \wedge {}_{\wedge}\Box_i((u_i = q_i) \rightarrow (q_i \leq p_i)) \leftrightarrow {}_{\wedge}(u_i = p_i) \wedge Nash$$

The formula is equivalent to say, if the utility of player i is p_i, the strategy played belongs to the Nash equilibrium of the game, if and only if no player i can unilaterally deviate and achieve something like q_i better than p_i. In the equation 2.3, some notations are assumed, e.g., $q \leq p$ for any natural number q and p, $u_i = p$ for every player i and number p states player i's utility at any state and finally $Nash$ denoting strategy which has Nash equilibrium characteristics. Also the notation ${}_{\wedge}$ means all sentences from a finite set which can be a set of players, or actions, or utilities. The differences of that work from the work of this thesis are:

- We provide a syntax for representing normal form games and the rationality of the player.
- We have not used iterated elimination of strictly dominated strategies as the method to find the solution, instead we focus on pure Nash equilibrium at which no player can gain better by deviating from the set of strategies that belong to the Nash equilibrium.

Although in that work no axiom system was proposed, de Bruin [32] provides axioms for normal form games and extensive games. The main focus of de Bruin's work is to formalise the characterisation of solution concepts in games using epistemic constructs. In addition, it is argued that given specific assumptions about a player's utility and rationality, we can predict the decision of the player due to epistemic characterisation formalisation. This formal language has besides atomic propositions, the certain propositions given as follows:

- The proposition letters $\mathbf{i}_m$ stand for the statement 'i plays its mth strategy i_m'.
- The proposition $\mathbf{u}_i(1_{k_1}, ..., N_{k_1}) = \mathbf{r}_{i,1_{k_1},...,N_{k_1}}$ denotes that the utility for player i, when the strategy profile $(1_{k_1}, ..., N_{k_1})$ is played, equals the number r.
- R is a set of countably many symbols such as r. The elements of R represent real numbers,but R is not taken as the set of real numbers.
- The proposition $\mathbf{rat}_i$ denotes the rationality of player i, in the sense that i is an expected utility maximiser.

In [32] axioms are proposed as "axioms for game playing situations" which are for 2 player normal form games:

- Start$\geq$ 1: $\bigvee_m \mathbf{i}_m$.
- Start$\leq$ 1: $\bigwedge_{m \neq n} \neg(\mathbf{i}_m \wedge \mathbf{i}_n)$.
- KnStart: $\bigwedge_m(\Box_i \mathbf{i}_m \leftrightarrow \mathbf{i}_m)$.
- KnUt: $\mathbf{u}_i(k,l) = \mathbf{r} \rightarrow K_i \mathbf{u}_i(k,l) = \mathbf{r}$.

where m ranges over the strategies available to player i. These axioms determine what players do and what they know when they play normal form games. The first axiom says, every player plays at least one strategy, while the second does not allow players to play more than one strategy. Moreover, axiom KnStart stipulates that every player knows its chosen strategy, and likewise axiom KnUt requires players to have a correct knowledge/belief about their own utility functions. The axiom RAT which is the formalism of utility maximisation captures the $\mathbf{rat}_i$ as the following implication:

$$\mathbf{rat}_i \leftrightarrow \bigwedge_m((\Box_i \bigwedge_{k,l} \mathbf{u}_i(k,l) = \mathbf{r}_{i,k,l} \wedge \bigwedge_l \mathbf{P}_i(\mathbf{j}_l) = \mathbf{p}_l \wedge \mathbf{i}_m) \rightarrow \bigwedge_k \sum_l \mathbf{p}_l.\mathbf{r}_{i,m,l} \geq \sum_l \mathbf{p}_l \mathbf{r}_{i,k,l}) \tag{2.4}$$

where $\mathbf{P}_i$ presents the probabilistic belief of player i. Equation 2.4 states that when player i chooses to play its mth strategy while it has certain beliefs about utility (captured by $\mathbf{r}_{i,k,l}$) and about its prospective strategies (captured by $\mathbf{p}_l$) then the mth strategy is better than other, given its beliefs. Therefore, player i is an expected utility maximiser when it has the information above.

By considering all the axioms in [32], a proof is provided for the theorem proposed by Aumann and Brandenburger [6] for two player normal form games:

- All players know their own utility function, which is: $\bigwedge_i \Box_i \bigwedge_{k,l} \mathbf{u}_i(k,l) = \mathbf{r}_{i,k,l}$
- All players are rational, which is: $\bigwedge_i \mathbf{rat}_i$
- All players know each player's actual choice of an action, which is: $\Box_2 \mathbf{1}_m \wedge \Box_1 \mathbf{2}_n$

Then the actual action profile played constitutes a Nash equilibrium. It means the solution concept for player 1 is

$$\bigwedge_k \mathbf{r}_{1,m,l} \geq \mathbf{r}_{1,k,l} \tag{2.5}$$

and for player 2 is

$$\bigwedge_l \mathbf{r}_{2,k,n} \geq \mathbf{r}_{2,k,l} \tag{2.6}$$

.

Our work differs from [32] because:

- We provide a semantics for representing normal form games and the rationality of the player.
- We consider players' preferences over a set of strategies.
- We extend the language for *n player normal form games.*

As we study the formalisation of normal form games we have the basic block foundation to study Bayesian games.

2.2.2 Representing and Reasoning about Simultaneous Games

To the best of our knowledge, Bayesian games have never been formalised. Although there are other approaches to study simultaneous, concurrent or parallel games, none of these approaches consider the lack of player's knowledge about the game that is being played. We briefly review some of these formalisations for concurrent games.

2.2.2.1 Alternating-time Temporal Logic

To verify parallel/concurrent programs such as operating systems and network communication protocols, linear time temporal logic is proposed [85]. Similar to all modal languages which study different modes of truth, this logic provides a formal system for qualitatively describing and reasoning about the truth values of assertions of time varying events. As linear time temporal logic singly focuses on the moment that has only one possible future moment, the sibling of this logic, computational tree logic [37] as branching time logic, studies events at a moment, where time may split into alternate courses representing different possible futures. However, this logic describes the transition of systems at a very abstract level in which it is not important what or who is involved in making transitions. Pauly in [81] and [82] provides a formalisation based on computational tree logic by which we can describe agents' abilities to influence system transitions. This formalism is important especially in game-theoretical scenarios at which we have to explicitly represent how different agents can contribute to the system's evolution. Nevertheless, Pauly just focuses on the cooperation of agents. It means that the language describes what strategy is achieved jointly. However, the strategies of each player in most games are required to be represented explicitly and this can be done by alternating-time temporal logic [3]. Alur, Henzinger, and Kupferman formalised this logic of two player concurrent games, which means in each round, both players choose their moves simultaneously and independently from each other, and the combination of two moves determines the next state. Semantically, the authors use alternating transition systems to model concurrent games. A tuple $S = \{\Sigma, Q, \Delta, \Pi, \pi\}$ is alternating transition system with the following components:

- Σ is the finite set of players.
- Q is set of states.
- $\Delta = \{\delta_i : Q \to 2^{2^Q} | i \in \Sigma\}$ is a set of transition functions. For every player, it maps each state to a non-empty set of choices. Generally, each choice is a set of possible next states.
- Π is a set of propositions.
- $\pi : \Pi \to 2^Q$ assigns each proposition to a set of states.

A game logic L is proposed and its formulae are interpreted over the states of alternating transition systems. In other words, for every L-formula φ and every alternating transition system S.

The grammar of this language is:

$$\varphi := \ \mid p \mid \neg\varphi \mid \varphi_1 \ \vee \ \varphi_2 \mid \langle\langle I\rangle\rangle \bigcirc \varphi \mid \langle\langle I\rangle\rangle \Box\varphi \mid \langle\langle I\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$$

where p is a proposition, $I \subseteq \Sigma$ belongs to teams of players. The formula $\langle\langle I\rangle\rangle \bigcirc \varphi$ says that the team of players I has a joint strategy for achieving φ at the next step. $\langle\langle I\rangle\rangle \Box\varphi$ expresses that the team of players I can maintain φ forever in the future. $\langle\langle I\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ states that the team of players I can maintain φ_1 until φ_2 holds.

This logic is extended to alternating-time μ-calculus logic [49], which is strictly more expressive than alternating-time temporal logic. However, all mentioned temporal logics represent perfect information games. In [119] and [51], the alternating-time temporal logic is extended with an epistemic accessibility relation $\sim_i$ for each player i. This logic is called alternating-time temporal epistemic logic and adds to alternating-time temporal logic operators for representing knowledge in the world of incomplete information. $K_a\varphi$ reads as "agent a knows that φ".

In [58], the uncertainty of players about the strategy is considered and they show the subtle distinction between an agent that knows that it has a suitable strategy and knows the strategy. The logic is called alternating-time temporal observational logic, and it is suitable for agents with bounded recall of the past.

Song, Goeckel, and Towsley [99] study incomplete information games at which agents have access to only their system state, known as information. It has been done by including an explicit description of the memory of the agents to the model.

However, none of previously mentioned logics cover Bayesian games, where agents do not know which game is about to be played.

2.2.2.2 Concurrent Dynamic Games Logic

In [111], concurrent dynamic game logic is introduced. The games of concurrent dynamic games, are two player normal form games. The language consists of two sorts, propositions and games. Given a set of atomic games Γ_0 and a set of atomic propositions Φ_0, game γ and proposition φ can have the following syntactic forms, yielding the set concurrent game logic games Γ and the set of concurrent game logic propositions Φ:

$$\gamma := g \mid \varphi? \mid \gamma_1 \,;\, \gamma_2 \mid \gamma_1 \cup \gamma_2 \mid \gamma_1 \times \gamma_2$$

$$\varphi := \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi \mid \langle\, \gamma \,,\, i \,\rangle\, \varphi$$

where $p \in \Phi_0$, $g \in \Gamma_0$ and i is the number of games. The formula $\langle\, \gamma \,,\, i \,\rangle\, \varphi$ expresses that player 1 has a strategy in ith game of the set of games γ and this formula ensures that the game ends in a state satisfying φ. The test game $\varphi?$ checks whether proposition φ holds at that position.

We avoid going through all the details, but informally γ states the relation between normal form games in concurrent dynamic games. For example, $\gamma_1 \cup \gamma_2$ says that the first player chooses which of the two normal form games to continue to play, and $\gamma_1 \,;\, \gamma_2$ states the sequential composition of two normal form games consists of first playing γ_1 and then γ_2. The important notation introduced in that work is $\gamma_1 \times \gamma_2$ which means the normal form games $\gamma_1$and γ_2 are played in parallel.

The semantics for this language follows the neighbourhood models or minimal models which are used in the semantics of non-normal modal logics.

As it is mentioned before this formalisation is not suitable for presenting Bayesian games as, in Bayesian games, the private knowledge of players should be considered.

2.2.2.3 Probabilistic Dynamic Epistemic Logic

In [61], a logic is proposed that combines dynamic epistemic logic, mentioned in section 2.2.1.1, and probabilistic epistemic logic. Probabilistic epistemic logic is proposed in [39] and semantically in the possible world model at each state, inferring each agent has a probability on the worlds that the agent considers possible. In [48], Heifetz and Mongin propose a probability logic for type spaces, which is very similar to probabilistic dynamic epistemic logic without dynamic capability. The language of probabilistic dynamic epistemic logic is given by the following formation rules:

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \ \vee \ \varphi_2 \mid K_a\varphi \mid [\varphi_1]\varphi_2 \mid q_1\mathbf{P}_a(\varphi_1) + ... + q_n\mathbf{P}_a(\varphi_n) \geq q$$

where p belongs to a countable set of propositions, a belongs to a finite set of agents and $q_1, ..q_n$ and q are rationales. A formula of the form $\mathbf{P}_a(\varphi) \geq q$ can be read as "the probability a assigned to φ is greater than or equal to b". A probabilistic epistemic model is $M = (W, R, V, P)$ such that:

- $W \neq \emptyset$ is a set of possible worlds.
- $R : \mathcal{A} \to 2^{W \times W}$ assigns an accessibility relation to each agent.
- $V : \mathcal{P} \to 2^W$ assigns a set of worlds to each proposition.
- $P : (\mathcal{A} \times W) \to (W \rightharpoonup [0, 1])$ with the following condition
  $\forall a \in \mathcal{A} \ \forall \omega \in W \quad \sum_{\upsilon \in dom(P(a,\omega))} P(a, \omega)(\upsilon) = 1$
  assigns a probability function to each agent at each world, such that its domain in a non-empty subset of the set of possible worlds.($\rightharpoonup$ means that it is a partial function; some worlds may not be in the domain of the function.)

where $\mathcal{P}$ is a countable set of propositions and $\mathcal{A}$ is a finite set of agents.

Truth definition for formula $\sum_{i=1}^{n} q_i\mathbf{P}_a(\varphi_i) \geq q$ in model M at states ω is as follows:

$$(M, \omega) \models \textstyle\sum_{i=1}^{n} q_i\mathbf{P}_a(\varphi_i) \geq q \text{ if and only if } \sum_{i=1}^{n} q_i P(a, \omega)(\varphi_i) \geq q$$

where $P(a, \omega)(\varphi_i) = P(a, \omega)(\{\upsilon \in dom(P(a, \omega)) | (M, \upsilon) \models \varphi_i\})$

In this thesis, we apply the same approach to represent the probabilistic belief. However, our approach has a slightly different semantic approach.

In [2] based on dynamic epistemic logic, public announcement logic is proposed by which we can describe actions in the form of public, truthful announcements. This logic makes epistemic logic dynamic by adding operator $\langle\psi\rangle$, where ψ is a formula. Formula $\langle\psi\rangle\,\varphi$ says after ψ is truthfully and publicly announced, φ will be true. In this work, Bayesian games are modelled as public announcement games by changing type's of each player to a signal that may be observed by player i. The method for finding Nash equilibrium strategies is by eliminating dominated strategies. In that work, axioms for games and rationality were not studied.

2.2.2.4 Set-Theoretic Beliefs

In [27], Chen and Micali formalise Bayesian games by modelling agents' belief about their types, belief about other's beliefs, etc. A possibilistic Kripke structure was used. In this structure beliefs are possibilistic (i.e. represented as sets) as opposed to being probabilistic. They refer to such a structure as type frame. If n is a set of player and $\Theta = \Theta_1 \times ... \times \Theta_n$ is a set of type tuples for each player (Θ_i), a type frame $\mathcal{V}$ for (n, Θ) is $\mathcal{V} = (\Omega, v, P_1, ..., P_n)$ such that :

- Ω is a finite set of states.
- v associates with each state $\omega \in \Omega$ a tuple of types $\vec{\nu} \in \Theta$.
- P_i for each player $i \in n$ associates with each each state $\omega \in \Omega$ a subset of Ω under the following conditions:
  1. $P_i(\omega) \subseteq [\![val_i(v_i(\omega))]\!]_V$, where $\forall\nu \in \Theta_i, [\![val_i(\nu)]\!]_V = \{\omega' : v_i(\omega') = \nu\}$ and $v_i(\omega)$ denotes player i's type in the type tuple $v(\omega)$.
  2. $P_i(\omega) \subseteq [\![belief_i(P_i(\omega))]\!]_V$ where for each subset $\pi \in \Omega$, $[\![belief_i(\pi)]\!]_V = \{\omega : P_i(\omega) = \pi\}$

The above conditions say that player i knows its own type and its own belief, which means in every state of the world the player considers possible, the state has the same type and beliefs.

This frame is extended for n-player games by adding an extra function and condition. A game structure M is a tuple $\{\Omega, s, v, P_1, ..., P_n\}$ where s associates with each state $\omega \in \Omega$ a pure strategy $s(\omega)$. Besides the condition mentioned above for $P_i(\omega)$, an additional condition is defined as:

- $P_i(\omega) \subseteq [\![play_i(s_i(\omega))]\!]_M$ where for each strategy σ_i for player i, $[\![play_i(\sigma_i)]\!]_M = \{\omega' : s_i(\omega') = \sigma_i\}$ and $s_i(\omega')$ denotes player i's strategy in the strategy tuple $s(\omega')$.

They also model a very weak notion of rationality, which is σ_i is a rational choice of strategy for player i, if for every alternative σ_i' for i, some state of the world exists that i considered possible, such that playing σ_i would perform at least as well as σ_i'. Thus, no alternative strategy σ_i' performs better than σ_i in every situation that player i considers possible. Using these definitions, they define the semantics of the weak rationality operator RAT_i as follows:

- $(M, \omega) \models RAT_i$ if and only if for every strategy σ_i' for player i some $\omega' \in P_i(\omega)$ exists such that

$$u_i(v(\omega'), (s_i(\omega), s_{-i}(\omega'))) \geq u_i(v(\omega'), (\sigma_i', s_{-i}(\omega')))$$

where u_i is player i's utility and $s_{-i}(\omega') = \{s_1(\omega'), ..., s_{i-1}(\omega'), s_{i+1}(\omega'), ..., s_n(\omega')\}$ is the strategy tuple of other players excluding the strategy of player i.

This work is different from the work in thesis as we consider probabilistic beliefs. Furthermore, the axioms for playing Bayesian games are not provided.

The concept of rationality leads us to detect solutions in the games. In [10] dynamic rationality is formalised by conditional doxastic logic which can model plausibility situation that are unavoidable in the belief revision structures. However, the rationality is studied in a backward induction procedure to find the solution of games. Furthermore, backward induction procedures can only be applied on extensive form games, as backward induction is the process of reasoning backwards in time. Thus, it starts from the end of a problem or situation, to determine a sequence of optimal actions. Therefore, it is not suitable for analysing simultaneous games.

In this section, we provided a review of the background and related works. In the next section, we briefly study suitable methods that can be used based on semantics of formal languages to verify properties in games.

2.3 Model Checking

In the domain of multi-agent systems and game theory, it is natural to reason about both, actions and states. Furthermore, a number of modalities such as epistemic or deontic are also formalised in terms of relations over states of a system or model. In this setting, it is worthy to have analysis techniques and tools, in which information can be assigned to both, states and transitions of the model. In addition, systems that have more than one relation over states can be considered within the same model. Model checking, which has been first proposed by Clarke and Emerson [29], is an approach to automated analysis of finite state concurrent systems.

Generally, in a model checking technique, specifications are formulated in a modal logic and the system is modelled as a state transition graph, which is the same as a Kripke structure for the applied logic. The procedure continues by checking whether the system satisfies its specifications given by a logical formula or not. Therefore, model checking techniques reduce verification to testing whether the Kripke structure is the model of the formula.

Temporal modal logic is the dominant logic used in model checking as it provides convenient formalisms for reasoning about distributed systems. Among temporal logic, computational tree logic receives many attentions in traditional model checking tools, as it can be used for reasoning about branching time [90], [84], [24]. However, different model checking techniques have been proposed for other logics, such as hybrid logic [41], modal μ-calculus [22], alternating-time temporal logic [94], [1] and epistemic logic.

Naturally, different variations of epistemic logics are suggested for the model checking technique. In [21], Boureanu, Jones and Lomuscio used temporal epistemic logic proposed by [79] and extended it by a modal operator for rewriting-knowledge modality, which combines equational theories with epistemic logic. Alternating-time temporal logic is extended with epistemic modality, which is called alternating temporal epistemic logic [51] and it

is suitable for checking proprieties of planning in multi-agent systems. In another work [119], alternating temporal epistemic logic is interpreted for alternating-time temporal epistemic transition systems.

Model checking, a combined logic of knowledge and linear time in synchronous systems with perfect recall is studied in [72]. The studied language is a propositional multi-modal language, based on a set of propositional constants with formulae generated by modalities $\bigcirc$ (next), $\mathcal{U}$ (until), and knowledge operator K_i for each agent $i \in \{1, ..., n\}$ and a common knowledge operator C_G for each group of agents $G \subseteq \{1, ..., n\}$. A model M of the form $\langle S, I, T, O, \pi, \alpha \rangle$ is assumed such that: S is a finite set of states, I is a subset of S, representing the possible initial states, $T \subseteq S^2$ is a transition relation, O is a tuple $(O_1, ..., O_n)$ of functions, where for each $i \in \{1, ..., n\}$ the component $O_i : S \to O$ is called the observation of agent i, $\pi : S \to \{0, 1\}^{propositions}$ is an interpretation for each proposition at each state and $\alpha \subseteq S$ is an acceptance condition. To decide the relation $M, \omega \models K_i\varphi$, where ω is a state and φ a formula, they factorise formulae into their temporal and knowledge components. Then each temporal component is mapped to a knowledge component, which means simultaneous substitution for each occurrence in φ of a temporal component, that the formula $K_i\psi$ such that ψ is true in that particular mapping. Although this technique works for applications in which knowledge changes over time, it does not cover probabilistic beliefs.

In [52] the temporal epistemic language for model checking is extended to express probability. As probabilistic interpreted systems are infinite structures, they are not suitable as input for a model checking algorithm. Therefore, they worked with a type of finite model called interpreted partially observed discrete-time Markov chain. By this model, we can express the probability of a transition.

Model checking for most of the modal logics usually has a trivial NP-hard lower bound, because these logics contain propositional logic. For propositional logic, model checking or satisfiability is the defining NP-complete problem. For branching time logics it is not clear whether it contains propositional logic, because it does not have primitive propositions [1]. Ågotnes, van der Hoek and Wooldridge [1] proved that the satisfiability problem for cooperative game logic is NP-complete, even for cooperative game logic formulae with one agent.

In [72], it was proven that the problem of determining a formula φ of the combined logic of knowledge and linear time logic in model M is decidable in polynomial space with following complexity:

$$|\varphi|.exp(depth(\varphi), O(|M|)).$$

where $depth(\varphi)$ is the deepest nesting of modal operators (here $K_i, \bigcirc$ and $\mathcal{U}$). Formulas without any modal operators have a modal depth of zero.

In the area of model checking not only many techniques are developed, but also many tools are designed and implemented. In some cases a tool or method is devised to translate the desired language to the input language of already implemented systems. An example is the game description language in [95], which is a special purpose declarative language. In [94], it was shown that the game description language and alternating-time temporal logic are intimately related at the semantic level. A link between these two languages was built as a translator of game description language to alternating-time temporal logic for model checking. Therefore, game-theoretical situations can be verified by alternating-time temporal logic and consequently with any model checker tool that supports this logic. One of the developed model checkers that support this logic is Mocha [74].

Mocha is an interactive model checker for system specification and verification. A model is specified in the language of reactive modules. This language allows the formal specification of systems with synchronous, asynchronous, and real-time components. It accepts specifications in alternating temporal logic and also computational tree logic. This tool verifies by checking trace containment between implementation and specification modules. Although Mocha can be adapted to model games, it does not support epistemic operators of epistemic logic.

PRISM [87] is a model checker for formal modelling and analysis of systems with random or probabilistic behaviour [65]. PRISM is a Prolog based statistical modelling language benefits from various learning methods other that MLE (maximum likelihood estimation) such as MAP (maximum a posterior) [55]. However, PRISM has a problem and it still does not support real numbers as a type, therefore, we cannot use this tool for modelling Bayesian games.

Another model checker is Spin [103]. Spin is a model checker to verify the correctness of distributed software models in a rigorous and automated fashion. Systems that can be verified are modelled by Promela (Process Meta Language), which supports modelling of asynchronous distributed algorithms. Properties that can be verified in these models are expressed as linear temporal logic formulas. Nevertheless, knowledge operator has not been implemented in Spin.

The model checker for the logic of knowledge is MCK [70]. The system is suitable as a testbed for a variety of approaches to model checking epistemic logic. This model checker supports several different ways of defining knowledge, given a description of a multi-agent system as the model with possible observations made by the agents. The observation can be done in different modes such as observation only, observation and time, and synchronous and asynchronous perfect recall of all observations. Both, linear and branching time temporal operators are supported. Even though it is extended to support probabilistic epistemic logic, it still does not support epistemic beliefs over games.

2.4 Remarks

In this chapter, we have reviewed game-theoretical scenarios in the multi-agent system context. We reviewed examples of how to model and find solutions in normal form games and Bayesian games. Different presentations exist and we have chosen those that are straightforward for reasoning with epistemic logic.

Based on the characteristic of desired games, modal logics offer the best approach to study reasoning about games. Different modal logics have been reviewed for both game categories and hardly any of them satisfied all the required expressiveness. Therefore, it is required to have a formal language in which inferences can be represented with a clear distinction between syntax and semantics, various notations of validity, and decision procedures.

We have also reviewed different approaches to automatically verify systems by formal languages. As a proof that a formal language can satisfy a desired property, it is a good practice to use it as a model checking technique. A variety of techniques and tools have already been developed, but each of them

alone cannot satisfy the characteristics of our formal language. Therefore, we decide to develop our own system as a proof of concept, which will be capable of addressing all characteristics of our formal language.

Based on the mentioned reasons, in Chapter 3 we introduce epistemic logics for normal form games and Bayesian games, for qualitatively describing them and reasoning about them.

Chapter 3

Games and Epistemic Logic

The first step in devising a formalism for reasoning about games is to decide what general properties of games we want that formalism to capture. In this thesis, we want to reason about rational players in two kinds of games, normal form games and Bayesian games. A rational player has knowledge about some aspects of games, and it also has clear preferences over a set of feasible strategies. Furthermore, it is able to discover an optimal strategy that maximises payoffs. Rationality implies that not only the chosen strategy is the best possible given player's knowledge, but also is derived from coherent inferences. The classical formalism for reasoning about rationality in games is epistemic logic.

In this chapter, we present game models for normal form games and Bayesian games. We use epistemic logic to represent these games. Moreover, we extend the epistemic logic to model Bayesian games. In addition, we reason about the rationality of players of these two game models.

3.1 Epistemic Logic for Normal Form Games

Game theory as a theory of practical reasoning [114] is a matter of detecting the behaviour or actions that have the best expected outcome given one's preferences. Consequently it tries to facilitate decision making in multi-agent systems [100], [96], by means of predicting or explaining the behaviour of agents under a sequence of interactions.

Two factors that define the foundation of an agent's rationality are the dependency of the agent on its reasoning ability and the information about other agents. Furthermore, the epistemic program in game theory [32], [18]

and [8] demonstrates the power of understanding rational behaviour through mutual expectation in game-theoretic interactions [92]. In a game, an agent may behave in a particular way if the agent knows that another agent is rational, yet behave differently if the agent is not aware of other agents rationality. As a result, knowledge and belief can be interpreted as equally important in game scenarios. For instance, some game theorists make the point that an agent's information should be included in game models [92] and epistemic logic can be used for the development of these models. Epistemic logic as an analytical toolbox that combines both philosophical logic and theoretical computer science themes, is used to analyse reasoning about information and the update of information. Epistemic logic lets players consider their own reasoning abilities. It is the logic of knowledge, and allows reasoning about the knowledge of agents in a group. Agents may have different information and thus different epistemic alternatives at each possible state. We start from the specific actions for specific agents and reason about what those agents can achieve.

Before we present the epistemic language, we review the definition of games that are studied in this thesis. Here, we focus on normal form games which are also known as the strategic or matrix form with imperfect information. Normal form games are important since Bayesian games or incomplete information games are formed by a set of normal form games. The following is the definition of normal form games.

The normal games model strategic interactions in which at least two or more rational decision makers determine the outcome of a decision situation. The two general branches of game theory are non-cooperative and cooperative games; the normal games are the standard model of non-cooperative game theory [112]. In a normal game, every player selects an action/strategy from a set of possible strategies and taken together a combination of choices of all players who determine the set of outcomes. At this stage the model is called a game form, which only deals with a combination of a set of strategies in each state but players are neutral about the different states. Adding preferences as utility functions over the set of outcomes, transfers a game form to a normal game. As an implicit principle, players become rational, i.e., their decisions involve the maximisation of the expected utility. As players are uncertain about other players preferences, a player might chose a strategy but this

depends on the actions of others. Therefore, the expected utility would not be the one to be used to maximise the outcome. These situations might happen because players play simultaneously and have imperfect information about the game which means players can not observe the selections of their opponents at the time of making decisions. To overcome this difficulty, game theory has formulated a number of solution concepts which specify a strategy or a set of strategies for each player as a solution for games such as iterated elimination of dominated strategies or the Nash equilibrium.

These games are represented by a tuple $(N, \{S_i\}_{i \in N}, u)$, where:

- N is a finite set of n players, indexed by i.
- $S = S_1 \times .... \times S_n$ where S_i is a finite set of strategies of player $i \in N$.
- $u = (u_1, ..., u_n)$ where $u_i : S_i \rightarrow \mathbb{R}$ is a real-valued utility (payoff/outcome) function for player i.

The last item u can be replaced with $\succeq$ which is the preference relation and is the equivalent definition for the utility function. A common way to present these games is via an n-dimensional matrix.

		player 2	
		cooperate	defect
player 1	cooperate	(a,a)	(b,c)
	defect	(c,b)	(d,d)

Table 3.1: Prisoner's dilemma

The game shown in table 3.1 is a common example of normal games, called the prisoner's dilemma. The numbers a, b, c, d are interpreted as a measure of an agent's level of happiness. They are called utility values and we assume in this example $c > a > d > b$. There are two players who are presumably suspected of a crime, where each of them has two options, which are either to cooperate or to defect. In each cell of table 3.1, the first letter represents the player 1's payoff and the second letter represents player 2's payoff. If the payoffs are all positive or all negative, their absolute values can represent the length of the jail term. If one player cooperates and the other defects, the player will lose by b and the other one will gain c. If both of them cooperate, they will gain a. The last scenario, to choose d, is the

one that will be adopted by any rational player based on game theory. A rational player should choose to defect because it looks at the game from its own point of view regardless of what the other players will do. This will lead to the Nash equilibrium which is to defect for both players.

3.1.1 Syntax for Normal Form Games

Game theory explains actions in terms of the reasons agents have to carry them out. Normal form games provide the basic structure of simultaneous games. These games are the basis of Bayesian games. Study of these games is useful in order to distinguish between possible actions a player can choose to perform, its preferred ordering between those actions, the rationality principle, the action eventually performed and finally the beliefs about all the four ingredients. B. de Burin [32] introduced an epistemic language for representing the solution concept in normal games and extensive games. He proposed a formula that expresses the solution concept such as the Nash equilibrium. This formula simply says that, if a player expects its opponent to be rational, and knows that its opponent knows the utility structure, these beliefs form a pure strategy Nash equilibrium. Thus,we first introduce the language for representing normal form games.

This epistemic logic is a multi-modal logic whose syntax is formed in the usual way [14], [26] by a countable set A of atomic propositions. The Boolean combinators (connectives) used are $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (equivalence). The conjunction (disjunction) of all sentences from a finite set Σ is abbreviated by $\bigwedge \Sigma$ ($\bigvee \Sigma$), assuming commutativity. If the φ_i enumerate Σ, it is written as $\bigwedge_i \varphi_i$ ($\bigvee_i \varphi_i$). Also, the logic has modal (knowledge) operators K_i for each player i. The basic atomic propositions for games are:

- The proposition letter $\mathbf{i}_m$ $(i = 1, ..., N)$ stands for the statement 'i plays its mth strategy i_m'. This notation has two important items of information, first about the player and the second about the strategy/action the player chooses to play.

- The proposition $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}}$ denotes that the utility for player i, when the strategy profile $(1_{k_1}, ..., N_{k_N})$ is played, equals the

number r. This notation shows the other players strategies, because the utility of a player depends on the selected strategies of others. For example, player 1 plays one of its strategies, which here is presented as k_1, and in the same manner player N plays k_N from its set of strategies. Consequently player i plays k_i.

- R is a countable set of symbols such as r. The elements of R represent real numbers.(Note that R is not the set of real numbers.)

- The proposition $\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N}}$ states that for player i its mth strategy is at least as good as his nth strategy while other players choose $\{1_{k_1}, ..., i-1_{k_{i-1}}, i+1_{k_{i+1}}, ..., N_{k_N}\}$ to play. Binary relation $\succeq$ satisfies the following properties: for all $r, r', r'' \in R$

  1. either $r \succeq_i r'$ or $r' \succeq_i r$ (completeness or connectedness),
  2. if $r \succeq_i r'$ and $r' \succeq_i r''$ then $r \succeq_i r''$(transitivity).

  That is, it is a total preorder (also called a preference relation).

- The proposition $\mathbf{rational}_i$ denotes the rationality of player i, in the sense that i is an expected utility maximizer.

A Hilbert-style proof system can be used to check the validity of a formula. However, the following axioms are needed for the knowledge modality:

1. Any axiomatisation for propositional logic (classical)
2. $(K_i\varphi \wedge K_i(\varphi \to \psi)) \to K_i\psi$ (K, knowledge or distribution property)
3. $K_i\varphi \to \neg K_i \neg \varphi$ (D or consistency)
4. $K_i\varphi \to K_iK_i\varphi$ (4 or positive introspective)
5. $\neg K_i\varphi \to K_i\neg K_i\varphi$ (5 or negative introspective)
6. $K_i\varphi \to \varphi$ (T or Knowledge of truth)

The inference rules are:

- If $\vdash \varphi \to \psi$ and $\vdash \varphi$ then $\vdash \psi$ (Modus Ponens)
- If $\vdash \varphi$ then $K_i\varphi$ (Necessitation or Knowledge Generalisation)

In addition, there are specific axioms for playing normal form games.

$$\bigwedge_i \bigvee_m \mathbf{i}_m \tag{$G1$}$$

$$\bigwedge_i \bigwedge_m \neg(\mathbf{i}_m \wedge \mathbf{i}_n) \tag{$G2$}$$

$$\bigwedge_i \bigwedge_m (K_i \mathbf{i}_m \leftrightarrow \mathbf{i}_m) \tag{$G3$}$$

$$\begin{aligned} &\mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}} \rightarrow \\ &K_i \mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}} \end{aligned} \tag{$G4$}$$

$$\begin{aligned} &(\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N}}) \vee \\ &(\mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}}) \end{aligned} \tag{$G5$}$$

Axiom G1 states that each player plays at least one strategy (where m ranges over the strategies available to player i). Axiom G2 says that a player must not choose more than one strategy. Consequently, these two axioms together imply that a player only chooses one strategy. Axiom G3 states that each player knows its selected strategy and consequently plays it. Furthermore, a player has information about that strategy when played. Axiom G4 is an immediate consequence of the necessitation rule and says a player knows its own payoff. The last, axiom G5, defines that payoffs are comparable under binary relation $\succeq$.

To validate these axioms and to connect the logic to game situations, a connector is needed, which is a semantics.

3.1.2 Semantics for Normal Form Games

Epistemic logic is a branch of modal logic which can support a different number of modal operators. The logic for normal games is based on a multi-modal logic with n operators $K_1, K_2, ..., K_n$, where $i = 1, ..., n$, $K_i\varphi$ means that player i knows that φ. A semantics is required to establish conditions for

the statement at which the logic is true or satisfied for a normal form game structure. Due to the knowledge operator **K** and the need for supporting other axioms such as **T**, **4** and **5**, the modal system **S5** is chosen as the epistemic semantic system. The common method to encode the agent's information starts with possible-worlds structures, also called Kripke structures. Kripke structures are those structures $\langle \Omega, \mathcal{K}_1, ..., \mathcal{K}_n \rangle$ in which Ω is a set of states or possible worlds and for every $i \in \{1, ..., n\}$, $\mathcal{K}_i$ is a binary relation on Ω. Because of axiom **T** the binary relation in the Kripke structure needs to be reflexive (a binary relation R over domain X is reflexive if and only if $\forall x \in X, xRx$).

The truth value of a non-modal sentence is determined at a possible world by the following satisfaction relation:

- $w \models p$ if p is true in $w \in \Omega$, for any atomic proposition p;
- $w \models \neg\varphi$ if and only if it is not the case that $w \models \varphi$;
- $w \models \varphi \vee \psi$ if $w \models \varphi$ or $w \models \psi$;
- $w \models \varphi \wedge \psi$ if $w \models \varphi$ and $w \models \psi$;
- $w \models K_i\varphi$ if for all $w' \in \mathcal{K}_i(w)$, we have $w' \models \varphi$;

where for every $w \in \Omega$ and for every $i \in \{1, ..., n\}$, $\mathcal{K}_i(w)$ is defined as :

$$\mathcal{K}_i(w) = \{w' \in \Omega : w\mathcal{K}_i w'\}$$

Given a normal form game $G = (N, \{S_i\}_{i \in N}, u)$ and the Kripke structure **S5** $\langle \Omega, \{\mathcal{K}_i\}_{i \in N} \rangle$, a structure for the game G is formed by adding n functions to F. These functions are $\sigma_i : \Omega \longrightarrow S_i (i \in N)$ satisfying that if $w' \in \mathcal{K}_i(w)$ then $\sigma_i(w') = \sigma_i(w)$. These n functions form at each state w a strategy profile $\sigma(w) = (\sigma_1(w), ..., \sigma_n(w))$ which is a combination of all players' strategies at each state. If we need to refer to the combination of all players' strategies excluding player i, we use the notation $\sigma_{-i}(w) = (\sigma_1(w), ..., \sigma_{i-1}(w), \sigma_{i+1}(w), ..., \sigma_n(w))$. G-structure is defined as $\langle \Omega, \{\mathcal{K}_i\}_{i \in N}, \{\sigma_i\}_{i \in N} \rangle$. A semantic model of G or G-model is obtained by adding the following valuation to the G-structure:

- $w \models \mathbf{i}_m$ if and only if $\sigma_i(w) = \mathbf{i}_m$,

It says that $\mathbf{i}_m$ is true in the G-model at state w when player i selects strategy $\mathbf{i}_m$ if and only if the strategy of player i related to state w by function σ is $\mathbf{i}_m$.

- $w \models \mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}}$ if and only if $\sigma_{-i}(w)$ exists such that $u(\sigma_i(w), \sigma_{-i}(w)) = r$,

It states that $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}}$ is true in the G-model at state w, the payoff of player i is $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}}$ if and only if the combination of players's strategies excluding player i related to state w by $\sigma_{-i}(w)$. In addition, player i's payoff at state w is $u(\sigma_i(w), \sigma_{-i}(w)) = r$.

- $w \models \mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N}}$ if and only if $u(\mathbf{i}_m, \sigma_{-i}(w)) \succeq u(\mathbf{i}_n, \sigma_{-i}(w))$.

It says that $\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N}}$ is true in the G-model at state w, when strategy $\mathbf{i}_m$ is at least as good as strategy $\mathbf{i}_n$ for player i if and only if the combination of $\mathbf{i}_m$ and $\sigma_{-i}(w)$ results at a payoff at which player i gains at least as good as the payoff of the combination of $\mathbf{i}_n$ and $\sigma_{-i}(w)$.

For reasoning about games, the rationality of players should be represented. Moreover, the definition of rationality as the player's characteristic results in reasoning about games. There are three criteria for rationality [96]:

- knowledge of the problem which says the player has a clear picture of the choice problem,
- clear preference, which means the player has an ordering over the entire set of strategies, and
- ability to optimise which captures the ability to discover the optimal course of action.

The proposition that captures the rationality of player i is called $\mathbf{rational}_i$. The axiom RATIONAL which is the formalism of utility maximisation captures $\mathbf{rational}_i$ as follows:

$$\textbf{RATIONAL} : \textbf{rational}_i \leftrightarrow (K_i \bigwedge_{i_{k_i}} \mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}} \wedge \mathbf{i}_m) \wedge \bigwedge_{i_{k_i}} (\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}})$$

The above axiom states that player i aims at its utility maximiser, if the player decides to play its mth strategy ($\mathbf{i}_m$) in a situation in which the player has information about the utility (captured by the $(K_i \bigwedge_{i_{k_1}} \mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}})$) then the mth strategy is better than any other, given its beliefs. This axiomatisation of rationality of player i also means that the strategy i_m is the Nash equilibrium for player i.

In game theory Nash equilibrium is the best response of a player after knowing other players' strategies [100]. The best strategy contains the concept of preference or ordering and also ability to optimise. Knowing other players' strategies is equal to having information about the problem. Therefore, a rational player chooses a strategy that is a Nash equilibrium.

However, to verify the axiom RATIONAL in a game structure, a criterion is needed. Thus, the satisfaction relation for G-model satisfies the following extra condition:

- $w \models \textbf{rational}_i$ if and only if, for every $s_i \in S_i$ there exists an $w' \in \mathcal{K}_i(w)$ such that $u(\mathbf{i}_m, \sigma_{-i}(w')) \succeq u(s_i, \sigma_{-i}(w'))$.

For normal form games, a logic with syntax and semantic is proposed. We now show the soundness of the logic. A system of modal logic Γ is sound with respect to a class of models C, when every theorem (axiom) of Γ is valid in C [26].

Proposition 1: The proposed logic is sound with respect to the class of G-models which is **S5** plus game axioms (**G1, ...,G5**).
Proof

- Axioms G1 and G2 imply that a player chooses only one strategy at each state. G1 and G2 are valid in every model because for every state w there is a unique strategy $i_m \in S_i$ such that $\sigma_i(w) = i_m$ by the validation rule $w \models \mathbf{i}_m$ if and only if $\sigma_i(w) = \mathbf{i}_m$.

- Axiom G3 (left to right) is an immediate consequence of the definition of the binary accessibility relation for modal K_i operator that if $w' \in \mathcal{K}_i(w)$ then $\sigma_i(w') = \sigma_i(w)$ and axiom G3 (right to left) is valid because the relation in this structure is reflexive (axiom **T**) so $w \in \mathcal{K}_i(w)$.

- Axiom G4 is valid because if $w' \in \mathcal{K}_i(w)$ then $\sigma_i(w') = \sigma_i(w)$ and $u(\sigma_i(w), \sigma_{-i}(w)) = u(\sigma_i(w'), \sigma_{-i}(w))$.

- Axiom G5 is valid because for every state w there is a unique profile strategy $\sigma_{-i}(w)$ of the players other than i and the ordering of $u_i(1_{k_1}, ..., N_{k_N})$ induces an ordering of $\bigwedge_m i_m$.

- Axiom $\mathbf{rational}_i$ is valid. Let's suppose that

$$w \models (K_i \bigwedge_{i_{k_i}} \mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}} \wedge \mathbf{i}_m)$$
$$\wedge \bigwedge_{i_{k_i}} (\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}})$$

  - $K_i \bigwedge_{i_{k_1}} \mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}}$ means that for every w' that $w' \in \mathcal{K}_i(w)$ based on the modal K_i operator definition $\sigma_i(w') = \sigma_i(w)$. Furthermore, for each state w there is a unique $\sigma_{-i}(w)$ and $(\sigma_i(w'), \sigma_{-i}(w)) = r$,
  - $\mathbf{i}_m$ means $\sigma_i(w) = \mathbf{i}_m$,
  - $\bigwedge_{i_{k_i}} (\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N}})$ says for all strategies of player i $u(\mathbf{i}_m, \sigma_{-i}(w)) \succeq u(\sigma_i(w), \sigma_{-i}(w))$ is true,

  Therefore, every $i_m \in S_i$ there exists a $w' \in \mathcal{K}_i(w)$ such that $u(\mathbf{i}_m, \sigma_{-i}(w'))$ $\succeq u(\sigma_i(w'), \sigma_{-i}(w'))$.

■

Since the proposed logic is sound, a rational player can reason about the best strategy at each state in a normal form game by using this logic. As an example of reasoning by this logic, we revisit the Prisoner's dilemma from

table 3.1 as a normal form game. In the game, the following set of strategies will be used:

$$S = \{(cooperate, cooperate), (cooperate, defect), \\ (defect, cooperate), (defect, defect)\}$$

The game has two players, $i \in 1, 2$, player $i = 1$ is considered as the row player and consequently player $i = 2$ is the column player.

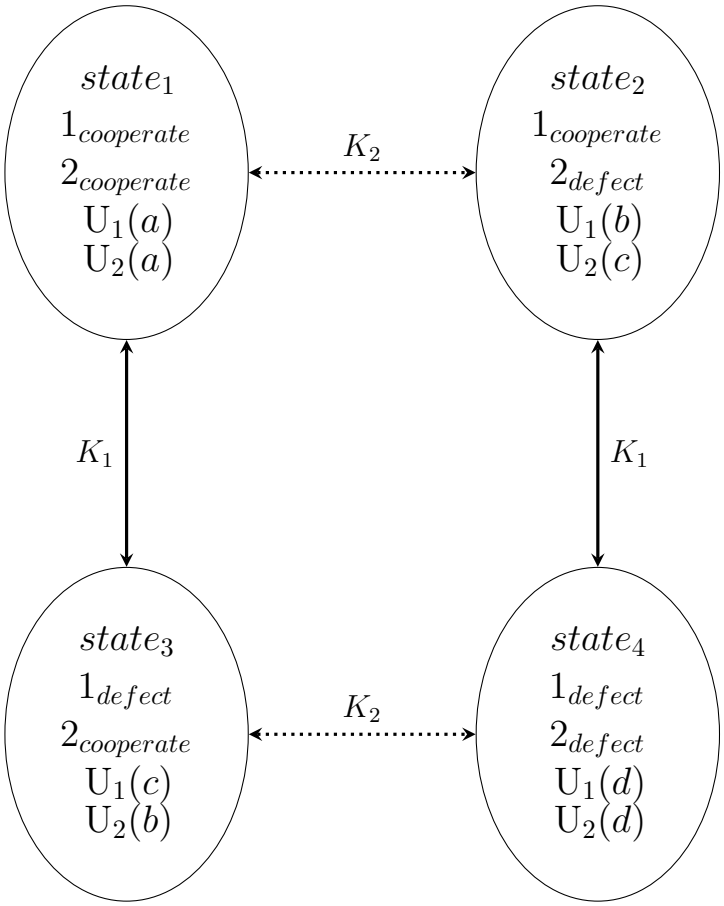

Figure 3.1: The four states of Prisoner's dilemma (table 3.1)

In figure 3.1, all the four states of the game from table 3.1 are shown. Player 1 knows player 2's strategy in states 1 and 3 and player 2 knows player 1's strategy in states 2 and 4. As it is shown in figure 3.1, player 2 has knowledge about player 1's strategy because in states 1 and 2, player 1's

strategies are the same (cooperation) $\sigma_2(state_1) = \sigma_2(state_2)$. Consequently, we have that $state_2 \in \mathcal{K}_2(state_1)$ holds in state 1. Figure 3.1 presents that player 1 has knowledge about player 2's strategy as in states 1 and 3, player 2's strategies are identical (cooperation) $\sigma_1(state_1) = \sigma_1(state_3)$. Thus, in state 1 we have that $state_3 \in \mathcal{K}_1(state_1)$. Based on this argument, state 1 implies the following valuation.

$$\begin{aligned} state_1 \models \mathbf{1}_{cooperate} \wedge \mathbf{2}_{cooperate} \wedge \\ (\mathbf{u}_2(1_{cooperate}, 2_{defect}) \succeq \mathbf{u}_2(1_{cooperate}, 2_{cooperate})) \wedge \\ (\mathbf{u}_1(1_{defect}, 2_{cooperate}) \succeq \mathbf{u}_1(1_{cooperate}, 2_{cooperate})) \end{aligned}$$

Therefore, both players are irrational at state 1 and this state is not a Nash equilibrium.

$$state_1 \models (\neg \mathbf{rational}_1 \wedge \neg \mathbf{rational}_2)$$

At state 2 the following valuation is true.

$$\begin{aligned} state_2 \models \mathbf{1}_{cooperate} \wedge \mathbf{2}_{defect} \wedge \\ (\mathbf{u}_1(1_{defect}, 2_{defect}) \succeq \mathbf{u}_1(1_{cooperate}, 2_{defect})) \wedge \\ (\mathbf{u}_2(1_{cooperate}, 2_{defect}) \succeq \mathbf{u}_2(1_{cooperate}, 2_{cooperate})) \end{aligned}$$

In state 2 player 2 is rational and player 1 is irrational.

$$state_2 \models (\neg \mathbf{rational}_1 \wedge \mathbf{rational}_2)$$

State 3 implies the following valuation:

$$\begin{aligned} state_3 \models \mathbf{1}_{defect} \wedge \mathbf{2}_{cooperate} \wedge \\ (\mathbf{u}_2(1_{defect}, 2_{defect}) \succeq \mathbf{u}_2(1_{defect}, 2_{cooperate})) \wedge \\ (\mathbf{u}_1(1_{defect}, 2_{cooperate}) \succeq \mathbf{u}_1(1_{cooperate}, 2_{cooperate})) \end{aligned}$$

In state 3 player 1 is rational and player 2 is irrational.

$$state_3 \models (\mathbf{rational}_1 \wedge \neg \mathbf{rational}_2)$$

Finally, the following valuation is true in state 4:

$$\begin{aligned} state_4 \models\ & \mathbf{1}_{defect} \wedge \mathbf{2}_{defect} \wedge \\ & (\mathbf{u}_1(1_{defect}, 2_{defect}) \succeq \mathbf{u}_1(1_{cooperate}, 2_{defect})) \wedge \\ & (\mathbf{u}_2(1_{defect}, 2_{defect}) \succeq \mathbf{u}_2(1_{defect}, 2_{cooperate})) \end{aligned}$$

In this state both players are rational and this state is the Nash equilibrium of the game.

$$state_4 \models (\mathbf{rational}_1 \wedge \mathbf{rational}_2)$$

In conclusion, we have proposed a formal representation for normal form games and reasoned based on the proposed logic about a game. In the following section we apply this logic and extend it for Bayesian games.

3.2 Bayesian Games and Epistemic Logic

In many disciplines such as philosophy, economics and artificial intelligence, reasoning about knowledge is not only about agents but also it is about the probability of certain events. In particular, game theory encounters uncertainty about events in a variety of situations. One scenario is given a set of games with no information available about which game is about to be played. A possible approach to the study of the interaction between players in these is all games should be considered as probable games. These games are known as Bayesian games. For reasoning about these games, the rationality of players should be considered. It means, if the rationality of players is represented by a logical language, the reasoning about these games will be explicit. In the previous section, epistemic logic has been used to represent

normal form games. It dealt with knowledge of the players and it represented the rationality of the players. For representing Bayesian games, the logical language has to be powerful enough to reason about the knowledge of the players as well as uncertainty of each player about the set of games being played. In the next section a formal logic is introduced to represent and reason about Bayesian games.

3.3 Epistemic Logic for Bayesian Games

Interactive decision making situations are modelled with Bayesian games. In these situations decision makers possess only partial information about the games and the other players. This lack of knowledge or uncertainty about the situations is typical in real-life decision making and reveals the importance of these games. In normal form games, the common assumption is that information about these games are common knowledge among the players. This assumption is strong and makes the normal form games inflexible for many real-life situations. Bayesian games are important because in these games this assumption is relaxed. Therefore, Bayesian games model the situations in which the game that is being played may not be common knowledge among the players. Instead, some players may hold a different payoff table or pure strategy set to be true. In a Bayesian game a player's beliefs include the knowledge of the game description (payoffs, strategies), as well as the probability distribution over the beliefs other agents may have. The set of beliefs held by a player is known as its epistemic type or simply type. The uncertainty is defined directly over a game's utility function. Therefore, a Bayesian game is a tuple (N, A, Θ, p, u) where:

- N is a finite set of n players, indexed by i.
- $A = A_1 \times ... \times A_n$ where A_i is a finite set of actions available to player i.
- $\Theta = \Theta_1 \times ... \times \Theta_n$, where Θ_i is the type space of player i.
- $p : \Theta \longmapsto [0, 1]$ is a common prior over types.
- $u = (u_1, ..., u_n)$ where $u_i : A \times \Theta_i \to \mathbb{R}$ is a real-valued utility (or payoff) function for player i.

The assumption is that all the above is common knowledge among the players, and each agent knows his own type. An example of a Bayesian game is given in table 3.2.

	$\theta_{2,1}$			$\theta_{2,2}$		
$\theta_{1,1}$	MP			PD		
		L	R		L	R
	U	2,0	0,2	U	2,2	0,3
	D	0,2	2,0	D	3,0	1,1
		p=0.3			p=0.1	
$\theta_{1,2}$	Coor			Bos		
		L	R		L	R
	U	2,2	0,0	U	2,1	0,0
	D	0,0	1,1	D	0,0	1,2
		p=0.2			p=0.4	

Table 3.2: A Bayesian game (taken from [100])

As illustrated in table 3.2, a Bayesian game consists of different normal form games which have the same number of players and strategies. In this example, four normal form games are played simultaneously, MP (Matching Pennies), PD (Prisoner's Dilemma), Coor (Coordination) and Bos (Battle for Sexes). This example has two players, player 1 as the row player and player 2 as the column player. Both players have two actions, player 1's are U and D and player 2's L and R. These actions can be interpreted differently in each normal form game. As an attempt to simplify notations, we consider the case that all normal form games have the same actions. Therefore, U means player 1 moves up and D is player 1 moves down. Similarly, L is a left movement and R is a right movement for player 2. As a way of defining uncertainty over a game's utility function, each player has two types, $\theta_{1,1}$ and $\theta_{1,2}$ for player 1 and $\theta_{2,1}$ and $\theta_{2,2}$ for player 2. The type of a player includes the player's private knowledge about the games which are not common knowledge. The game is played in three steps:

1. a chance move chooses a particular game using the probability distribution p.

2. every player knows its type θ but not the game that will be played or

the other players' types.

3. the players simultaneously choose an action and receive a payoff.

In Bayesian games, the pure strategy of a player is a map from player's types or information to actions $s_i : \Theta_i \longrightarrow A_i$. The notation $s_i(a_i|\ \theta_i)$ is used to denote the probability under strategy s_i when player i chooses action a_i, given that i's type is θ_i. As Bayesian games have different sources of uncertainty, the notation of expected utility has three different notions: ex post, ex interim and ex ante. Ex post considers the actual player's type, ex interim is computed when the player knows its own type but does not have any information about the types of the other players. The last one, ex ante, is expected utility under the setting that the player knows nobody's actual type. The question is, which of these expected utilities should be considered as the expected utility to find the equilibrium in Bayesian games. Harsanyi [47] proved that any equilibrium in ex ante condition is also an equilibrium under the ex interim condition. This is justified by the assumption that when a player knows its type, the player also knows that the other players do not know its type. Thus, they might consider the player may have a different type which affects their decisions. Therefore, the player has to consider its different types as well even if the player knows its actual type. This fact makes Bayes-Nash equilibrium a natural extension of the Nash equilibrium.

Here, the method for reasoning about Bayesian games is the same as that for reasoning about normal form games. Reasoning about the games stands for reasoning about the rationality of players. Representing the rationality of players implies that they maximise their payoff. In the next section, we propose a language for representing and reasoning about Bayesian games.

3.3.1 Language for Bayesian Games

In this section, we extend the epistemic logic for normal form games for representing and reasoning about the rationality of players in Bayesian games. Consequently reasoning about the rationality of players leads us to reasoning about Bayes-Nash equilibrium. Most of the propositions are the same as those in normal form games, however, we define new propositions for utility and strategy. Before we present the propositions and axioms of the language,

we should give a clear meaning for notation θ_{i_n}. This notation represents a type of player i, from the set $\{\theta_{i_1}, \theta_{i_2}, ..., \theta_{i_n}\}$. In this definition, θ_{i_n} represents the fact that player i has n different types. In addition, the notation θ_{-i_n} is a set of the types of all the players excluding player i, therefore, $\theta_{-i_{(n)}} = \{\theta_{1_n}, ..., \theta_{i-1_n}, \theta_{i+1_n}, ..., \theta_{n_n}\}$. The other critical notation is $\theta_{i_n,-i_n}$. It describes a set of types for player i in combination with other players' types.

- The propositional letter $\mathbf{i}_{m_{\theta_{i_n}}}$ stands for the statement 'i plays its mth strategy in its θ_{i_n} type ' and reveals information about the player and the strategy, and also the type of player.

- The proposition $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n}, \theta_{-i_n}) = \mathbf{r}_{1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ denotes that the utility for player i with type θ_{i_n}, when the strategy profile $(1_{k_1}, ..., N_{k_N})$ is played in combination with other players' types θ_{-i_n}, equals the number r.

- R is a countable set of many symbols such as r. The elements of R represent real numbers.

- The proposition $\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n},\theta_{-i_n}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ states that for player i its mth strategy is at least as good as its nth strategy while other players choose $1_{k_1}, ..., N_{k_N}$ to play when the player has type θ_{i_n} and the type of the other players are θ_{-i_n}.

- The proposition $\mathbf{rational}_i$ denotes the rationality of player i in the sense that i is an expected utility maximiser.

We use the syntax introduced by [39] for probabilistic expressions. $\mathbf{P}_i(.) = .$ represents i's probabilistic belief of player i's type and arbitrary finite sums of such expressions when φ_i's are certain sentences, $\mathbf{q}_i$ and q are rational numbers $\mathbf{P}_i(\varphi_1).\mathbf{q}_1 + ... + \mathbf{P}_i(\varphi_n).\mathbf{q}_n \geq \mathbf{q}$. This formula is called an i-probability formula or simply a probability formula, if i is not mentioned. i-probability formulae are allowed for only one player in formulae. Therefore, the case $\mathbf{P}_i(\varphi_1).\mathbf{q}_1 + \mathbf{P}_j(\varphi_2).\mathbf{q}_2 \geq \mathbf{q}$ would not be valid for $i \neq j$. Also obvious abbreviations use the Σ notation.

To capture probabilistic reasoning, the Kolmogorov axioms are used.

- Nonnegativity: for each i : $\mathbf{P}_i(\theta) \geq 0$.

- True: for each i :$\mathbf{P}_i(\top) = 1$ the probability of the event true is 1.
- False: for each i : $\mathbf{P}_i(\bot) = 0$ the probability of the event false is 0.
- Additivity: for each i : $\mathbf{P}_i(\theta) = \mathbf{P}_i(\theta \wedge \psi) + \mathbf{P}_i(\theta \wedge \neg\psi)$.
- Distributivity: for each i : $\mathbf{P}_i(\theta) = \mathbf{P}_i(\psi)$ whenever $\theta \leftrightarrow \psi$ is a propositional tautology.

In order to ensure that probabilistic and non-probabilistic beliefs are related in the right way, two additional axioms are useful [32].

- Consistency assumption: $K_i\varphi \leftrightarrow \mathbf{P}_i(\varphi) = 1$.

This axiom says that the set of states that player i considers possible has the probability 1. In other words, a formula is inconsistent if a player knows an event is false then the player does not hold 0 probability on that event.

- Uniformity assumption: $\varphi \rightarrow K_i\varphi$ for φ an i-probability formula (the sentence starts with $\mathbf{P}_i$ or Boolean combinations thereof).

This axiom says in a given state, player i knows all i-probability formulae that are true in that state.

Then, there are specific axioms for Bayesian games. Without loss of generality, we consider 2 player Bayesian games.

$$\bigwedge_i \bigwedge_m \neg(\mathbf{i}_{m_{\theta_{i_n}}} \wedge \mathbf{i}_{n_{\theta_{i_n}}}) \tag{$G2'$}$$

$$\bigwedge_i \bigwedge_m (K_i\mathbf{i}_{m_{\theta_{i_n}}} \leftrightarrow \mathbf{i}_{m_{\theta_{i_n}}}) \tag{$G3'$}$$

$$\mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n,-i_n}) = \mathbf{r} \rightarrow \\ K_i\mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n,-i_n}) = \mathbf{r} \tag{$G4'$}$$

$$(\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n,-i_n}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N},\theta_{i_n,-i_n}}) \vee \\ (\mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N},\theta_{i_n,-i_n}} \succeq \mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n,-i_n}}) \tag{$G5'$}$$

Axiom G1′ says that every player plays at least one strategy based on its type θ_{i_n} in each state where m ranges over the available strategies in type θ_{i_n}. Axiom G2′ states that each player cannot choose two or more strategies at each state. Axiom G3′ implies that every player knows its own strategies likewise G4′ says that every player knows its own utilities. The last axiom G5′ states payoffs are comparable under the binary relation $\succeq$ or$\preceq$.

This section has presented the axioms about Bayesian games. However, to validate these axioms a semantics frame for these axioms has to be presented. The semantic applicable to Bayesian games is introduced in the next section.

3.3.2 Semantics for Bayesian Games

We want to extend our logic to cover formulae such as $\mathbf{P}_i(\varphi) \geq b$, which means that "according to player i, formula φ holds with a probability of at least b", where b is an arbitrary real number between [0,1]. We start from a classic Kripke structure and add required features to this structure to develop semantics for Bayesian games.

The standard Kripke model for normal form games was defined in previous section as:

- *Game*-structure = $\langle \Omega, \{\mathcal{K}_i\}_{i \in N}, \{\sigma_i\}_{i \in N} \rangle$.

Ω is a set of states or possible worlds, $\mathcal{K}_i$ is a binary relation on Ω that for every $w \in \Omega$ and for every $i \in N$, $\mathcal{K}_i$ is defined as:

- $\mathcal{K}_i = \{w' \in \Omega : w\mathcal{K}_i w'\}$.

σ_i are functions that:

- $\sigma_i : \Omega \longrightarrow S_i (i \in N)$

satisfying that:

- if $w' \in \mathcal{K}_i(w)$ then $\sigma_i(w') = \sigma_i(w)$.

A basic assumption for the set of subsets of possible worlds Ω to which probability assigned is that these subsets fulfil some closure properties. For example, if both u and v have some probability then it is possible that $u \cup v$

and $\neg u$ have a probability too. Here, $\neg$ is not negation complement, for example if $u = (\mathbf{P}_i(\varphi) > b)$ then $\neg u$ is $\mathbf{P}_i(\varphi) \leq b$.

This is the definition of σ-algebra of subsets with one extra assumption that a probability is assigned to the union of sets for countable sets of worlds.

From probability theory, a probability space is a tuple $(\Gamma, \mathcal{H}, \mu)$ where Γ is a set called the sample space, $\mathcal{H}$ is a σ-algebra of subsets of Γ, whose elements are called measurable sets, and a probability measure $\mu : \mathcal{H} \to [0, 1]$ satisfies the following two properties:

- $\mu(\Gamma) = 1$.
- $\mu(u \cup v) = \mu(u) + \mu(v)$ if u and v are disjoint elements of $\mathcal{H}$.

Given a *Game*-structure $= \langle \Omega, \{\mathcal{K}_i\}_{i \in N}, \{\sigma_i\}_{i \in N} \rangle$ to check whether a probability formula is true at a state w, a probability space is assigned to each state w. Thus, we extend *Game*-structure by adding $\mathcal{P}_i$ which is a probability assignment to each player i and state $w \in \Omega$ a probability space $\mathcal{P}(i, w) = (\Omega, \mathcal{H}_{i,w}, \mu_{i,w})$ where $\mathcal{H}_{i,w} = \mathcal{H}_i(w)$ and $\mathcal{H}_i(w) \subseteq \Omega$. Without loss of generality, we assume $\mathcal{H}_i(w)$ is measurable. The extended *Game*-structure with a probability space is *Game*-structure $= \langle \Omega, \{\mathcal{K}_i\}_{i \in N}, \{\sigma_i\}_{i \in N}, \{\mathcal{P}_i\}_{i \in N} \rangle$ $\mathcal{P}(i, w)$ can be considered as $\Delta(\Omega)$ that denotes the set of probability distributions over Ω. Therefore $\mathcal{P}_i : \Omega \longrightarrow \Delta(\Omega)$. It means that $\mathcal{P}_i$ are the sets $\{\mu \in \Delta(\Omega) : \mu(E) \geq \alpha\}$ for all $E \in \sigma_i(w)$ and real number $\alpha \in [0, 1]$.

Game-model (structure) is also extended by adding the following valuation:

- $w \models \mathbf{i}_{m\theta_{i_n}}$ if and only if $\sigma_i(w) = \mathbf{i}_{m\theta_{i_n}}$ and $\mu_i(\sigma_i(w)) = \theta_{i_n}$.

It says that $\mathbf{i}_{m\theta_{i_n}}$ is true in the *Game*-model at state w when player i with type θ_{i_n} selects strategy $\mathbf{i}_m$ if and only if the strategy of player i related to state w by function σ is $\mathbf{i}_m$ and the probability measure of $\sigma_i(w)$ happening is equal to θ_{i_n}.

- $w \models \mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n}, \theta_{-i_n}) = \mathbf{r}_{i,1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ if and only if $u(\sigma_i(w), \sigma_{-i}(w)) = r$ and $\mu_{i,w}(\sigma_i(w), \sigma_{-i}(w)) = \theta_{i_n,-i_n}$.

It states that $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n}, \theta_{-i_n}) = \mathbf{r}_{i,1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ is true in the *Game*-model at state w, where the payoff of player i with type θ_{i_n} in combination with the type of other players θ_{-i_n} is given by $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n}, \theta_{-i_n}) =$

$\mathbf{r}_{i,1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ if and only if the strategy profile of all players with type θ_{-i_n}, related to state w with function $\sigma_{-i}(w)$. In addition, $\sigma_{-i}(w)$ in combination with player i's strategy profile is equal to r $(u(\sigma_i(w), \sigma_{-i}(w)) = r)$. Furthermore, the probability measure of tuple $(\sigma_i(w), \sigma_{-i}(w))$ is equal to $\theta_{i_n,-i_n}$.

- $w \models (\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n},\theta_{-i_n}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N},\theta_{i_n},\theta_{-i_n}})$ if and only if $\mu_{i,w}(u(\mathbf{i}_{m_{\theta_{i_n}}}, \sigma_{-i}(w))) \succeq \mu_{i,w}(u(\mathbf{i}_{n_{\theta_{i_n}}}, \sigma_{-i}(w)))$.

It says that $\mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n},\theta_{-i_n}} \succeq \mathbf{r}_{1_{k_1},...,i_n,...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ is true in the *Game*-model at state w, where player i's strategy $\mathbf{i}_m$ with type θ_{n_i} is at least as good as strategy $\mathbf{i}_n$ with the same type if and only if probability measure of combination of $\mathbf{i}_m$ with $\sigma_{-i}(w)$ results at a payoff at which player i gains at least as good as the payoff of probability measure of combination of $\mathbf{i}_n$ with $\sigma_{-i}(w)$.

- $w \models \mathbf{P}_i(\varphi) \geq b$ if and only if $\mu_{i,w}(\mathcal{H}_{i,w}(\varphi)) \geq b$ while $\mathcal{H}_{i,w}(\varphi) = \{w' \in \mathcal{H}_{i,w} | w' \models \varphi\}$.

It says formula φ holds with a probability of at least b if and only if measure $\mathcal{H}$ is at least b.

For reasoning about Bayesian games as in normal form games, the rationality of players also needs to be represented. We extend the axiom RATIONAL for normal form games to the axiom $\mathbf{rationaltype}_i$ which is the formalism of utility maximisation by considering a player's type. It is defined for N player Bayesian games as follows:

$$\begin{aligned}\mathbf{RATIONALTYPE}: \mathbf{rationaltype}_i \leftrightarrow (K_i \bigwedge_{i_{k_i}} \sum_{i=1}^{q} (\mathbf{P}_i(\theta_{i_n,-i_n}) \\ (\mathbf{u}_i(1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}) \wedge \mathbf{i}_{m_{\theta_{i_n}}})) \\ \rightarrow \bigwedge_{i_{k_i}} \sum_{i=1}^{q} \mathbf{P}_i(\theta_{i_n,-i_n}) \mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n},\theta_{-i_n}} \geq \\ \sum_{i=1}^{q} \mathbf{P}_i(\theta_{i_n,-i_n}) \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}})\end{aligned}$$

Here q is the number of available types for player i. The axiom $\mathbf{rationaltype}_i$ states that player i is a utility maximiser whenever, the player decides to play its m_{θ_i}th strategy in a situation in which the player knows probabilistic beliefs $\mathbf{P}_i(\theta_{i_n,-i_n})$ and utility (captured by the $K_i \bigwedge_{i_{k_i}} \sum_{i=1}^{m}(\mathbf{P}_i(\theta_{i_n,-i_n})$ $(\mathbf{u}_i(1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}))$. According to this, the $m_{\theta_{n_i}}$th strategy is better than any other, given the player's beliefs.

At this stage the valuation function for *Game*-model satisfies the following extra condition:

- $w \models \mathbf{rationaltype}_i$ if and only if, for every $s_{i_{\theta_{n_i}}} \in S_i$ there exists $w' \in \mathcal{K}_i(w)$ such that $u(\mathbf{i}_{m_{\theta_{n_i}}}, \sigma_{-i}(w')) \succeq u(s_{i_{\theta_{n_i}}}, \sigma_{-i}(w'))$ and $\mu_{i,w}(\sigma_i(w), \sigma_{-i}(w)) = \theta_{n_i,n_{(-i)}}$.

The syntax and semantics have been proposed for Bayesian games by considering different types for each player. We established the soundness of the logic below.

Proposition 2: The proposed logic is sound with respect to the class of *Game*-models.

Proof

- Axioms G1′ and G2′ say that at each state for each type of player only one strategy is played. G′1 and G′2 are valid because at each state w and each type θ_{i_n} of player i, there is a unique strategy such that $\mathbf{i}_{m_{\theta_{i_n}}}$ by validation rule $w \models \mathbf{i}_{m_{\theta_{i_n}}}$ if and only if $\sigma_i(w) = \mathbf{i}_{m_{\theta_{i_n}}}$ and $\mu_i(\sigma_i(w)) = \theta_{i_n}$.

- Axiom G3′ is valid because if $w' \in \mathcal{K}_i(w)$ then $\sigma_i(w) = \sigma_i(w')$ and by using μ_i we have $\mu_i(\sigma_i(w')) = \mu_i(\sigma_i(w)) = \theta_{n_i}$.

- Axiom G4′ is valid because if $w' \in \mathcal{K}_i(w)$, then $\sigma_i(w') = \sigma_i(w)$, and then we have $u(\sigma_i(w), \sigma_{-i}(w)) = u(\sigma_i(w'), \sigma_{-i}(w))$. Also by applying μ_i we have $\mu_i(\sigma_i(w')) = \mu_i(\sigma_i(w)) = \theta_{i_n}$. Furthermore, because each $\sigma_{-i}(w))$ is unique and has unique type θ_{-i_n} we have $\mu_{i,w}(\sigma_i(w'), \sigma_{-i}(w)) = \theta_{i_n,-i_n}$.

- Axiom G5′ is valid because for every state w there is a unique $\sigma_{-i}(w)$ for players excluding player i and the ordering of $u_i(1_{k_1},...,N_{k_N})$ induces an ordering of $\bigwedge_m u(i_m, \sigma_{-i}(w))$. In addition, by using μ_i we have $\mu_{i,w}(u(\mathbf{i}_{m_{\theta_{i_n}}}, \sigma_{-i}(w))) \succeq \mu_{i,w}(u(\mathbf{i}_{n_{\theta_{i_n}}}, \sigma_{-i}(w)))$.

- **rationaltype**$_i$ is valid. Suppose that:
  $w \models (K_i \bigwedge_{i_{k_i}} \sum_{i=1}^{q} (\mathbf{P}_i(\theta_{i_n,-i_n})$
  $(\mathbf{u}_i(1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}) = \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}) \wedge \mathbf{i}_{m_{\theta_{i_n}}})$
  $\rightarrow \bigwedge_{i_{k_i}} \sum_{i=1}^{q} \mathbf{P}_i(\theta_{i_n,-i_n}) \mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n},\theta_{-i_n}} \geq$
  $\sum_{i=1}^{q} \mathbf{P}_i(\theta_{i_n,-i_n}) \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$

  - $K_i \bigwedge_{i_{k_i}} \sum_{i=1}^{q} (\mathbf{P}_i(\theta_{i_n,-i_n})(\mathbf{u}_i(1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}) =$
    $\mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}})$ means that for every w' where $w' \in K_i(w)$ based on the definition of the binary relations $\mathcal{K}_i$, $\sigma_i(w') = \sigma_i(w)$ and as for each state w there is a unique $\sigma_{-i}(w)$, and probabilistic beliefs $\mathbf{P}_i(\theta_{i_n,-i_n})$, there exists $\mu_i(\sigma_i(w'), \sigma_{-i}(w)) = r$,
  - $\mathbf{i}_{m_{\theta_{i_n}}}$ means $\sigma_i(w) = \mathbf{i}_{m_{\theta_{i_n}}}$ and $\mu_i(\sigma_i(w)) = \theta_{i_n}$,
  - $\bigwedge_{i_{k_i}} \sum_{i=1}^{q} \mathbf{P}_i(\theta_{i_n,-i_n}) \mathbf{r}_{1_{k_1},...,i_m,...,N_{k_N},\theta_{i_n},\theta_{-i_n}} \geq$
    $\sum_{i=1}^{q} \mathbf{P}_i(\theta_{i_n,-i_n}) \mathbf{r}_{1_{k_1},...,i_{k_i},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$ says for all strategies and types of player i we have $\mu_{i,w}(u(\mathbf{i}_{m_{\theta_{i_n}}}, \sigma_{-i}(w))) \succeq \mu_{i,w}(u(\mathbf{i}_{n_{\theta_{i_n}}}, \sigma_{-i}(w)))$,

  Therefore for every $i_m \in S_i$ there exists a $w' \in \mathcal{K}_i(w)$ such that $u(\mathbf{i}_{m_{\theta_{i_n}}}, \sigma_{-i}(w')) \succeq u(s_{i_{\theta_{i_n}}}, \sigma_{-i}(w'))$ and $\mu_{i,w}(\sigma_i(w), \sigma_{-i}(w)) = \theta_{i_n,-i_n}$.

■

We have demonstrated that, the formal logic is sound and can be applied by a rational player to make decisions in a Bayesian game. The Bayesian game shown in table 3.2 consists of four normal form games. To represent this game by a formal language and reason about it, the system has to be represented as a finite state system. A finite state system has a finite number of states and relations between these states. For this game the figures 3.2 and 3.3 show states and relations, respectively. The number of players is two throughout all the four games. In each game the row player is player 1 and the column player is player 2. The set of strategies in each normal form game is:

$$S = \{(U, L), (U, R), (D, L), (D, R)\}$$

Each normal form game is represented by four states. In section 3.1.2, the knowledge of each player in the first normal game, which is called MP, is represented in figure 3.3. The relation between states 1 and 2, indicated by a dashed line, stands for player 2's knowledge, and the same applies to the relation between states 3 and 4. For player 1, games MP and PD are the same since they have its first type. Player 2 cannot distinguish between games MP and Coor from each other as in these games player 2 has identical types. However, in each game the combination of players' types is unique, which is represented by a probability about every game. Therefore, each game of four states is separated from the other states by the probability about the game. In figure 3.2, the probability for player 1 with type 1 is shown as $\mathbf{P}_1(\theta_1)$. The combination of this notation with player 2's notation $\mathbf{P}_2(\theta_1)$ is equal to the probability of playing the game MP. Each player knows this distribution, which means, there are relations between games in a Bayesian game in figure 3.3. These relations are depicted for each first state of every game as solid lines. There are relations between each state equal to the ones drawn for states 1, 5, 9 and 13. In other words, each state from one game has relations to all the other states of other games. The order of the game is not important, because at the end of a Bayesian game, a player has to consider all the games. In fact if a player plays the game MP either first or last, this does not effect the final payoff.

Each player can choose the same number of strategies as the number of its types. For example, in the game that is shown in figure 3.2, every player has two types, therefore, the player has to select two numbers of strategies. The reason is that, a Bayesian game is divided in to different sets of normal form games, by the observation of each player. This observation is the type of the player. Hence each player has to choose a strategy from each set of normal form games. In the following equation, player 1 chooses U for each type and depending on strategies player 2 chooses, a final payoff for both players will be determined.

$$\begin{aligned}\models (\mathbf{1}_{U_{\theta_{1_1}}} \wedge \mathbf{1}_{U_{\theta_{1_2}}}) \wedge ((\mathbf{2}_{L_{\theta_{2_1}}} \wedge \mathbf{2}_{L_{\theta_{2_2}}}) \vee (\mathbf{2}_{L_{\theta_{2_1}}} \wedge \mathbf{2}_{R_{\theta_{2_2}}}))\wedge \\ (\mathbf{P}_1(\theta_{1_1,2_1})(\mathbf{u}_1(1_U, 2_L, \theta_{1_1}, \theta_{2_1})) \wedge \mathbf{P}_1(\theta_{1_2,2_2})(\mathbf{u}_1(1_U, 2_L, \theta_{1_2}, \theta_{2_2}))) \succeq \\ (\mathbf{P}_1(\theta_{1_1,2_1})(\mathbf{u}_1(1_U, 2_L, \theta_{1_1}, \theta_{2_1})) \wedge \mathbf{P}_1(\theta_{1_2,2_2})(\mathbf{u}_1(1_U, 2_R, \theta_{1_2}, \theta_{2_2})))\end{aligned}$$

Figure 3.2: Representing the states of a Bayesian game

Player 2 is irrational if it chooses $(\mathbf{2}_{L_{\theta_{2_1}}} \wedge \mathbf{2}_{R_{\theta_{2_2}}})$. From figure 3.2 we can infer:

- the payoff for each player when the strategy profile is $(\mathbf{1}_{U_{\theta_{1_1}}} \wedge \mathbf{1}_{U_{\theta_{1_2}}}) \wedge (\mathbf{2}_{L_{\theta_{2_1}}} \wedge \mathbf{2}_{L_{\theta_{2_2}}})$

    1. utility for player 1: 2*0.3+2*0.1+2*0.2+2*0.4=2
    2. utility for player 2: 0*0.3+2*0.1+2*0.2+1*0.4=1

- the payoff for each player when the strategy profile is $(\mathbf{1}_{U_{\theta_{1_1}}} \wedge \mathbf{1}_{U_{\theta_{1_2}}}) \wedge (\mathbf{2}_{L_{\theta_{2_1}}} \wedge \mathbf{2}_{R_{\theta_{2_2}}})$

    1. utility for player 1: 2*0.3+0*0.1+2*0.2+0*0.4=1
    2. utility for player 2:0*0.3+3*0.1+2*0.2+0*0.4=0.7

Therefore player 2 is rational if it selects strategies $(\mathbf{2}_{L_{\theta_{2_1}}} \wedge \mathbf{2}_{L_{\theta_{2_2}}})$

3.4 Remarks

In this chapter, we have proposed a formal approach to specifying players' knowledge and probability beliefs in games, specifically in normal form games and Bayesian games. Moreover, we have represented these games by this formal logic. We have also represented the rationality of players. Therefore, we can reason about the behaviour of a rational player in a game. This has been done by detecting the set of strategies that satisfy the characterisations of rationality. This is important because this logic is intended to make precise the informal notation of rationality that is appropriate for the situation in which rational players interact in the way specified in the theoretical description of games [104]. Furthermore, an implicit analysis of game theoretic rationality is interpreted as the specification of strategies-definition of a solution to a game [104]. Thus, by representing and reasoning about the rationality of the players, we have represented and reasoned about solutions to games.

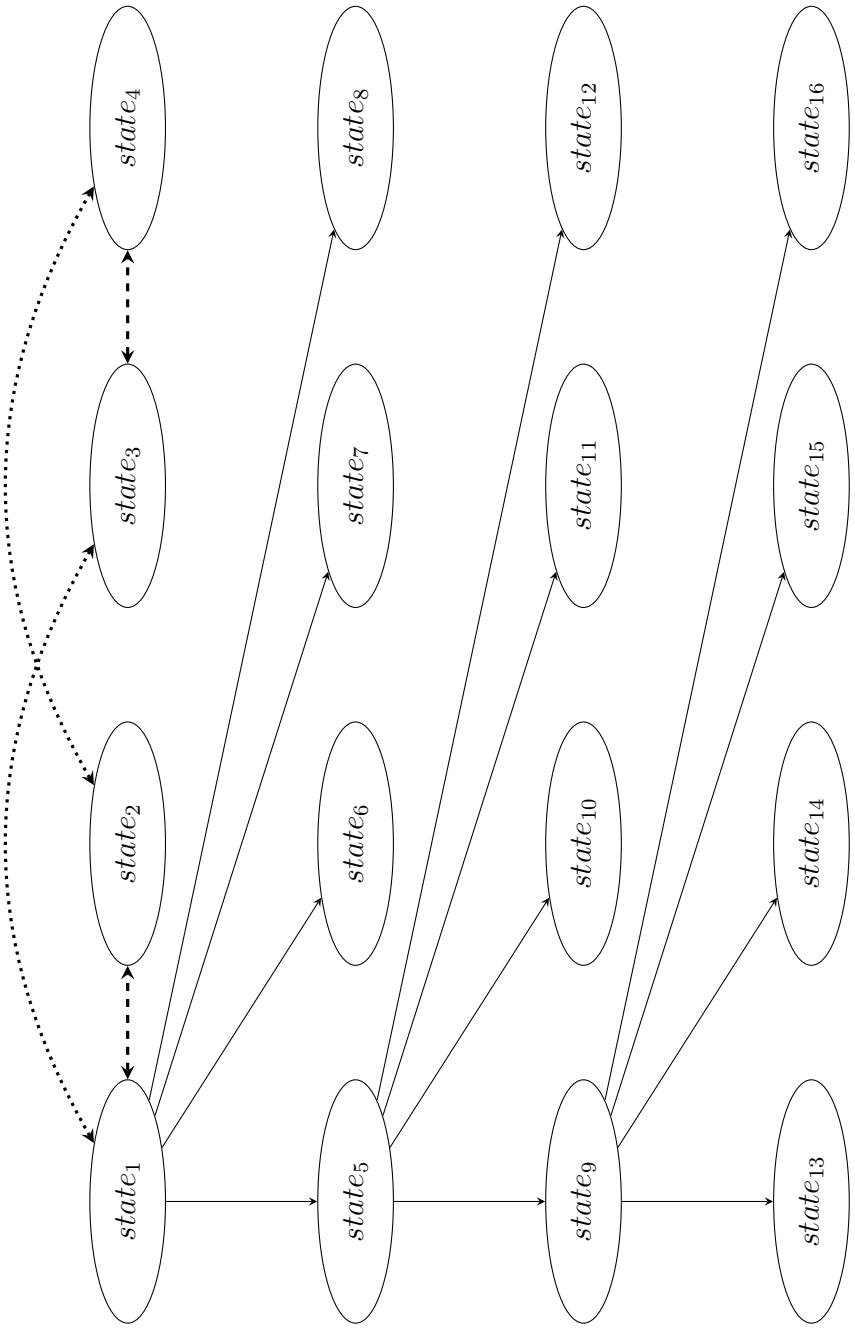

Figure 3.3: Representing relations between the states of a Bayesian game

Chapter 4

Model Checking

Model checking is a method in the domain of formal verification. Formal verification is a systematic approach that benefits from mathematical reasoning to check the correctness of a system. There are a variety of tools that offer model checking for different modal languages. However, due to the characteristics of the proposed formal language, the already developed tools show lack of setting to model a system based on the language.

We have implemented a model checking system which supports epistemic logic for normal form games and Bayesian games. The system uses the syntax and the semantics that was proposed in chapter 3. This chapter introduces the system, the language of the system to model games and defines the language which is used to describe the specifications in this system. Furthermore, different algorithms are presented which are implemented in the system.

4.1 Model Checking Games

The system analyses the scenarios that have the following structure. A situation as a game is modelled where some number of agents (players) interact. In a state of a model based on agents' knowledge and the chosen strategies payoffs can be determined. The agents' rationality is therefore used as a criterion to choose a strategy. The input to the system consists of a file which contains different formulae that represent a variety of concepts:

- the possible states of the model,
- the names of agents,
- the strategy of each agent,

- other useful information such as agent's utility,
- the probability belief in Bayesian games, the games' specification.

The source of a deadlock is often a wait-for relation between states. To prevent deadlocks, we assume that all above concepts are non-empty when the model is a Bayesian game. An exception is made for the probability belief if the probability belief is not given, the model is a normal form game.

The system accepts different approaches to representing the epistemic aspect of the formulae. The descriptions of the formulae are based on the epistemic logic that are developed in chapter 3. This logic for normal form games and Bayesian games belongs to a set of modal logics. In this logic, the modal operator is related to the information available to agents in a distributed or multi-agent system. The syntax of the logic for these games is defined in the previous section. The grammar applied in this section is:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid K_i\varphi \mid P_i\theta$$

where p is an atomic proposition and $i \in \{1...n\}$ is an agent in a game with n agents.

The semantics that is introduced in the previous chapter described knowledge modal operator K in the Kripke structure and $\mathbf{P}$ as probability belief. We model a normal form game or a Bayesian game by this semantics as a finite-state transition system. Therefore, verifying a property in this model can be achieved by model checking techniques. Each model of games consists of a set of states. In the normal form games and Bayesian games, players have to choose between different strategies simultaneously. To represent games as finite state systems, in addition to states, the transition between states should be defined. For these games a transition is a set of strategies that are chosen when each agent has information about those strategies. Moreover, the set of strategies should fulfil certain conditions such as maximising the payoff. This condition relates to the agent's rationality.

In a game, each agent starts from one state to play the game and uses its observation to gain information about the model since agent makes inferences about the state just based on its last observation. As the order of a state does not have any effect on the final result of a game, the starting state can

be randomly chosen from a set of states. The reason is that, all the states in a game are observed by agents to optimise for the best strategy therefore the order of checking is irrelevant. The observation and the behaviour of the agents are precisely determined in the system by formulae of the logic.

The system reads each formula from an input file. Then, a formula is parsed as a tree by using ANTLR (ANother Tool for Language Recognition) [80]. ANTLR is a powerful parser generator for reading, processing, executing, or translating structured text or binary files.

To obtain each formula as a tree structure, we have to define a grammar of the developed epistemic logic as executable programs first. The language of these programs is a domain-specific language (DSL), which is designed for expressing language structure. Then ANTLR can automatically convert grammars to parsers for us. Figure 4.1 depicts an example of trees generated by ANTLR based on our grammar.

In the tree shown in figure 4.1, agents as player 1 (the row player) and player 2 (the column player), are represented as ?1 and ?2, respectively. The player's nodes have leaves as the strategy of the player, e.g. player 1's child node is the strategy U and player 2's child node is the strategy D. The other leaves represent the probability (type) of the game as $(p = 0.3)$. The other leaf nodes in the format of $(u = 2)$ shows the utility for each player. Therefore, each leaf node of these trees is an atomic proposition which is a formula, and when φ and ψ are formulae, other formulae such as $\neg\varphi$, $\varphi \wedge \psi$ and $K_i(\varphi)$ for each $i = 1, ..., n$ can be defined. This technique for representing each state by a formula is known as a symbolic model checking technique. A symbolic technique is used because a formula is assigned to every state.

The grammar is generated into two Java files, lexer and parse by ANTLR version 1.4.3.

- a lexer: reads an input character, divides it into tokens using the developed epistemic logic grammar, and generates a token stream as output. It can also flag some tokens such as white space and comments as hidden using a protocol that ANTLR parsers automatically understand and respect.

- a parser: reads a token stream (generated by a lexer), and matches phrases in the developed epistemic logic language via the specified

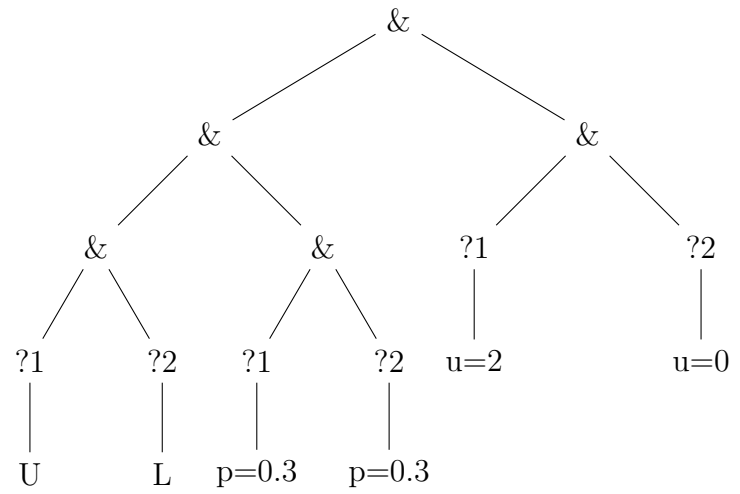

Figure 4.1: Parsing a state formula

rules, and typically performs some semantic action for each phrase (or sub-phrase) matched. Each match could invoke a custom action, or generate an abstract syntax tree for additional processing.

The system reads these tree structures and with the provided information it generates a finite-state transition system. The language which is used for implementing the system is Java.

Figure 4.2 lists the main components of the system that implements the algorithms which are present in section 4.4. The steps that are performed automatically upon invocation of the system are as follows:

- In step 2, the input file is parsed. In this step various parameters are stored in temporary lists.
- In step 3, the formula to be checked is read from a text file, and parsed appropriately.
- In step 4, the list obtained in step 2 are traversed to build states for the verification algorithms, then verification is performed by running the algorithms. The list obtained in step 3 are traversed to build states of formula. The states of model in which a formula holds are computed.
- In step 5, the set of reachable states is detected based on the knowledge accessibility.

- In step 6, the set of reachable states is detected based on the probability belief accessibility.
- In step 7, compute the expected utility for states that has relations.
- In step 8, compare the expected utility of combination of related states to detect the greatest expected utility.
- In step 9, the set of reachable states from step 5 and 6 are compared with the states in which a formula holds. If there is set of states in the model that is same as states of the formula, the system return a positive output or the greatest expected utility.
- In step 10, the set of reachable states from step 5 and 6 are compared with the states in which a formula holds, If there is no set of states in the model that is same as states of the formula, the system return a negative output.

Step 2 and 3, inside the light grey box, are performed parallel. Step 7 and 8 are performed parallel too.

In the next section, we explain step 2 and step 4 in more details.

4.2 Model Checker's Input Language

Model checking is based on the construction and analysis of a model. In addition to a model there is a desired property that should be verified against all available states in the model. Depth first search can be used to check a property in a set of all available states. In this section, we discuss the input language of the developed model checker for normal form games and Bayesian games. An input script consists of a set of states declarations and one or more specifications as illustrated in figure 4.3. We begin by describing the lexical conventions, the structure of state declarations, and finally specification language.

The model checker's lexical structure is as follows:

- *Comments*: the system supports single line comments that begin with "__".

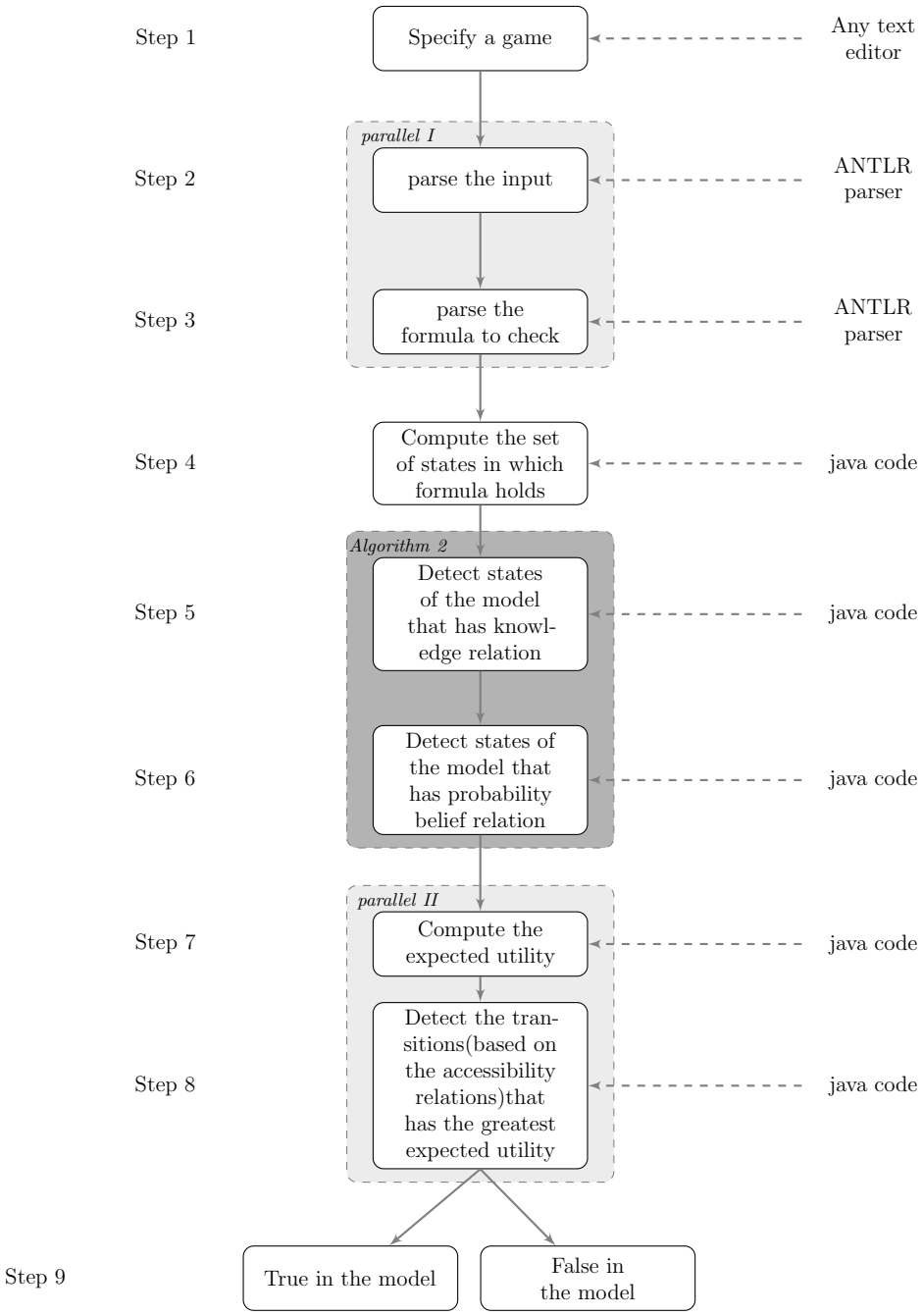

Figure 4.2: System structure

- *Constants*: strategies, agent names, agent types, and agent utilities are constants. They begin with lower-case letters followed by any number of a mix of alphanumeric characters and underscores.

The system also supports different arithmetic types, such as natural

numbers and rational numbers.
The subformula that is not only an atomic proposition is called an expression. The combination of the expressions forms formulae to represent games. The syntax of expressions depends on the type of the value they represent, namely boolean and numerical.

- *boolean expressions*: are expressions which are composed of constants atomic propositions using boolean operators & (and), |(or) and $->$ (implication).
- *numerical expressions*: are expressions that are formed from numerical constants using operators * and + .

Besides the mentioned expressions, several examples of the basic expressions are given in table 4.1.

Operator	Expression
Equality	$u = 0$
Relational	$u > 0$
Agent's name	?2

Table 4.1: Examples of basic expressions

The input to the system shown in figure 4.3 describes a Bayesian game that represented different states in table 3.2 (on page 57). As mentioned before, the structure of each state is determined by a formula in which the agent's name, the agent's strategy and the type of the game are mentioned. As shown in figure 4.3, input lines without p : represent the states and the last line with p : is the property that is needed to be verified in the model.

4.3 Model Checker's Specification Language

Same as the input language of the system, the specification language is slightly different from the syntax of the proposed logic. The specification language is able to express different aspects of agent's knowledge. The available operators for knowledge and their descriptions are as follows:

$(?1(U)\&?2(L))\&(?1(p=0.3)\&?2(p=0.3))\&(?1(u=2)\&?2(u=0))$

$(?1(U)\&?2(R))\&(?1(p=0.3)\&?2(p=0.3))\&(?1(u=0)\&?2(u=2))$

$(?1(D)\&?2(L))\&(?1(p=0.3)\&?2(p=0.3))\&(?1(u=0)\&?2(u=2))$

$(?1(D)\&?2(R))\&(?1(p=0.3)\&?2(p=0.3))\&(?1(u=2)\&?2(u=0))$

$(?1(U)\&?2(L))\&(?1(p=0.1)\&?2(p=0.1))\&(?1(u=2)\&?2(u=2))$

$(?1(U)\&?2(R))\&(?1(p=0.1)\&?2(p=0.1))\&(?1(u=0)\&?2(u=3))$

$(?1(D)\&?2(L))\&(?1(p=0.1)\&?2(p=0.1))\&(?1(u=3)\&?2(u=0))$

$(?1(D)\&?2(R))\&(?1(p=0.1)\&?2(p=0.1))\&(?1(u=1)\&?2(u=1))$

$(?1(U)\&?2(L))\&(?1(p=0.2)\&?2(p=0.2))\&(?1(u=2)\&?2(u=2))$

$(?1(U)\&?2(R))\&(?1(p=0.2)\&?2(p=0.2))\&(?1(u=0)\&?2(u=0))$

$(?1(D)\&?2(L))\&(?1(p=0.2)\&?2(p=0.2))\&(?1(u=0)\&?2(u=0))$

$(?1(D)\&?2(R))\&(?1(p=0.2)\&?2(p=0.2))\&(?1(u=1)\&?2(u=1))$

$(?1(U)\&?2(L))\&(?1(p=0.4)\&?2(p=0.4))\&(?1(u=2)\&?2(u=1))$

$(?1(U)\&?2(R))\&(?1(p=0.4)\&?2(p=0.4))\&(?1(u=0)\&?2(u=0))$

$(?1(D)\&?2(L))\&(?1(p=0.4)\&?2(p=0.4))\&(?1(u=0)\&?2(u=0))$

$(?1(D)\&?2(R))\&(?1(p=0.4)\&?2(p=0.4))\&(?1(u=1)\&?2(u=2))$

$P:(?1(U)\&?2(L))\&(K(?1(rational))\&K(?2(rational)))$

Figure 4.3: The system input format

Operator	Description
K	knowledge
P	probability belief

Table 4.2: Knowledge operators

The knowledge modality is written $K(?agent(formula))$. The probability belief for each agent is written $?agent(p =$ numerical type$)$.

The probability belief should be presented in a Bayesian game, because each state of the game has a probability. In this system, the representation of type (probability belief) is different from the formal logic. To illustrate this difference, we explain the first line of the input in figure 4.3:

$$(?1(U)\&?2(L))\&(?1(p = 0.3)\&?2(p = 0.3))\&(?1(u = 2)\&?2(u = 0))$$

The formula consists of a variety of expressions. The formula that contains expressions such as $?1(p = 0.3)$ indicates that the formula is related to a Bayesian game. Each expression is explained as follows:

- $?1(U)$ is the representation of $\mathbf{1}_{U_{\theta_0}}$
- $?2(L))$ is the representation of $\mathbf{2}_{L_{\theta_0}}$
- $?1(p = 0.3)$ is the representation of $\mathbf{P}_1(\theta_0)$
- $?2(p = 0.3)$ is the representation of $\mathbf{P}_2(\theta_0)$
- $?1(u = 2)$ is the representation of $\mathbf{u}_1(\mathbf{1}_{U_{\theta_0}}, \mathbf{2}_{L_{\theta_0}})$
- $?2(u = 0)$ is the representation of $\mathbf{u}_2(\mathbf{1}_{U_{\theta_0}}, \mathbf{2}_{L_{\theta_0}})$

A game model has different states that represent situations of a game and relations between these states. In this system, there are different relations for representing different scenarios. These relations are the representation of player's knowledge. From the epistemic perspective, agents use their observations in different ways to consider what they know. A player's observation determines its knowledge about the other players' strategies. This knowledge is represented as the relation that connects states to each other. How states are accessible from each other then represents the relation between states. It means that player i has knowledge about the other players' strategies when in the possible worlds (states) for player i, the strategies of the other players are the same. Therefore, $K_i(strategy_j)$ means if player i in state r has access to state p then $strategy_j$ in r is equal to $strategy_j$ in p.

In a normal form game with two strategies for each player, each player has two knowledge relations with the other player's strategy as shown in figure

3.1. In addition to this knowledge, in a Bayesian game, a set of states has also a relation to other set of states, when an agent knows the type of each state. Therefore, in a Bayesian game, different games are played and since all of these games are probable players consider all games to be played. These games might be played by different or equal probabilities. To represent the probability distribution over these games in the formal language, we used types as it is proposed by [47]. A type in a Bayesian game can be presented from two different perspectives.

Players have different types in each normal form game of a Bayesian game or the normal form games have different types. We can prove that these two points are equal. Considering that players have different types, so they play games with different types. This however, does not mean that for every game a player has a different type because this would then imply that all players have complete knowledge about the games. Without loss of generality, we assume a two player Bayesian game. A set of games are indistinguishable for a player when the other player has different types in this set. However, the first player has one constant type in all these games. If the first player has different types in each of these games, the set of these games would not be indistinguishable. Although the players have the same types, in some games the combination of the player's type and the other player's type is unique in each game. This combination represents the probability of each game to be played. Thus, each normal form game has a type in a Bayesian game.

For the purpose of implementation, we represent the probability belief as the type of a game. As mentioned above, the type of a game is the result of the combination of players' types and represents the probability of playing the game. Player i in state r has knowledge about the type of a set of states when the player has access to all these states and the states have the same type. It is represented in formal logic as $\mathbf{P}_i(\theta_{i_n}, \theta_{-i_n})$ for the states which are accessible for player i. Thus, two atomic propositions θ_{i_n} and θ_{-i_n} as the type of player i and the type of other players, respectively, are represented as one proposition. This proposition is $?i(p =)$.

The specification language is used to represent the desired properties of games that we want to check. For example consider the following property corresponds to the last line in figure 4.3:

$$((K_1(\mathbf{rationaltype}_{2_{\theta_0}})) \wedge (K_2(\mathbf{rationaltype}_{1_{\theta_0}}))) \wedge (\mathbf{1}_{U_{\theta_0}} \wedge \mathbf{2}_{L_{\theta_0}})$$

The result from the system is states 0,4,8,12 which means:

(4.1)
$$(((K_1(\mathbf{rationaltype}_{2_{\theta_0}})) \wedge (K_2(\mathbf{rationaltype}_{1_{\theta_0}}))) \wedge (\mathbf{1}_{U_{\theta_0}} \wedge \mathbf{2}_{L_{\theta_0}})) \rightarrow (\mathbf{1}_{U_{\theta_1}} \wedge \mathbf{2}_{L_{\theta_1}})$$

Based on table 3.2, equation 4.2 is the correct move. When player 1 chooses to play U with its first type θ_0 and player 2 plays L with its first type θ_0, the best solution is for both players to play the same strategies, U and L with their second types θ_1. This is the meaning of equation 4.2. As in a game, a different solution could be possible, so an alternative best response in this game with the same start is states 0,5,10,15:

(4.2)
$$(((K_1(\mathbf{rationaltype}_{2_{\theta_0}})) \wedge (K_2(\mathbf{rationaltype}_{1_{\theta_0}}))) \wedge (\mathbf{1}_{U_{\theta_0}} \wedge \mathbf{2}_{L_{\theta_0}})) \rightarrow (\mathbf{1}_{D_{\theta_1}} \wedge \mathbf{2}_{R_{\theta_1}})$$

Equation 4.2 shows an alternative solution for the same game from table 3.2. If the players start with the same assumption as above, this solution is that player 1 plays D and Player 2 chooses R with their second types θ_1.

The above example shows how to define game properties with the specification language of the system. The next section discusses the algorithms that are implemented in the system to model and verify the desired properties in games.

4.4 Model Checker's Algorithms

Model checking has a variety of techniques, such as symbolic model checking using Binary Decision Diagram, explicit state model checking, and bounded model checking. Although the system can use the symbolic model checking approach, the default technique used in this thesis is explicit state model checking. That technique in model checking explicitly constructs all reachable states of a model and then inductively checks these states with the expressions of the specification formula that hold at these states. This technique is well known as a classical approach in model checking.

In this section, we propose an algorithm for model checking of the games. The problem of model checking can be formally stated as follows: given a property (or a logical formula) φ, and a model M, return the set of states Ω such that $w \in \Omega$ if and only if φ is true at state w in M. The input model M here for the model checking algorithm is $\langle \pi, \langle \Omega, \mathcal{K}_1, ..., \mathcal{K}_n, \mathbf{P}_1, ..., \mathbf{P}_n \rangle \rangle$, which is the same as the model defined in section 3, in addition to the extra function π. This function and other input parameters for the model checking algorithm are as follows:

- The boolean variables for a normal form game $\mathbf{i}_m$,$\mathbf{u}_i(1_{k_1}, ..., N_{k_N}) = \mathbf{r}_{i,1_{k_1},...,N_{k_N}}$, $\mathbf{r}_{i,1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{i,1_{k_1},...,i_n,...,N_{k_N}}$ and $\mathbf{rational}_i$.
- The boolean variables for a Bayesian game $\mathbf{i}_{m\theta_{i_n}}$, $\mathbf{u}_i(1_{k_1}, ..., N_{k_N}, \theta_{i_n}, \theta_{-i_n}) = \mathbf{r}_{i,1_{k_1},...,N_{k_N},\theta_{i_n},\theta_{-i_n}}$, $\mathbf{r}_{i,1_{k_1},...,i_m,...,N_{k_N}} \succeq \mathbf{r}_{i,1_{k_1},...,i_n,...,N_{k_N}}$ and $\mathbf{rationaltype}_i$.
- The function $\pi(p)$ which gives the set of states in which the atomic proposition p holds.
- Boolean functions R_i^K encoding the epistemic accessibility relations (for each agent i this function takes as input a state w and a binary relation R on the set of states, and returns the set of states accessible from w via R)
- Boolean functions $R_i^{\mathbf{P}}$ encoding the probability belief accessibility relations (for each agent i this function takes as input a state w and a binary relation R on the set of states, and returns the set of states accessible from w via R)

This algorithm is similar, to model checking techniques for modal logics [88] [51], without temporal modalities.

Algorithm 1 Break down a formula in the system

function VERIFY (φ, M) ▷ returns a subset of states
φ is an atomic formula:
return $\pi(\varphi)$
φ is $\neg\varphi_1$:
return $\Omega\backslash$ verify (φ_1, M)
φ is $\varphi_1 \vee \varphi_2$:
return verify$(\varphi_1, M) \cup$ verify(φ_2, M)
φ is $\varphi_1 \wedge \varphi_2$:
return verify$(\varphi_1, M) \cap$ verify(φ_2, M)
φ is $K_i\varphi$:
return $\{g | R_i^K(g) \subseteq verify(\varphi, M)\}$
φ is $\mathbf{P}_i\varphi$:
return $\{g | R_i^{\mathbf{P}}(g) \subseteq verify(\varphi, M)\}$
end function

In algorithm 1, we only have cases for logical symbols $\neg, \wedge, \vee$ but the formal language from section 3 supports the other symbols such as $\rightarrow, \leftrightarrow$. The reason is that the algorithm covers this set and by this set we can express the other symbols. The use of this set simplifies the model checking algorithm.

This algorithm is correct, which means if, the algorithm is passed a formula φ and the model $M = \langle \pi, \langle \Omega, \mathcal{K}_1, ..., \mathcal{K}_n, \mathbf{P}_1, ..., \mathbf{P}_n \rangle \rangle$, it returns the set of states at which φ is true. Above all, this algorithm terminates, because the recursive calls are all on sub-formulae of the original formula.

A constructed parse tree of a formula, such as shown in figure 4.1, is traced upward towards the root. We recursively compute the set of states satisfying each expression of the parse tree by algorithm 1, and at the end the system has determined the set of states satisfying the formula.

Those formulae as the properties of these models that we are interested to check are shown in table 4.3.

Provided Information	Properties to be checked
a player's strategy and type	other players' strategies and types
a set of players'strategies and types	other set of players'strategies and types

Table 4.3: Properties to be checked

We might also assume that a player knows that the other player is rational which can be written as $K_i(\mathbf{rationaltype}_{j_{\theta_{j_n}}})$. By checking the properties from table 4.3 with this assumption in the model, the Nash equilibrium is determined. For verifying properties in the model we use different algorithms.

Algorithm 2 builds the relations between games in a Bayesian game. This algorithm is marked in figure 4.2 with a dark grey box. This algorithm applies the additional algorithms 3 and 4 to model the relations. A set of relations based on information about the strategies is developed and another set of relations is built based on the type of games.

Algorithm 2 Build Relations in a Bayesian Game

S: array of states in a Bayesian game
R: array of relations in a Bayesian game

```
R_K : null                                  ▷ array of knowledge relations
R_P : null                                  ▷ array of probabilistic belief relations
NoGame: number of normal form games
NoStateInGame: number of states in a normal form game
NoPlayer: number of players in a normal form game
for i := 1, i ≤ count (states) do
    s_i:=set the current state
    for j := 1, j ≤ NoGame do
        g_j:=set the current game
        for n := 1, n ≤ NoPlayer do
            p_n:=set the current player
            if Knowledge(p_n, g_j(s_i), g_j(s_{i+1})) then
                R_K.add(s_i,s_{i+1})
            end if
            for m := 1, m ≤ NoStateInGame do
                if KnowledgeType(g_j, g_{j+1}, p_n, , g_j(s_i), g_{j+1}(s_m)) then
                    s'_i:=s_m (i' is m)
                    R_P.add(s_i,s'_i)
                end if
            end for
        end for
    end for
end for
```

Algorithm 3 checks the strategies in the states. If a strategy for a player is the same in some states, the other players have knowledge about that strategy. This algorithm corresponds to step 5 in figure 4.2.

Algorithm 3 Algorithm Build Knowledge Strategy

```
function KNOWLEDGE (player, state1, state2) ▷ returns a subset of states
    r:=T[player.id, state1.id,state2.id]
    if r ≠ null then
        return r                                    ▷ have already evaluated
    end if
    r:=new pair of states
    T[player.id, state1.id, state2.id]:=r            ▷ add tuple to table
    if player.state1.strategy = player.state2.strategy then
        return r
    end if
end function
```

Algorithm 4 checks the types of the states. The same types of the states means that those states belong to one game. There is relation between games, that are built based on the type of games. For example all the states in a game have one type, therefore from point of view of other states in the model, they are indistinguishable. Step 6 in figure 4.2 corresponds to this algorithm.

Algorithm 4 Algorithm Build Knowledge Type

```
function KNOWLEDGETYPE (game1, game2, player, state1, state2)  ▷
returns a subset of states
    r:=T[game1.id,game2.id, player.id, state1.id,state2.id]
    if r ≠ null then
        return r                                    ▷ have already evaluated
    end if
    r:=new pair of states
    T[game1.id,game2.id, player.id, state1.id, state2.id]:=r  ▷ add tuple to
table
    if game1.player.state1.type = game2.player.state2.type then
        return r
    end if
end function
```

If the input language does not contain a formula with a **P** operator,

the system models states for a normal form game. Otherwise, states are considered for a Bayesian game.

Another important algorithm in this work is the computation of expected utility in a Bayesian game. Algorithm 5 represents step 7 in figure 4.2.

Algorithm 5 Algorithm Computation of Expected Utility in a Bayesian Game

Array of states in a Bayesian games Array of states, each of them from distinct normal form game, that has relations with a specific state and transitions through them give the maximum utility

```
NoGame:number of normal form games
NoStateInGame:number of states in a normal form game
NoPlayer:number of players in a normal form game
for i:=1, i ≤ count states do
    s_i:=set the current state
    for  m := 1, m ≤ NoGame do
        g_m:=set the current game
        for  j := 1, j ≤ NoStateInGame do
            for  n := 1, n ≤ NoPlayer do
                if EXPUTIL (g_m.player_n.s_i) + EXPUTIL (g_{m+1}.player_n.s_j)
is maximum then
                    return (s_i, s_j)
                end if
            end for
        end for
    end for
end for
```

Function EXPUTIL simply computes the expected utility in a state for each player, based on the player's utility times type of the game (obtained by probability distribution over games).

The system by applying the presented algorithms build the whole model's states. It computes the set of states in which a formula holds, and checks whether or not a formula holds in the whole model. Furthermore, it returns states where by passing through them players can obtain the greatest total expected utility.

4.5 Remarks

We have implemented a model checking system that verifies a Bayesian game's properties in a game model. The system constructs a game's model based on the syntax and semantics of epistemic logic. The model checking algorithms are polynomial in the size of the states in the model and the algorithms are linear in the size of formula which equals the number of logical connectives and modal operators.

The model checker is a proof-of-concept implementation. Computational complexity of a model checker is determined by the size of the given model and the size of the given formula to be checked against the model[89]. Previous studies [89], [38] show that the complexity of model checkers for logics such as temporal-epistemic logic or the logic of knowledge and linear time is PSPACE (polynomial space) complete. Although the logic in this thesis is different from those logics, the model checker implemented in this chapter follows the same approach in [89] and [38]. Therefore, we expect this model checker to also have the PSPACE completeness.

In the next chapter, we use the developed formal logic in chapter 3 and the model checking system from chapter 4 to verify properties in different scenarios. These scenarios are chosen since they are examples of Bayesian games from the literature.

Chapter 5

Applications

The study of multi-agent systems focuses on systems in which many intelligent agents interact with each other. The area of multi-agent systems is not only about game theory, however, there is an important discussion that game theory is a key tool in the field [100]. Although the normal form game is conceptually fundamental in game theory, game representations such as Bayesian games are also important. The reason is that normal form games are sometimes unsuitable for modelling large or realistic game-theoretic settings. In many realistic situations, agents might have private information that affects their own payoffs and other agents might have only probabilistic information about each others' private information. These situations are modelled as Bayesian games. In this chapter, we focus on examples of Bayesian games in wireless networks and cloud computing.

There are many game-theoretic approaches in addressing different forms of security and privacy problems in computer and communication networks [69]. The reason is that, there is a fundamental relationship between the decision making of agents and network security problems. Independent decision makers, e.g., devices or softwares, can be cooperative, selfish, or malicious. The behaviour of these agents can be modelled by game-theoretic approaches. Consequently, we can avoid inadequate stability points and design security mechanisms that coverage to the optimal possible solution.

Game theory is also addressed in different situations in cloud computing. In cloud computing, computational resources are rarely allocated completely to a user. In this area, there are conflicts of interest between cloud providers and multiple consumers using the cloud simultaneously [56]. Game theory can solve utilisation problems between a cloud provider and customers, by determining equilibrium solutions in usage of cloud resources.

Since we have the models for these two scenarios as Bayesian games, we can reason about them. It is important in these scenarios to express knowledge of agents precisely and to capture the reasoning that leads to solutions. In this chapter, the intention is to delve in enough details in each application to be able to represent them by the proposed logic in chapter 3 and to reason about them.

These two scenarios represent different application domains where Bayesian games can be used as a modelling technique. The general characteristics of these scenarios are:

- they involve more than one agent/player.
- they can be divided into independent normal form games.
- all the normal form games might be played by players simultaneously and it is not determined which games would be played. Therefore, all the games should be considered plausible under a probability distribution.

Although, in this chapter, we focus only on two specific application domains, these general characteristics might be satisfied by many other applications.

5.1 Wireless Network and Game Theoretic Approach

Wireless networks cover a wide spectrum of architectures such as wireless metropolitan mesh networks, sensor networks and ad hoc networks [117]. Collaboration in these networks improves the connectivity [102]. For example with an ad hoc network, the number of sensor nodes can collaboratively collect information and then collaboratively send the information back [76]. However, since in these networks collaborative nodes are responsible for all functions, such as packet forwarding, routing and network management, they are sensitive to the misbehaviour of nodes. The nodes' misbehaviours that affect these operations may range from simple selfishness or lack of collaboration due to the need for power saving to active attacks aiming at Denial of Service (DoS) [73].

While advanced cryptographic techniques can be used, the security challenges in wireless networks , e.g., DoS attacks, are not fully addressed because of the distributed nature of the networks. Hence, it is desirable that security schemes can be modelled from the nodes' perspective. Wang et al. [115] used game theory to capture and analyse the interaction between an attacker and a regular node in wireless networks. They modelled the scenario as Bayesian attacker detection games, in which nodes have imperfect information because the attacker can disguise as a regular node and the actions are hidden because of the noise and imperfect observation.

Games with incomplete information or Bayesian games such as Bayesian attacker detection games provide a natural and compelling model that enables understanding actions of players under uncertainty [100]. However, these models do not provide any mechanisms for players (nodes) to reason about different situations in these models. One solution is to use formal logics for these models. The positive aspects of logic approaches are that we can specify the properties of agents and multi-agent systems as logical axioms and theorems in the language with clear semantics. Therefore, there is no ambiguity in the specification and everything is explicit. Furthermore, properties, interrelationships and inferences are open to examination. In comparison to logic, computer programs need implementation and control aspects within. Thus, the issues, which are to be tested, can often become confusing [43].

The aim is to represent and reason about Bayesian attacker detection games using formal logics. There are several approaches to consider uncertainty in a logic that involves the quantification of uncertainty [59]. The developed formal logic for representing Bayesian games in chapter 3, used beliefs about uncertainty to model Bayesian games. The logic for Bayesian games is the extension of the epistemic logic for normal form games. We model Bayesian attacker detection games by this logic and then we use this logic for reasoning about the solution concept of these games.

5.1.1 Security in Wireless Network with Channel Uncertainty

An important concern of security in networks is jamming and eavesdropping, where communication channels may suffer from attacks. These attacks happen in both wired and wireless networks, but they are greater in wireless networks. These malicious behaviour are shown in figure 5.1 and 5.2.

Eavesdropping is a passive attack in the network, since the eavesdropper node listens to the network and captures data without interacting with the network. Here, we do not consider these nodes as they do not interrupt the network.

Jamming as an active attack can disrupt data transmission. An attacker can disrupt the communication of a putative victim by transmitting at the same time as the victim. In general, for a jamming attack, no special hardware is needed in order to be launched. It can be done by listening to an open medium and broadcasting in the same frequency band as the targeted network. Consequently, attacks can be done with huge success and with considerably low costs for the attacker. This attack is usually implemented at the level of physical layers by means of a high transmission power signal that corrupts a communication link or an area. The conventional solution for physical jamming can be too energy consuming to be deployed in resource constrained sensors. In addition, these attacks can also occur at the access layer. In this layer, attacker might corrupt control packets or reverse the channel for the maximum allowable number of slots. Therefore, other nodes experience low throughput by not being able to access the channel.

Figure 5.1: An eavesdropper can passively listen to the communication.

The common approach to detect attackers is based on external detection if the transmission of the attacker nodes are fixed and known [68]. Malicious behaviour in wireless networks can be modelled as game models by relating attackers with a different type of a utility function. The utility function represents gain at the expense of performance degradation of other nodes in the network. An attacker node has a conflicting interest with other nodes and is attempting to minimise their utility. In [98], conditions have been obtained

under which the type of nodes' identities should be concealed or revealed to improve the cooperative nodes' performance or to reduce the performance of attackers. In practice, attacker nodes would rather conceal their intention, i.e., if nodes may have incomplete information regarding the types of other nodes. We study the effects of such incomplete information as Bayesian games.

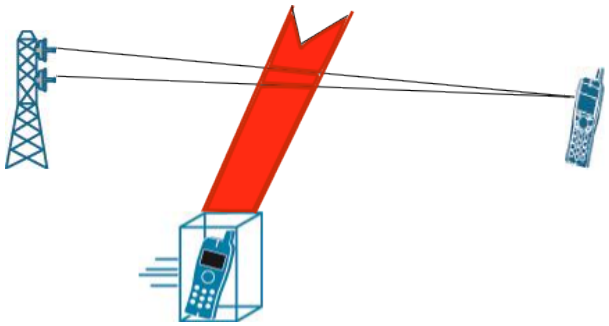

Figure 5.2: A jammer actively transmits signals to inference and interrupt the communication.

5.1.2 Bayesian Attacker Detection Game

Attacker detection games are proposed by Wang et al [115]. In this section, we briefly summarise this game and its analysis. These games model the interaction between the potential attacker node i and the regular node j. The regular node j is unable to tell by default if node i is an attacker or not, instead it can only detect the attacker through observations. Nodes can have different types, and these types are their private information.

		Node j $\theta_j = 0$			
		Monitor		Idle	
Node i $\theta i_1 = 1$	Attack	$-u_A - c_A$	$-u_A - u_M$	$u_A - c_A$	$-u_A$
	Cooperate	$-u_C$	$-u_M$	$-u_C$	0

		Node j $\theta_j = 0$			
		Monitor		Idle	
Node i $\theta i_2 = 0$	Cooperate	$-u_C$	$-u_M$	$-u_C$	0

Table 5.1: Payoff matrix of an attacker detection game [115]

Here, as the game has two nodes (players), we have two types θi_n for node i and θj_n for node j. While $\theta j_n = 0$, i.e., always regular, θi_n can be either 1 (attacker) or 0 (regular), depending on its true type. Because node j has only one type, we omit the subscript and refer to node j's type as θj.

This game is a Bayesian game because the type of node i is hidden and the observation is usually inaccurate due to noise. The strategies s_i of node i are based on its type. For $\theta_{i_1} = 1$, s_i={Cooperate} that is, the only strategy available to a regular node is cooperation. For $\theta i_2 = 0$, s_i={Attack, Cooperate}, i.e., an attacker can camouflage itself as regular. Node j has the option to monitor or be idle regardless of whether node i is attacking or not, thus, node j has two available strategies s_j={Monitor, Idle}. Because the scenario is modelled based on the game theory approach, we need a payoff matrix. For this purpose, the following values are assumed. u_A is considered as the payoff of an attacker node if it successfully attacks. The cost associated with such an attack is c_A. The cost of monitoring is u_M for the regular node j and 0 if it is idle. Therefore, for the strategy profile (s_i, s_j)=(Attack, Idle), the net utility for a successful attacker i is $u_A - c_A$, the loss for node j is $-u_A$ due to the attack. Similarly, if the strategy profile is (s_i, s_j) = (Attack, Monitor), the attacker node i loses $u_A + c_A$, and the net gain for node j is $u_A - u_M$.

Nonetheless, if an attacker node chooses to cooperate, the cost is u_C. Based on the types of node i and node j and their strategies, the payoffs matrices are shown in table 5.1. Here, we also assume that $u_A > u_m > u_C > c_A$.

In this game, due to monitoring, both nodes develop knowledge about their opponents over time. Developing the knowledge is useful because it decreases the costs for both players. For the regular node, it is not optimum to monitor continuously due to the cost of monitoring. It is also not suitable for the attacker node to attack all the time because of an increased chance of detection. While node j is monitoring, it acquires a knowledge about node i on whether it is an attacker or not. This knowledge is updated over time whenever node i is observed to be an attacker. This observation is possible from the attacker node's point of view. Despite of the fact that the uncertainty of the wireless medium makes the observations inaccurate, the more often the attacker attacks, the quicker node j can develop knowledge about its attacker type. The strategies adopted by node i is only determined by the current

state of the knowledge, i.e., when the knowledge update process takes place. However, the knowledge held by node j is its private information, and node i does not have access to this information. Thus, it is important for node i to develop its own knowledge system.

In [115], formulae are developed to predict the probability distribution of each of the normal form games of this Bayesian game being played. Here we assume, the first game happens with probability 0.55 and the probability of the second game is 0.45.

Uncertainty is unavoidable in this model and in order to deal with uncertainly we need to be able to represent it and reason about it. We overcome it by applying the epistemic logic.

5.1.3 Reasoning About Bayesian Attacker Detection Games by Epistemic Logic

Due to the uncertainty in Bayesian games, a Bayesian game is modelled as a set of games that differ only in their payoffs, and a common prior defined over them. For Bayesian games the counterpart of the Nash equilibrium is called the Bayes-Nash equilibrium. This equilibrium for agent i is a mixed strategy profile which is the best response to a mixed strategy profile of the other player. The Bayes-Nash equilibrium may seem conceptually complicated. The solution, however, is to construct a normal form representation that corresponds to a given Bayesian game. This representation is called an induced normal form. We now reason why we transformed the Bayesian games to their induced normal forms.

For attacker detection games with two players (attacker and regular nodes) if the current state is a member of the equivalence states (the states that are connected by an accessibility relation in the Kripke models) that the attacker node considers all equivalence states as possible states. It also might consider many possibilities what the equivalence states of the regular node might be. Thus, the attacker node must take into account what the regular node is likely to do in all of these circumstances, but the regular nodes choice depends on the states, which it considers possible. It may also consider a state possible that is not in the actual equivalence states of the attacker node. We therefore have to take into account all reachable states, that is (because our

		Node j			
		Monitor		Idle	
Node i	attack, cooperate	p$(-u_A - c_A)$+(1-p)$(-u_C)$	p$(u_A - u_M)$+(1-p)$(-u_M)$	p$(u_A - c_A)$+(1-p)$(-u_C)$	p$(-u_A)$
	cooperate, cooperate	$-u_C$	$-u_M$	$-u_C$	0

Table 5.2: The induced normal form of Bayesian attacker detection games

model is connected) every state in the model. This means that the attacker node has to know what the regular node might do in any of the states in the model, independent from regular node's actual state, and vice versa for the attacker node. This explains why strategies are formulated as contingencies for every state in the model, i.e., as functions from every state to a choice of strategies in that state. These strategies and their expected payoffs define normal induced games for Bayesian games, such as the attacker detection games described here. Payoffs are computed by taking the average over all states in the model. It is clear that it does not suffice to look only in the current state, as each agent also might consider other states possible. One solution might be to compute a players payoff by taking the average over all the states that that agent considers possible, i.e., that agents equivalence class. However, we cannot apply this solution, because the strategic game must be common knowledge, in order for solution concepts such as the Nash equilibrium to make sense.

Here, for the game from table 5.1 we have two players and two games, to model the scenario based on the logic from 3.3, we consider the attacker node as **1** and the regular node as **2**. The players do not know which game is about to be played. Therefore, we represent the game as an induced normal form game. The assumption is that $\theta i_1 = 1$ with probability p and $\theta i_2 = 0$ with probability 1-p. For the first normal game the logical notation for the attacker node's strategy profile is $(\mathbf{1}_{co}, \mathbf{1}_{att})$ with 'co' and 'att' abbreviating cooperate and attack, respectively. Note that in the Bayesian game the attacker node has three possible pure strategies. These pure strategies are derived from the two types and the two actions of the player. Then the attacker's three strategies in a Bayesian game can be labelled "the first action in first type" $(att\theta_{i1})$ "the second action in first type" $(co\theta_{i1})$ and "the second action in second type" $(co\theta_{i2})$. The logical notations respectively are $\mathbf{1}_{att_{\theta_{11}}}$, $\mathbf{1}_{co_{\theta_{11}}}$ and $\mathbf{1}_{co_{\theta_{12}}}$.

The regular node has only one type and two pure strategies $\mathbf{2}_{monitor}$ and $\mathbf{2}_{idle}$. Now we have a 2×2 normal form game in which the utilities are the ex-ante expected utilities in the individual games, given the agents' common prior belief. The ex-ante utility is an expected utility in which players know nothing about the other players actual type. The payoff matrix for this attacker detection game is constructed, which is the induced normal form of

this game (table 5.2).

We assume that $u_A > u_m > u_C > c_A$ then we have $u_A - c_A > u_A - u_M > 0 > -u_C > -u_M > -u_A > -u_A - c_A$. We now specify the axioms for the attacker detection game:

- $\mathbf{1}_{co_{\theta_{11}}} \vee \mathbf{1}_{att_{\theta_{11}}} \vee \mathbf{1}_{co_{\theta_{12}}}$
- $\mathbf{2}_{monitor} \vee \mathbf{2}_{idle}$

The formulae above mean that each node should choose one strategy at each state.

- $(\mathbf{1}_{co_{\theta_{11}}} \rightarrow \neg(\mathbf{1}_{att_{\theta_{11}}} \vee \mathbf{1}_{co_{\theta_{12}}})) \wedge (\mathbf{1}_{att_{\theta_{11}}} \rightarrow \neg(\mathbf{1}_{co_{\theta_{11}}} \vee \mathbf{1}_{co_{\theta_{12}}})) \wedge (\mathbf{1}_{co_{\theta_{12}}} \rightarrow \neg(\mathbf{1}_{co_{\theta_{11}}} \vee \mathbf{1}_{att_{\theta_{11}}}))$
- $\neg(\mathbf{2}_{monitor} \wedge \mathbf{2}_{idle})$

These formulae say that each node can not choose more than one strategy at each state

- $K_1 \mathbf{1}_{co_{\theta_{11}}} \leftrightarrow \mathbf{1}_{co_{\theta_{11}}}$

It says that attacker node knows its own strategies which here is cooperation with type θ_{11}.

- $u_1(\mathbf{1}_{co}, \mathbf{2}_{monitor}, \theta_{11}) = -u_A - c_A \rightarrow$
  $K_1 u_1(\mathbf{1}_{co}, \mathbf{2}_{monitor}, \theta_{11}) = -u_A - c_A$

  It says that attacker node knows its own utility at this state.

- $\mathbf{r}_{1,1_{co},\mathbf{2}_{monitor},\theta_{11}} \succeq \mathbf{r}_{1,1_{att},\mathbf{2}_{monitor},\theta_{11}}$

It shows that the ordering of strategies are complete.

- $\mathbf{rationaltype}_1 \leftrightarrow K_1((\mathbf{P}_1(\theta_{11})\mathbf{u}_1(\mathbf{1}_{att}, \mathbf{2}_{monitor}) \wedge \mathbf{P}_1(\theta_{12})\mathbf{u}_1(\mathbf{1}_{co}, \mathbf{2}_{monitor})) = \mathbf{r}_{1,1_{att},1_{co},\mathbf{2}_{monitor},\theta_{11},\theta_2} \wedge (\mathbf{P}_1(\theta_{11})\mathbf{u}_1(\mathbf{1}_{co}, \mathbf{2}_{monitor}) \wedge \mathbf{P}_1(\theta_2)\mathbf{u}_1(\mathbf{1}_{co}, \mathbf{2}_{monitor})) = \mathbf{r}_{1,1_{co},1_{co},\mathbf{2}_{monitor},\theta_{11},\theta_2} \wedge (\mathbf{1}_{co_{\theta_{11}}} \wedge \mathbf{1}_{co_{\theta_2}})) \rightarrow \mathbf{p}_{1_{co\theta_1}} \wedge \mathbf{p}_{1_{co\theta_0}} \mathbf{r}_{1,1_{co},1_{co},\mathbf{2}_{monitor},\theta_{11},\theta_2} \geq \mathbf{p}_{1_{att\theta_{11}}} \wedge \mathbf{p}_{1_{co\theta_2}} \mathbf{r}_{1,1_{att},1_{co},\mathbf{2}_{monitor},\theta_{11},\theta_2}$

The axiom set above states that player *1* is a utility maximiser whenever, player 1 decides to play $(\mathbf{1}_{co_{\theta_{11}}} \wedge \mathbf{1}_{co_{\theta_{12}}})$ or $(\mathbf{1}_{att_{\theta_{11}}} \wedge \mathbf{1}_{co_{\theta_{12}}})$ strategy in a situation that it has probabilistic beliefs $\mathbf{P}_1(\theta_{11}) \wedge \mathbf{P}_1(\theta_{12})$ about utility then the $(\mathbf{1}_{co_{\theta_{11}}} \wedge \mathbf{1}_{co_{\theta_{12}}})$ strategy is better than the other, given his beliefs. This pattern of reasoning can be continued to find the best response for both nodes. The final result is the Bayes-Nash equilibrium.

Now we can have an axiom for the detection of attack.

$$detect\_attack \leftrightarrow K_2(\mathbf{P}_1(\theta_{12})\mathbf{1}_{co_{\theta_0}}) \wedge \mathbf{P}_1(\theta_{11})\mathbf{1}_{att_{\theta_{11}}}) \wedge \neg\mathbf{2}_{idle}$$

This axiom says that the regular node can detect the attacker node when it develops knowledge about strategies and forms a probabilistic belief about the performance of the attacker node and when it is not idle. The axiom is valid because the regular node develops the knowledge about the strategies of the attacker node and its types, inferring that in any state that the regular node considers possible and the accessibility relation is related to equivalence states, given that the regular node's strategy is not idle, the regular node can detect attacks.

Since we have the model for detecting attackers in a wireless network as a Bayesian game and we represent the game by epistemic logic, we can check properties of this scenario with the model checker introduced in chapter 4. The set of states of this scenario is shown in figure 5.3. These are the presentation of states which are generated by the system. The system receives the input file as presented in figure 5.4 and generates the corresponding states.

Game played as a Bayesian game is presented with alphabet letters as utility functions (figure 5.2). For the purpose of model checking, we need to provide the system with numeric information. Therefore, we assume $u_A = 10$,$u_m = 7$, $u_C = 5$, and $c_A = 3$. Consequently, we have $u_A - c_A = 7$, $u_A - u_M = 3$, $-u_C = -5$,$-u_M = -7$,$-u_A = -10$, $-u_A - c_A = -13$, and $-u_A - u_M = -17$.

The result from the system is state 6. The reason is, that if node i cooperated in the first game, it is rational and knows that node j is rational too, in the second game node i gains more when choosing to cooperate again.

In this section, we have transformed the Bayesian attacker detection games into the induced normal form and then modelled the game based on

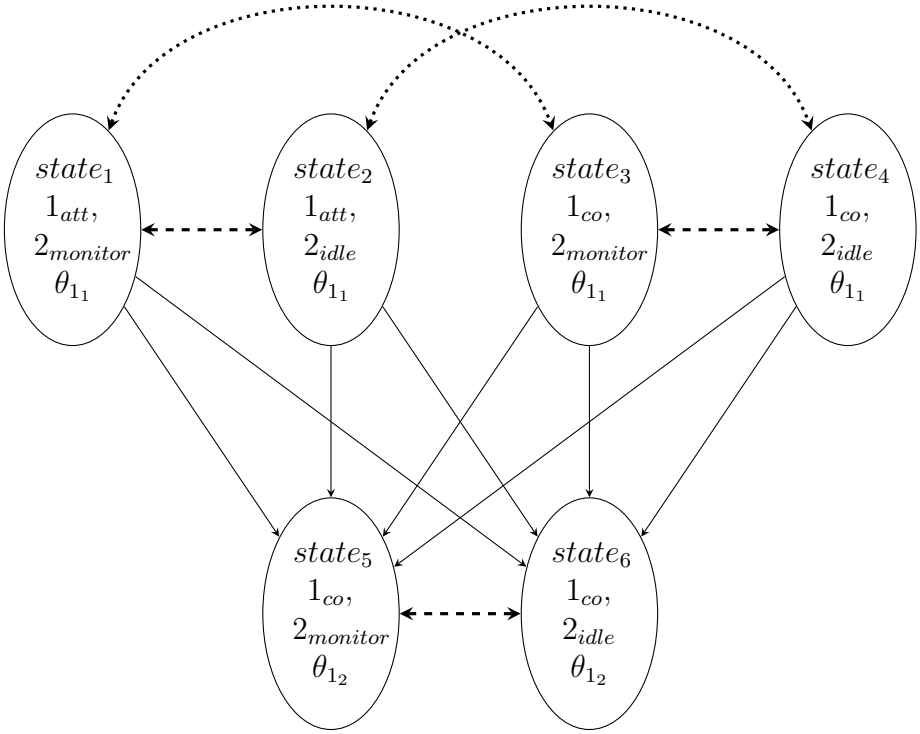

Figure 5.3: Representing relations between states of the attacker detection game of a Bayesian game

the epistemic logic developed in section 3.3. We show that if the nodes are rational they try to maximise their own utilities. Furthermore, we showed in which state the regular node can detect the attacker. Finally, we modelled the scenario by the model checker. The system we proposed is a straightforward application to verify different situations in attacker detection games.

The next section is another example that we investigated, with the aim to model the situation as a Bayesian game and represented it by epistemic logic.

5.2 Cloud Computing

The term cloud computing implies computing performed on centralised facilities provided by third-parties for compute and storage utilities [106]. Clients use cloud computing as an alternative resource of computing infrastructures, easy accessibility and on-demand usage. Cloud providers rent their resources to multiple clients concurrently and charge them depending on the amount

$(?1(att)\&?2(monitor))\&(?1(p = 0.55)\&?2(p = 0.55))\&(?1(u = -13)\&?2(u = -17))$

$(?1(att)\&?2(idle))\&(?1(p = 0.55)\&?2(p = 0.55))\&(?1(u = 7)\&?2(u = -10))$

$(?1(co)\&?2(monitor))\&(?1(p = 0.55)\&?2(p = 0.55))\&(?1(u = -5)\&?2(u = -7))$

$(?1(co)\&?2(idle))\&(?1(p = 0.55)\&?2(p = 0.55))\&(?1(u = -5)\&?2(u = 0))$

$(?1(co)\&?2(monitor))\&(?1(p = 0.45)\&?2(p = 0.45))\&(?1(u = -5)\&?2(u = -7))$

$(?1(co)\&?2(idle))\&(?1(p = 0.45)\&?2(p = 0.45))\&(?1(u = -5)\&?2(u = 0))$

$P : (?1(co)\&?2(monitor))\&(?1(rational)\&?2(rational))$

Figure 5.4: The system input format

of resources used by the customers. The current pricing strategies are quite preliminary [56], as these strategies do not consider the consequence of clients sharing the cloud's resources. Although the isolation of a client's usage of the cloud is guaranteed, a client's job can take longer to run when the cloud is heavily loaded. In [64], the authors have proposed a game-theoretical approach to analyse the characteristics of benefits of cloud computing for clients and providers. In this section, we model the utilisation of cloud services as Bayesian games, in which clients have imperfect information because there are different load capacities when using the cloud.

Incomplete information games which also called Bayesian games offer a natural model that helps to understand actions of players under uncertainty. However, these models do not provide any structure for players (clients and providers) to reason about different situations in these models. These models can be precisely modelled using formal logic. Therefore, the properties of agents and multi-agent systems are represented as logical axioms and theorems in the language with clear semantics. This advantage of formal logic helps to present the specification of models without ambiguity. Then we can examine properties, interrelationships and inferences of these games using reasoning techniques which can be automated. Thus, the issues which are to be tested and/or verified, can have determined results [43]. Modal logics also enable automated verification by model checking specifications. To verify a model

of a system, the model should be constructed, and tested that this model satisfies a formula specifying the system.

We model cloud computing as Bayesian games due to the uncertainty about different load capacities. We represent and reason about the utilisation of cloud computing with formal logics, apply the epistemic logic from chapter 3 and model the scenario by the model checker from chapter 4. An example is provided to verify game's property by this system.

5.2.1 Cloud Computing Characteristics

A company-owned data centre is costly in regards to equipment and operation. The main costs would be categorised as the IT, networking, facilities capital expenses and operating expenses such as architectural and engineering fees. To assess true total costs of building, owning, operating and maintaining a company-owned data centre (one of the most financially concentrated assets) is complicated [62]. Cloud service is an alternative to such data centres. There are different cloud services available such as lower-level services that are famous as Infrastructure-as-a-Service (IaaS) while higher level services are called Platform-as-a-Service (Paas). In this work we focus on IaaS cloud provides. The utilisation rate of cloud computing is technically a complicated economy to scale[45]. It might be based on server utilisation (CPU) and in practice it is rare to constantly fully utilise available server capacities [60]. Therefore, there are always compromises between resource over-utilisation and under-utilisation. Over-utilisation causes inadequate service providence and results in negative financial performance.

Based on the graph used by [5] the cost impact of over-utilisation and under-utilisation is losing customers. Therefore to develop a cloud, one should considers the workload and by this consideration the assumption of constant price is insufficient. In [64], the reason for the previous claim is clarified. If the cloud provider chooses a price that is less than the cost of an owned data centre, the demand for cloud will stay constant. If the price of cloud is higher than the cost of the owned data centre at full workload the owned data centre is preferred. We can define break-even workload at which the suggested price and the cost of the data centre is equal. The break-even property can be considered as a decreasing function of price. For a client with a workload

higher than the break-even workload, it is not suitable to use the cloud at the given price. Therefore, a price higher than the break-even price makes the cloud more expensive compared to the data centre option. Based on the setting above, a provider should offer different prices not only based on the cost of maintaining the cloud but also for different workloads. When the price is offered to the customer less than the break-even price, the customer might use the cloud and build a data centre when the price is higher than the break-even price. The other option is that the client combines the use of cloud and data centre depending on different workloads.

The proposed game-theoretic approach by [64] tries to model cloud computing as a suitable alternative to a company-owned data centre. This study provides a more analytical perspective that leads to better understanding of the financial aspects, e.g., knowledge may prevent a wrong decision on both

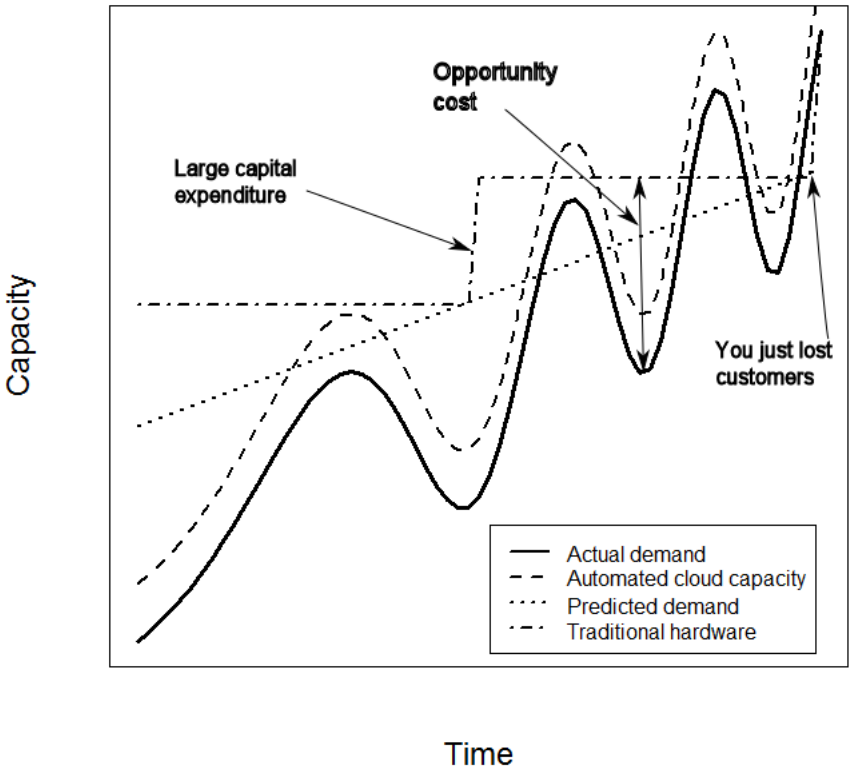

Figure 5.5: The capacity versus utilisation curve [86]

the client and the provider sides. We model the combination of cloud and owned data centre as Bayesian games. The recommended action for the client based on different load capacities is the solution of these games.

5.2.2 Cloud Computing as Bayesian Games

A game for using the cloud for processing is introduced in [64]. The game has two players, the client and the cloud provider. The cloud provider has the strategy to offer different prices based on different criteria such as network hardware cost, maintenance cost and increasing revenue. The client can build its own data centre or use the cloud. The question for the client is whether to use the cloud or not. This question can be interpreted for the provider as is to make the best decision for offering the prices. This game can be played under different load profiles which model the capacity in use. Some capacity is always in use (base load) and some capacity is idle at times (peak load) [64].

		player 2	
		build data centre	use the cloud
player 1	price1=22 ct/h	0,-1.29	0.23,-0.23
	price2=44 ct/h	0,-1.29	0.45,-0.45
	price3=66 ct/h	0,-1.29	0.66,-0.66

Table 5.3: Payoff matrix based on 2% peak load capacity $\theta = 0$ [64]

The payoffs shown in table 5.3 and table 5.4 are based on the case study discussed in [64]. There are a pair of numbers in each cell, where the first number in the pair is the provider's profit and the second number is the client's cost. From the client's perspective, some situations are disproportionally expensive to self provide, such as when the total capacity demand exceeds base loads. The reason is that costs are only amortised over the time during when the necessary capacity is actually used. The solution is to build a smaller data centre to meet base load and buy instances from the cloud to meet peak demand. Therefore, we can have different games with the same strategy and different payoffs. As the provider does not know which action a client might play, the provider might consider all the possible situations. The client plays all the games at the same time. These games are different

based on demanding different load capacities, and the provider offers different prices for different load capacities.

		player 2	
		build data centre	use the cloud
player 1	price1=22 ct/h	0,-1.29	0.46,-0.46
	price2=44 ct/h	0,-1.29	0.91,-0.91
	price3=66 ct/h	0,-1.29	1.36,-1.36

Table 5.4: Payoff matrix based on 50% peak load capacity $\theta = 1$ [64]

In [64] they propose formulas for client and provider payoffs. They show that there is a subgame perfect Nash equilibrium with the client combining building a data centre and using the cloud. Another way of solving the problem of using the cloud or not, is to model it as a Bayesian game, which means to play all these games at the same time with some probability.

To represent the uncertainty in Bayesian games, a Bayesian game is regarded as a set of games that have the same number of players and strategies with different payoffs, and a common prior defined over them. As mentioned before, the counterpart of the Nash equilibrium for Bayesian games is called the Bayes-Nash equilibrium. This equilibrium is defined for agent i as a mixed strategy profile which is the best response to a mixed strategy profile of the other player. Although, the Bayes-Nash equilibrium of a given Bayesian game is conceptually complicated to compute, one solution is to develop a normal form representation for the game.

Because we have only two games (based on the case study), we assume the probability of 2% peak load capacity game happening is equal to that of 50% peak load capacity game happening. This representation is called an induced normal form. The next step is to build an induced normal form game for this Bayesian game.

The matrix of payoff the induced normal game (table 5.5) is obtained by considering the probability of the game in table 5.3 happening is equal to 1/2 and the probability of the game in table 5.4 happening is equal to 1/2. The payoff matrix shows that if the provider proposes price2 in game 1 (table 5.3) and price 1 in game 2 (table 5.4) and if the client in game 1 chooses to build the data centre and in game 2 uses the cloud services, the provider gets 0.23 and the client pays 0.875. The Nash equilibrium for the Bayesian

		player2			
		build build	build use	use build	use use
player1	price1 price1	0,-1.29	0.23,-0.875	0.115,-0.758	0.345,-0.345
	price1 price2	0,-1.29	0.445,-1.100	0.115,-0.758	0.56,-0.56
	price1 price3	0,-1.29	0.68,-1.325	0.115,-0.758	0.795,-0.795
	price2 price1	0,-1.29	0.23,-0.875	0.225,-0.870	0.455,-0.455
	price2 price2	0,-1.29	0.445,-1.100	0.225,-0.870	0.670,-0.670
	price2 price3	0,-1.29	0.68,-1.325	0.225,-0.870	0.905,-0.905
	price3 price1	0,-1.29	0.23,-0.875	0.33,-0.975	0.56,-0.56
	price3 price2	0,-1.29	0.445,-1.100	0.33,-0.975	0.775,-0.775
	price3 price3	0,-1.29	0.68,-1.325	0.33,-0.975	1.01,-1.01

Table 5.5: Induced normal form

game is recognisable from its induced normal form. As an example, if the provider offers price 2 and price 3 in game 1 and game 2, respectively, the Nash equilibrium for the client is to use the cloud in game 1 and build a data centre in game 2.

This is the Bayesian game approach to model whether to use the cloud or not. In the next section we propose an epistemic logic approach to model the situation and reason about the solution.

5.2.3 Epistemic Logic for Cloud Computing as Bayesian Games

In this section, we present a formal language for expressing cloud utilisation benefits. We consider the problem of specifying the criteria of using the cloud for a client in regards to the price and load capacities. In order to achieve this, we need a formal language in which we may express rules of using the cloud. This could be a set of rules that determines the optimum strategies under different criteria. This language can capture different load capacity, is abstract and does not dependent on any specific condition. As in section 5.2.2 we have shown that using the cloud can be modelled as a Bayesian game, this formal language should be able to be used to specify the rules of Bayesian games.

The example, rules for playing this game are:

1. The client should choose one action for each peak load and the provider

should offer one price.

2. The provider knows the price and the client knows his actions.
3. The client and the provider know their own payoffs.
4. The strategy of the client is rational if it maximises the client's expected payoff.

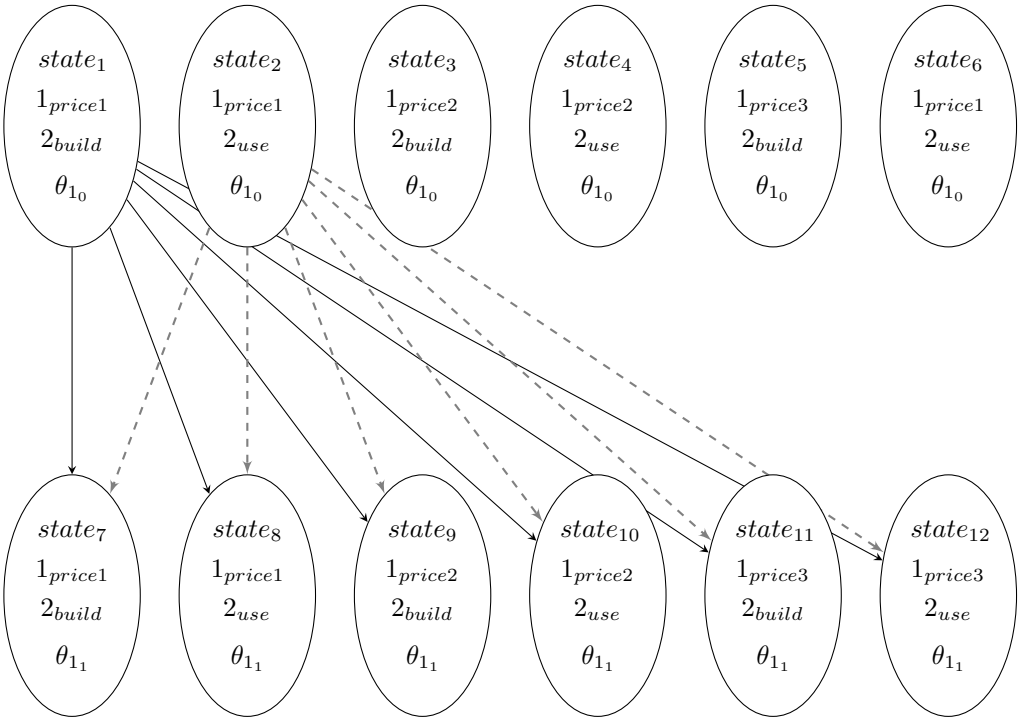

Figure 5.6: States of the game

This game can be modelled by the state transition diagram shown in figure 5.6. The diagram is regarded as a Kripke structure which can be used to provide semantics for the epistemic logic, which is a kind of modal logic [26]. From in total 12 states (5.6) the first six states are the different states of game table 5.3 and the rest of the states are the states of game table 5.4. The symbol θ in the states is the type of players based on different load capacity. For differentiating between the various types, we use a subscript for θ, where θ_0 means game 1(table 5.3) and θ_1 means game 2(table 5.4). In this scenario, we consider the type for each normal form game. Therefore, we omit the subscript of θ for each player.

We develop a language for representing Bayesian games based on a particular vocabulary. In figure 5.6 we used the propositional symbol 1_{price1} meaning the provider (player 1) offers *price1*. To find the game solution for a Bayesian game we need to model the induced normal form game. Figure 5.6 models the induced normal form game table 5.5 in which players, their actions and different games are more straightforward to recognise compared to those properties in matrix payoff induced normal form game. The lines

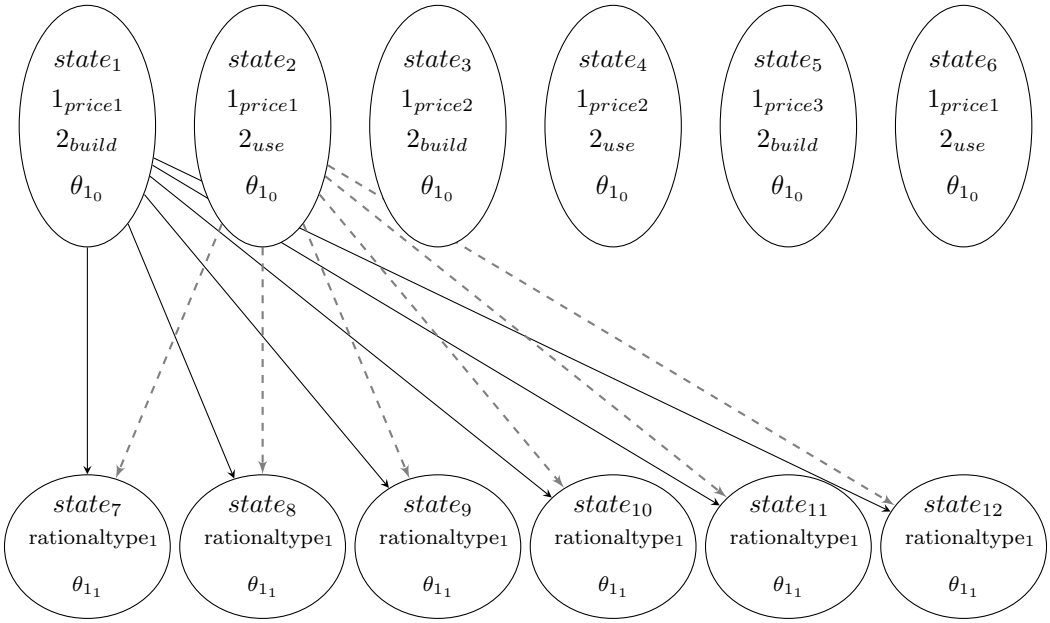

Figure 5.7: Knowledge about rationality

between the states represent reachable states. To keep the figures readable, we only drew connecting lines for to state 1 and 2 with no different meaning of solid or dash lines. The reachable states mean that if the client and the provider are in state 1 they can choose for the next game to play any of the states 7 to 12. It should be considered that the client knows that the provider is rational and also the provider knows that the client is rational. The propositional symbols $\mathbf{rationaltype}_1$ and $\mathbf{rationaltype}_2$ capture the rationality of the client and the provider, respectively. The knowledge that the client knows the provider is rational is shown in figure 5.7. It is clear that the states 7 to 12 represent the same knowledge and $\mathbf{rationaltype}_1$ is true in all of them. Therefore, the states 7 to 12 for the client in state 1 are all equally possible, which is considered as the knowledge of the client. In this

section, we intuitively apply the epistemic logic to represent the game, and in the section 5.2.4 we use the formal logic for this purpose.

5.2.4 Representing and Reasoning About Cloud Computing by Epistemic Logic

In order to illustrate the syntax and examine the expressivity of the language, we present some rules for playing the game using the formal language:

1. The client should choose exactly one action for each peak load and the provider should offer exactly one price.

   - $(\mathbf{1}_{price1_{\theta_0}} \rightarrow \neg(\mathbf{1}_{price2_{\theta_0}} \vee \mathbf{1}_{price3_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_1}} \vee \mathbf{1}_{price2_{\theta_1}} \vee \mathbf{1}_{price3_{\theta_1}})) \wedge$
     $(\mathbf{1}_{price2_{\theta_0}} \rightarrow \neg(\mathbf{1}_{price1_{\theta_0}} \vee \mathbf{1}_{price3_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_1}} \vee \mathbf{1}_{price2_{\theta_1}} \vee \mathbf{1}_{price3_{\theta_1}})) \wedge$
     $(\mathbf{1}_{price3_{\theta_0}} \rightarrow \neg(\mathbf{1}_{price2_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_1}} \vee \mathbf{1}_{price2_{\theta_1}} \vee \mathbf{1}_{price3_{\theta_1}})) \wedge$
     $(\mathbf{1}_{price1_{\theta_1}} \rightarrow \neg(\mathbf{1}_{price2_{\theta_0}} \vee \mathbf{1}_{price3_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_0}} \vee \mathbf{1}_{price2_{\theta_1}} \vee \mathbf{1}_{price3_{\theta_1}})) \wedge$
     $(\mathbf{1}_{price2_{\theta_1}} \rightarrow \neg(\mathbf{1}_{price2_{\theta_0}} \vee \mathbf{1}_{price3_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_1}} \vee \mathbf{1}_{price3_{\theta_1}})) \wedge$
     $(\mathbf{1}_{price3_{\theta_1}} \rightarrow \neg(\mathbf{1}_{price2_{\theta_0}} \vee \mathbf{1}_{price3_{\theta_0}} \vee \mathbf{1}_{price1_{\theta_0}} \vee \mathbf{1}_{price2_{\theta_1}} \vee \mathbf{1}_{price1_{\theta_1}}))$
   - $(\mathbf{2}_{build_{\theta_0}} \rightarrow \neg(\mathbf{2}_{build_{\theta_1}} \vee \mathbf{2}_{use_{\theta_0}} \vee \mathbf{2}_{use_{\theta_1}})) \wedge (\mathbf{2}_{build_{\theta_1}} \rightarrow \neg(\mathbf{2}_{build_{\theta_0}} \vee \mathbf{2}_{use_{\theta_0}} \vee \mathbf{2}_{use_{\theta_1}})) \wedge (\mathbf{2}_{use_{\theta_0}} \rightarrow \neg(\mathbf{2}_{build_{\theta_1}} \vee \mathbf{2}_{build_{\theta_0}} \vee \mathbf{2}_{use_{\theta_1}})) \wedge (\mathbf{2}_{use_{\theta_1}} \rightarrow \neg(\mathbf{2}_{build_{\theta_1}} \vee \mathbf{2}_{build_{\theta_0}} \vee \mathbf{2}_{use_{\theta_0}}))$

2. The provider knows the price, and also the client knows his actions.

   - $K_1\mathbf{1}_{price1_{\theta_1}} \leftrightarrow \mathbf{1}_{price1_{\theta_1}}$
   - $K_2\mathbf{2}_{build_{\theta_1}} \leftrightarrow \mathbf{2}_{build_{\theta_1}}$

3. The client and the provider know their own payoffs.

   - $u_1(\mathbf{1}_{price1}, \mathbf{2}_{use}, \theta_0) = \mathbf{r}_{1,1_{price1},2_{use},\theta_0} \rightarrow K_1(u_1(\mathbf{1}_{price1}, \mathbf{2}_{use}, \theta_0) = \mathbf{r}_{1,1_{price1},2_{use},\theta_0})$

4. The strategy of the client is rational if it maximises the client's expected payoff.

- $\mathbf{rationaltype}_2 \leftrightarrow K_2((\mathbf{P}_2(\theta_0)\mathbf{u}_2(\mathbf{1}_{ptice1}, \mathbf{2}_{build}) \wedge \mathbf{P}_2(\theta_1)\mathbf{u}_2(\mathbf{1}_{price2}, \mathbf{2}_{build}))$
  $= \mathbf{r}_{2,1_{price1},1_{price2},2_{build},2_{build},\theta_0,\theta_1}) \wedge$
  $(\mathbf{P}_2(\theta_0)\mathbf{u}_2(\mathbf{1}_{ptice1}, \mathbf{2}_{build}) \wedge \mathbf{P}_2(\theta_1)\mathbf{u}_2(\mathbf{1}_{price2}, \mathbf{2}_{use}))$
  $= \mathbf{r}_{2,1_{price1},1_{price2},2_{build},2_{use},\theta_0,\theta_1}) \wedge$
  $(\mathbf{P}_2(\theta_0)\mathbf{u}_2(\mathbf{1}_{ptice1}, \mathbf{2}_{use}) \wedge \mathbf{P}_2(\theta_1)\mathbf{u}_2(\mathbf{1}_{price2}, \mathbf{2}_{build}))$
  $= \mathbf{r}_{2,1_{price1},1_{price2},2_{use},2_{build},\theta_0,\theta_1}) \wedge$
  $(\mathbf{P}_2(\theta_0)\mathbf{u}_2(\mathbf{1}_{ptice1}, \mathbf{2}_{use}) \wedge \mathbf{P}_2(\theta_1)\mathbf{u}_2(\mathbf{1}_{price2}, \mathbf{2}_{use}))$
  $= \mathbf{r}_{2,1_{price1},1_{price2},2_{use},2_{use},\theta_0,\theta_1}) \rightarrow$
  $\mathbf{p}_{2_{use\theta_0}} \wedge \mathbf{p}_{2_{use\theta_1}} \mathbf{r}_{2,1_{price1},1_{price2},2_{use},2_{use},\theta_0,\theta_1}$

The set of rules are complete and by following them we have the interaction between the client and the cloud provider based on the game rules. In the next section we show that we can also use the formal language as a query language to verify the solution of the game.

The property of these models, we are interested in checking is, if the provider proposes different prices what the client should do. We assume that the provider knows that the client is rational, which can be written as $K_1($ $\mathbf{rationaltype}_{2_{\theta_1}})$. Furthermore, the provider offers price 1 in game 1 which is $\mathbf{1}_{price1_{\theta_0}}$ and price 3 in game 2 which is $\mathbf{1}_{price3_{\theta_1}}$. Based on these information, the rational client should use the cloud service in game 1 and build the data centre in game 2. Such a formula can be directly written as:

$$(K_1(\mathbf{rationaltype}_{2_{\theta_1}}) \wedge (\mathbf{1}_{price1_{\theta_0}} \wedge \mathbf{1}_{price3_{\theta_1}})) \rightarrow (\mathbf{2}_{use_{\theta_0}} \wedge \mathbf{2}_{build_{\theta_1}}) \tag{5.1}$$

If this formula holds in the model, it is also the Nash equilibrium of the subgame.

To investigate the application described above, we have used the model checker from chapter 4.

The input of this application is shown in figure 5.8. Each state of cloud computing scenario is determined by a formula in which the player, the player's strategy and the type of the game are mentioned. For example the notations in the first line of the input:

$(?1(price1)\&?2(use))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0.23)\&?2(u=-0.23))$

$(?1(price1)\&?2(build))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0)\&?2(u=-1.29))$

$(?1(price2)\&?2(use))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0.45)\&?2(u=-0.45))$

$(?1(price2)\&?2(build))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0)\&?2(u=-1.29))$

$(?1(price3)\&?2(use))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0.66)\&?2(u=-0.66))$

$(?1(price3)\&?2(build))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0)\&?2(u=-1.29))$

$(?1(price1)\&?2(use))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0.46)\&?2(u=-0.46))$

$(?1(price1)\&?2(build))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0)\&?2(u=-1.29))$

$(?1(price2)\&?2(use))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0.91)\&?2(u=-0.91))$

$(?1(price2)\&?2(build))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0)\&?2(u=-1.29))$

$(?1(price3)\&?2(use))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=1.36)\&?2(u=-1.36))$

$(?1(price3)\&?2(build))\&(?1(p=0.5)\&?2(p=0.5))\&(?1(u=0)\&?2(u=-1.29))$

$P : (?1(price1)\&?2(use))\&(?1(rational)\&?2(rational))$

Figure 5.8: The system input format

(?1(price1) & ?2(use)) & (?1(p=0.5) & ?2(p=0.5)) & (?1(u=0.23) & ?2(u=-0.23))

are assigned to their logical form as following:

- $?1(price1)$ is the representation of $\mathbf{1}_{price1_{\theta_0}}$
- $?2(use))$ is the representation of $\mathbf{2}_{use_{\theta_0}}$
- $?1(p=0.5)$ is the representation of $\mathbf{P}_1(\theta_0)$
- $?2(p=0.5)$ is the representation of $\mathbf{P}_2(\theta_0)$
- $?1(u=0.23)$ is the representation of $\mathbf{u}_1(\mathbf{1}_{price1_{\theta_0}}, \mathbf{2}_{use_{\theta_0}})$
- $?2(u=-0.23)$ is the representation of $\mathbf{u}_2(\mathbf{1}_{price1_{\theta_0}}, \mathbf{2}_{use_{\theta_0}})$

The cloud provider and the cloud client as players are represented as ?1 and ?2, respectively. Strategies of the players are presented after the player's name in the parentheses. The first line of the input shows the strategy of each player as *price*1 for player 1 and *use* for player 2. The probability belief (type) of the game is symbolised as $(p = 0.5)$ and the utility of each player as $(u = 0.23)$.

The last line of figure 5.8 is the representation of the following property. This property is checked by the system.

$$((K_1(\mathbf{rationaltype}_{2_{\theta_0}})) \wedge (K_2(\mathbf{rationaltype}_{1_{\theta_0}}))) \wedge (\mathbf{1}_{price1_{\theta_0}} \wedge \mathbf{2}_{use_{\theta_0}})$$

The result for the system is state 6 which means:

$$(((K_1(\mathbf{rationaltype}_{2_{\theta_0}})) \wedge (K_2(\mathbf{rationaltype}_{1_{\theta_0}}))) \wedge (\mathbf{1}_{price1_{\theta_0}} \wedge \mathbf{2}_{use_{\theta_0}})) \rightarrow (\mathbf{1}_{price1_{\theta_1}} \wedge \mathbf{2}_{use_{\theta_1}}) \tag{5.2}$$

Based on table 5.5, equation 5.2 is the correct move. In this state the client should invest 0.345. If the cloud provider offers any price in the second game and the client decides to build its own data centre, it will lose 0.758. However, the client has to invest more if the client decides to use the cloud and the cloud provider offers a different price rather than *price*1.

As we mentioned in chapter 4 we can verify different properties in a Bayesian games. In the above formula we provide information about each player and also the type of each player. The result was a state that determined the strategies of players with alternative types. We can define new properties by changing the strategies and types of each player.

Another set of properties that we can check has the template equal to equation 5.1. We need to provide information about one player but in all its types. For example, for the cloud provider we determine the strategy of its first type as *price*1 in type θ_{1_1} and *price*3 in type θ_{1_2}. The result from

the model checker is the states that have the same strategies for the cloud provider in each of its type. As a result, we are able to find the best move of the client, from these states. By changing parameters, such as players or strategies, we are able to check different properties of the scenario.

5.3 Remarks

In this chapter, we have studied two different examples of Bayesian games. As shown in this chapter, both scenarios can be modelled as Bayesian games.

Because of an increasing number of applications of wireless networks with collaborative nodes, the security of these networks has been receiving an increasing attention among researchers in recent years. However, little has been done so far in terms of the definition of security needs specific to different types of scenarios that can be defined for wireless networks. One approach is to model attacker detection with uncertainty of node types as a Bayesian game in the game theoretic scenario.

As another application, we have adopted a game-theoretic approach with uncertainty to analyse the interaction between a cloud client and a cloud provider. We have modelled this interaction with respect to different load capacities as Bayesian games.

However, the conceptual study of Bayesian game-playing situations cannot be used to derive stable results as long as no appropriate formalism is available to model the situation. The main purpose of this chapter was to show that a formal tool, namely epistemic logic for normal form games can be used to represent and reason about Bayesian games. We have shown that this language provides reasoning about the solution as pure Nash equilibrium for the normal node in wireless network and the client of the cloud in Bayesian games. By using the language for representing and reasoning about Bayesian attacker detection games and cloud computing games, two representative examples of the application of this language are provided. Although we have shown the use of the extended language to verify some specifications of Bayesian games such as the solution concept, these verifications for these games can also be performed through model checking. The inputs to a model checker are the description of a system to be analysed and a number of

properties, often expressed as formulae of one kind of logic. We have modelled and verified desired properties for both scenarios.

Chapter 6

Conclusions

In this thesis, we have proposed a theory of practical reasoning which was driven by two contemporary paradigms: instrumental rationality from game theory, and epistemic reasoning from philosophical logic and computer science. This provides a unified theory of rational agencies, which is a theory of how agents deliberate when they take into account the demands of instrumental rationality, their background of future-directed desires and the information they have about the rationality and information of others. The approach is not limited to two player games, but can be applied to multi-player systems.

We conclude the thesis by summarising its contributions and several promising future research directions.

6.1 Discussion

We have shown in Chapter 2 that a broad perspective can account for modelling game settings, mainly, normal form games and Bayesian games by means of modal logic, because this formal approach provides a natural way to represent games. We have explained the different approaches to detect solution concepts in these games, as these approaches are the foundations for studying interactions of agents in multi-agent systems.

In Chapter 3, we have shown that an epistemic modal logic for normal form games can have an axiom for agents' preferences. Consequently, it is more precise to reason about rationality in these game settings. An agent is rational if it chooses a particular strategy while believing that this strategy is at least as good as other alternative strategies and it considers possible that the chosen one is actually better than the other strategies. The notion of rationality corresponds to the detection of solution concepts in games. Therefore, by

reasoning about rational behaviour, we analyse solution concepts in games. With the help of the epistemic language for normal form games, we were able to develop an epistemic approach to specifying Bayesian games. We have explicitly represented agents' knowledge about payoffs, preferences, other players and uncertainty beliefs over a set of games. Furthermore, we have studied how rational agents try to opt their expected utilities and act rational, which is a synonym to achieve solution concepts such as the Nash equilibrium in games. Finally, we have outlined axiomatic proof systems for the epistemic logic, which gives it an explicit representation of practical reasoning in games with uncertainty.

Model checking is a technique to establish the correctness of a system. In contrast to testing, model checking tools look at all possible behaviours of a system. While testing can find errors only, formal verification by means of model checking can not only detect errors but also prove their absence. In Chapter 4, we have presented a verification technique for multi-agent systems to verify game properties, such as rules of games and solution concepts. The system performs model checking of logics with modalities for both, the knowledge of agents in the system and the probability belief of agents. We have used the proposed language in Chapter 3 as a specification language for the system which allowed us to verify the rational behaviour of agents and we found this property valuable in many applications. Because the formal language is the input and specification language of the system (notwithstanding the adopted syntactic encoding of formulae), it is not necessary to translate the system to be verified to the input language of the model checker.

In Chapter 5, we connect the abstract representation schemes and models developed in the thesis to real-world applications. We have shown that the new formal approach is helpful to explicitly represent and analyse these applications. It was also helpful to better understand uncertainty in interactive settings. Two applications that we have analysed are attacker detection in wireless networks with channel uncertainty, and recognising the benefit of using cloud computing. We have tested our implementation of the model checker technique by the means of those two examples. These experimental results indicate that our formal approach is a faithful interpretation of these two challenges.

6.2 Future Work

In game theory, the term Bayesian coalitional games [54] is used to specify games with cooperation and uncertainty in movements. These games describe optimal actions for rational agents, which are artificial computational entities. However, these descriptions do not consider the reasoning abilities of agents. One of the possible future directions is to develop a formal language that will directly and transparently support reasoning about Bayesian coalitional games. This task could be satisfied by breaking Bayesian coalitional games into some sub games. The Bayesian coalitional games have two main components of Bayesian games, and coalitional games. This breakdown would enable us to apply our developed formal language, in combination with coalitional logic [1], to specify and analyse Bayesian coalitional games explicitly.

In this thesis, we consider games in which, under fairly general conditions, a universal space of epistemic types exists. Its elements are sequences of probability measures, corresponding to progressively higher order beliefs. Thus, essentially any statement about players' reciprocal beliefs has a representation in the universal space. However, in [11], it is proposed that this type can be constructed by considering a space whose elements are sequences of collections of conditional probabilities. Therefore, the elements of the universal space are actually infinite hierarchies of conditional probability systems. This framework applies to situations where agents hold interacting beliefs conditional on a fixed collection of hypotheses about the prevailing external state. Considering these interacting beliefs in our formal language appears worthy as by keeping the history, an agent could update its beliefs. It might enable us to specify scenarios that agents' beliefs change based on receiving new information.

Besides abstracting and specifying the behaviour of complex systems by means of formalisms based on logic, researchers have been concerned with the problem of verifying systems. As we have implemented a system for model checking Bayesian games, we could extend our implementation by relying on a symbolic data structure, which would enable us instead of a single state representation to represent it as sets of states. The concept behind symbolic representation is the exploitation of regularity and structures in models. This results in highly compressed representation of the state-space. An example of

a symbolic data structure is ordered binary decision diagrams (OBDD) which can be used in the verification of epistemic and correct behaviour modalities in systems. OBDDs as data structure for representing boolean functions can be created and manipulated using CUDD library [101]. This package provides the benefits of binary decision diagrams but by reordering, it reduces the size of the decision diagrams. This facility optimises the performance of the model checker tool.

Another future agenda could be coping with infinity that arises from loop unfolding and from recursive data structures. Strategies must be defined so that in most practical cases the analysis by the model checker tool is able to produce an answer.

Finally, in the future the complexity of the model checker should be investigated. As mentioned in chapter 4, there are several studies reported in the literature on the complexity of model checkers for logics such as temporal-epistemic logic or the logic of knowledge and linear time [89] [38]. They show that in the best case scenario the computational complexity of model checking for those logics is PSPACE-complete (that is, the model checking algorithm requires polynomial space). Therefore we also expect that the complexity of this model checker would be PSPACE-complete. The same techniques used in those previous studies can also be used here in the complexity study.

Bibliography

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. On the logic of coalitional games. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 153–160, New York, NY, USA, 2006. ACM.

[2] T. Ågotnes and H. P. van Ditmarsch. What will they say? - public announcement games. *Synthese*, 179(Supplement-1):57–85, 2011.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, Sept. 2002.

[4] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran. Mocha: Modularity in model checking. In *Proceedings of the 10th International Conference on Computer Aided Verification*, CAV '98, pages 521–525, London, UK, 1998. Springer-Verlag.

[5] Amazon web services. `http://aws.amazon.com/`. Accessed: 2014-01-13.

[6] R. Aumann and A. Brandenburger. Epistemic conditions for nash equilibrium. *Econometrica*, 63(5):pp. 1161–1180, 1995.

[7] R. J. Aumann. Agreeing to disagree. *The Annals of Statistics*, 4(6):1236–1239, 1976.

[8] R. J. Aumann. Interactive epistemology i: Knowledge. *International Journal of Game Theory*, 28(3):263–300, 1999.

[9] A. Baltag. A logic for suspicious players: Epistemic actions and belief-updates in games, 2002.

[10] A. Baltag, S. Smets, and J. Zvesper. Keep hoping for rationality: a solution to the backward induction paradox. *Synthese*, 169(2):301–333, 2009.

[11] P. Battigalli and M. Siniscalchi. Hierarchies of conditional beliefs and interactive epistemology in dynamic games. *Journal of Economic Theory*, 88(1):188–230, 1999.

[12] P. Battigalli and M. Siniscalchi. Interactive epistemology in games with payoff uncertainty. *Research in Economics*, 61(4):165–184, December 2007.

[13] K. Binmore. *Fun and Games: A Text on Game Theory*. D. C. Heath Canada, Limited, 1992.

[14] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2002.

[15] O. Board. Dynamic interactive epistemology. *Games and Economic Behavior*, 49(1):49–80, Oct. 2004.

[16] G. Bonanno. Modal logic and game theory: two alternative approaches. *Risk, Decision and Policy*, 7:309–324, 12 2002.

[17] G. Bonanno. A syntactic characterization of perfect recall in extensive games. *Research in Economics*, 57(3):201 – 217, 2003. Logic and the Foundations of the Theory of Games and Decisions.

[18] G. Bonanno. Two lectures on the epistemic foundations of game theory. Working Papers 72, University of California, Davis, Department of Economics, Feb 2007.

[19] G. Bonanno. A syntactic approach to rationality in games with ordinal payoffs. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory*, LOFT 7, pages 59–86. Amsterdam University Press, 2008.

[20] R. Bordini, M. Fisher, M. Wooldridge, and W. Visser. Model checking rational agents. *Intelligent Systems, IEEE*, 19(5):46 – 52, sept.-oct. 2004.

[21] I. Boureanu, A. V. Jones, and A. Lomuscio. Automatic verification of epistemic specifications under convergent equational theories. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, pages 1141–1148, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[22] J. Bradfield. Introduction to modal and temporal mu-calculi. In L. Brim, M. Kretnsk, A. Kucera, and P. Jancar, editors, *CONCUR 2002 Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 98–98. Springer Berlin Heidelberg, 2002.

[23] J. Broersen. Ctl.stit: Enhancing atl to express important multi-agent system verification properties. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 683–690, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[24] F. Buccafurri, T. Eiter, G. Gottlob, and N. Leone. Enhancing model checking in verification by AI techniques. *Artificial Intelligence*, 112(12):57 – 104, 1999.

[25] N. Bulling, W. Jamroga, and J. Dix. Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):51–114, 2008.

[26] B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[27] J. Chen and S. Micali. Mechanism design with set-theoretic beliefs. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 87–96, 2011.

[28] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[29] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, Apr. 1986.

[30] T. Clausing. Belief revision in games of perfect information. *Economics and Philosophy*, 20(01):89–115, 2004.

[31] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, pages 82–90, New York, NY, USA, 2006. ACM.

[32] B. de Bruin. *Explaining Games: The Epistemic Programme in Game Theory*. Synthese Library. Springer, 2010.

[33] O. Dianat and M. A. Orgun. Model checking bayesian cloud resource utilisation game with epistemic logic. Submitted on October 31, 2013.

[34] O. Dianat and M. A. Orgun. Modelling bayesian attacker detection game in wireless networks with epistemic logic. In *CollaborateCom*, pages 210–215, 2012.

[35] O. Dianat and M. A. Orgun. Representing and reasoning about utilization of cloud computing as bayesian games with epistemic logic. In *ANT/SEIT*, pages 40–47, 2013.

[36] H. v. Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[37] E. A. Emerson. Handbook of theoretical computer science (vol. b). chapter Temporal and Modal Logic, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[38] K. Engelhardt, P. Gammie, and R. van der Meyden. Model checking knowledge and linear time: Pspace cases. In S. Artemov and A. Nerode, editors, *Logical Foundations of Computer Science*, volume 4514 of *Lecture Notes in Computer Science*, pages 195–211. Springer Berlin Heidelberg, 2007.

[39] R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. *J. ACM*, 41(2):340–367, Mar. 1994.

[40] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[41] M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic*, 4(3):279 – 304, 2006.

[42] O. Friedmann, M. Latte, and M. Lange. A decision procedure for ctl* based on tableaux and automata. In J. Giesl and R. Hhnle, editors, *Automated Reasoning*, volume 6173 of *Lecture Notes in Computer Science*, pages 331–345. Springer Berlin Heidelberg, 2010.

[43] J. R. Galliers. *A theoretical framework for computer models of cooperative dialogue, acknowledging multiagent conflict*. PhD thesis, 1988. AAIDX87541.

[44] P. Gammie and R. van der Meyden. Mck: Model checking the logic of knowledge. In *Computer Aided Verification*, pages 256–259. 2004.

[45] J. Hamilton. Cloud computing economics of scale, 2010.

[46] P. Harrenstein, J.-J. Meyer, W. van der Hoek, and C. Witteveen. A modal characterization of nash equilibrium. *Fundam. Inf.*, 57(2-4):281–321, Feb. 2003.

[47] J. C. Harsanyi. Games with incomplete information played by bayesian players, i- iii. the basic probability distribution of the game. *MANAGEMENT SCIENCE*, 14(7):486–502, 1968.

[48] A. Heifetz and P. Mongin. Probability logic for type spaces. *Games and Economic Behavior*, 35(12):31 – 53, 2001.

[49] T. Henzinger, R. Majumdar, F. Mang, and J.-F. Raskin. Abstract interpretation of game properties. In J. Palsberg, editor, *Static Analysis*, volume 1824 of *Lecture Notes in Computer Science*, pages 220–239. Springer Berlin Heidelberg, 2000.

[50] J. Hintikka. *Knowledge and Belief.* Ithaca, N.Y.,Cornell University Press, 1962.

[51] W. V. D. Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002*, pages 1167–1174. ACM Press, 2002.

[52] X. Huang, C. Luo, and R. van der Meyden. Symbolic model checking of probabilistic knowledge. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK XIII, pages 177–186, New York, NY, USA, 2011. ACM.

[53] S. Ieong. *Cooperation in Competition: Efficiently Representing and Reasoning About Coalitional Games.* PhD thesis, Stanford, CA, USA, 2008. AAI3313818.

[54] S. Ieong and Y. Shoham. Bayesian coalitional games. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 1*, AAAI'08, pages 95–100. AAAI Press, 2008.

[55] M. Ishihata, Y. Kameya, and T. Sato. Variational bayes inference for logic-based probabilistic models on bdds. In S. Muggleton, A. Tamaddoni-Nezhad, and F. Lisi, editors, *Inductive Logic Programming*, volume 7207 of *Lecture Notes in Computer Science*, pages 189–203. Springer Berlin Heidelberg, 2012.

[56] V. Jalaparti, G. D. Nguyen, I. Gupta, and M. Caesar. Cloud resource allocation games, 2010.

[57] W. Jamroga. Strategic planning through model checking of atl formulae. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L. Zadeh, editors, *Artificial Intelligence and Soft Computing - ICAISC 2004*, volume 3070 of *Lecture Notes in Computer Science*, pages 879–884. Springer Berlin Heidelberg, 2004.

[58] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundam. Inform.*, 63(2-3):185–219, 2004.

[59] A. Jøsang. Probabilistic logic under uncertainty. In *Proceedings of the thirteenth Australasian symposium on Theory of computing - Volume*

65, CATS '07, pages 101–110, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[60] M. Klems, J. Nimis, and S. Tai. Do clouds compute? a framework for estimating the value of cloud computing. In C. Weinhardt, S. Luckner, and J. Stößer, editors, *Designing E-Business Systems. Markets, Services, and Networks*, volume 22 of *Lecture Notes in Business Information Processing*, pages 110–123. Springer Berlin Heidelberg, 2009.

[61] B. P. Kooi. Probabilistic dynamic epistemic logic. *J. of Logic, Lang. and Inf.*, 12(4):381–408, Sept. 2003.

[62] J. Koomey. a simple model for determining true total cost of ownership for data centers, 2007.

[63] S. Kripke. Semantical Considerations on Modal Logic. *Acta Phil. Fennica*, 16:83–94, 1963.

[64] J. Künsemöller and H. Karl. A game-theoretical approach to the benefits of cloud computing. In *Proceedings of the 8th international conference on Economics of Grids, Clouds, Systems, and Services*, GECON'11, pages 148–160, 2012.

[65] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, volume 2324 of *Lecture Notes in Computer Science*, chapter 13, pages 113–140. Springer Berlin / Heidelberg, Berlin, Heidelberg, Apr. 2002.

[66] H. E. Kyburg, Jr. *Uncertainty logics*, pages 397–438. Oxford University Press, Inc., New York, NY, USA, 1994.

[67] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction (Synthesis Lectures on Artificial Intelligence and Machine Learning)*. Morgan and Claypool Publishers, 1 edition, June 2008.

[68] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1307–1315, 2007.

[69] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux. Game theory meets network security and privacy. *ACM Comput. Surv.*, 45(3):25:1–25:39, July 2013.

[70] `http://cgi.cse.unsw.edu.au/~mck/pmck`. Accessed: 2014-01-13.

[71] S. Merz. Modeling and verification of parallel processes. chapter Model checking: a tutorial overview, pages 3–38. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[72] R. Meyden and N. Shilov. Model checking knowledge and time in systems with perfect recall. In C. Rangan, V. Raman, and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 432–445. Springer Berlin Heidelberg, 1999.

[73] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation. In *in Mobile Ad Hoc Networks. Communication and Multimedia Security*, 2002.

[74] `http://www.cis.upenn.edu/~mocha`. Accessed: 2014-01-13.

[75] H. Moulin. *The strategy of social choice*. Advanced textbooks in economics. North-Holland Publishing Company, 1983.

[76] H. Ochiai, P. Mitran, H. Poor, and V. Tarokh. Collaborative beamforming for distributed wireless ad hoc sensor networks. *Signal Processing, IEEE Transactions on*, 53(11):4110 – 4124, nov. 2005.

[77] C. H. Papadimitriou. Games, algorithms, and the internet. In *WWW*, pages 5–6, 2011.

[78] P. Parikh. Situations, rules, and conventional meaning: Some uses of games of partial information. *Journal of Pragmatics*, 39(5):917 – 933, 2007. ¡ce:title¿Focus-on Issue: Formal and Philosophical Aspects of Pragmatics¡/ce:title¿.

[79] R. Parikh and R. Ramanujam. Distributed processes and the logic of knowledge. In R. Parikh, editor, *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 256–268. Springer Berlin Heidelberg, 1985.

[80] T. Parr. *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Bookshelf, 2007.

[81] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[82] M. Pauly. On the complexity of coalitional reasoning. *IGTR*, 4(3):237–254, 2002.

[83] M. Pauly and R. Parikh. Game logic - an overview. *Studia Logica*, 75(2):165–182, 2003.

[84] C. Pecheur and F. Raimondi. Symbolic model checking of logics with actions. In S. Edelkamp and A. Lomuscio, editors, *Model Checking and Artificial Intelligence*, volume 4428 of *Lecture Notes in Computer Science*, pages 113–128. Springer Berlin Heidelberg, 2007.

[85] A. Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57, 1977.

[86] `http://www.opengroup.org/cloud/cloud/roi.htm`. Accessed: 2014-01-13.

[87] Prism. `http://www.prismmodelchecker.org/`. Accessed: 2014-01-13.

[88] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation, 2004.

[89] F. Raimondi and A. Lomuscio. The complexity of symbolic model checking temporal-epistemic logics. In *In Proceedings of Concurrency, Specification Programming (CSP), Ruciane-Nida*, 2005.

[90] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. *Journal of Applied Logic*, 5(2):235–251, 2007.

[91] R. Ramanujam and S. Simon. Dynamic logic on normal form games. In *Pre-proceedings of the KR2008-workshop on Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS), Sydney, September 2008*, page 140, 2008.

[92] O. Roy. *Thinking before acting: intentions, logic, rational choice*. ILLC Dissertation Series. Institute for Logic, Language and Computation, 2008.

[93] O. Roy. Epistemic logic and the foundations of decision and game theory. *Journal of the Indian Council of Philosophical Research*, 27(2):283–314, 2010.

[94] J. Ruan, W. van der Hoek, and M. Wooldridge. Model Checking GDL through MOCHA: A Case Study. Technical report, University of Liverpool, 2009.

[95] J. Ruan, W. van der Hoek, and M. Wooldridge. Verification of games in the game description language. *J. Log. and Comput.*, 19:1127–1156, December 2009.

[96] A. Rubinstein. *Modeling Bounded Rationality*, volume 1 of *MIT Press Books*. The MIT Press, August 1997.

[97] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1995.

[98] Y. Sagduyu, R. Berry, and A. Ephremides. Mac games for distributed wireless network security with incomplete information of selfish and malicious user types. In *Game Theory for Networks, 2009. GameNets '09. International Conference on*, pages 130–139, 2009.

[99] P.-Y. Schobbens. Alternating-time logic with imperfect recall. *Electr. Notes Theor. Comput. Sci.*, 85(2):82–93, 2004.

[100] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, NY, USA, 2008.

[101] F. Somenzi. CUDD: BDD package, University of Colorado, Boulder. `http://vlsi.colorado.edu/~fabio/CUDD/`. Accessed: 2014-01-13.

[102] S. Song, D. Goeckel, and D. Towsley. Collaboration improves the connectivity of wireless networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 –11, april 2006.

[103] `http://spinroot.com/spin/whatispin.html`. Accessed: 2014-01-13.

[104] R. Stalnaker. On the evaluation of solution concepts. *Theory and Decision*, 37(1):49–73, 1994.

[105] R. Stalnaker. Knowledge, belief and counterfactual reasoning in games. *Economics and Philosophy*, 12(02):133–163, 1996.

[106] F. Teng and F. Magoules. A new game theoretical resource allocation algorithm for cloud computing. In P. Bellavista, R.-S. Chang, H.-C. Chao, S.-F. Lin, and P. Sloot, editors, *Advances in Grid and Pervasive Computing*, volume 6104, pages 321–330. Springer Berlin Heidelberg, 2010.

[107] J. Van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[108] J. van Benthem. Rational dynamics and epistemic logic in games. *IGTR*, 9(1):13–45, 2007.

[109] J. van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.

[110] J. van Benthem and A. Gheerbrant. Game solution, epistemic dynamics and fixed-point logics. *Fundam. Inform.*, 100(1-4):19–41, 2010.

[111] J. van Benthem, S. Ghosh, and F. Liu. Modelling simultaneous games with concurrent dynamic logic. In J. van Benthem, S. Ju, and F. Veltman, editors, *A Meeting of the Minds-Proceedings of the Workshop on Logic, Rationality and Interaction*, Beijing, 2007.

[112] W. van der Hoek and M. Pauly. 20 modal logic for games and information. In J. V. B. Patrick Blackburn and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 1077 – 1148. Elsevier, 2007.

[113] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*, pages 280–291, 2004.

[114] R. J. Wallace. The Stanford Encyclopedia of Philosophy, 2003.

[115] W. Wang, M. Chatterjee, and K. Kwiat. Attacker detection game in wireless networks with channel uncertainty. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1 –5, may 2010.

[116] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[117] C. Westphal. A study of the percolation threshold for k-collaborative wireless networks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1 –6, june 2009.

[118] M. Wooldridge. Logic for automated mechanism design and analysis. In R. Bergmann, G. Lindemann, S. Kirn, and M. Pechoucek, editors, *Multiagent System Technologies*, volume 5244 of *Lecture Notes in Computer Science*, pages 1–1. Springer Berlin Heidelberg, 2008.

[119] M. Wooldridge and W. Hoek. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. In F. Arbab and C. Talcott, editors, *Coordination Models and Languages*, volume 2315 of *Lecture Notes in Computer Science*, pages 4–4. Springer Berlin Heidelberg, 2002.

[120] S. Zamir. Bayesian games: Games with incomplete information. In R. A. Meyers, editor, *Computational Complexity*, pages 238–253. Springer New York, 2012.