

# Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems



# Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems

Stefano V. Albrecht[a], Peter Stone[b]

[a]*The University of Edinburgh, United Kingdom*
[b]*The University of Texas at Austin, United States*

**Abstract**

Much research in artificial intelligence is concerned with the development of autonomous agents that can interact effectively with other agents. An important aspect of such agents is the ability to reason about the behaviours of other agents, by constructing *models* which make predictions about various properties of interest (such as actions, goals, beliefs) of the modelled agents. A variety of modelling approaches now exist which vary widely in their methodology and underlying assumptions, catering to the needs of the different sub-communities within which they were developed and reflecting the different practical uses for which they are intended. The purpose of the present article is to provide a comprehensive survey of the salient modelling methods which can be found in the literature. The article concludes with a discussion of open problems which may form the basis for fruitful future research.

*Keywords:* autonomous agents, multiagent systems, modelling other agents, opponent modelling

**Contents**

## 1. Introduction

A core area of research in modern artificial intelligence (AI) is the development of autonomous agents that can interact effectively with other agents. An important aspect of such agents is the ability to reason about the behaviours, goals, and beliefs of the other agents. This reasoning takes place by constructing *models* of the other agents. In general, a model is a function which takes as input some portion of the observed interaction history, and returns a prediction of some property of interest regarding the modelled agent (cf. Figure 1). The interaction history may contain information such as the past actions that the modelled agent took in various situations. Properties of interest could be the future actions of the modelled agent, what class of behaviour it belongs to (e.g. "defensive", "aggressive"), or its current goals and plans.

An autonomous agent can utilise such a model in different ways, but arguably the most important one is to inform its decision making. For example, if the model makes predictions about the actions of the modelled agent[1], then the modelling agent can incorporate those predictions in its planning procedure to optimise its interaction with the modelled agent. If instead the model makes predictions about the class of behaviour of the modelled agent, then the modelling agent could choose a precomputed strategy which it knows to work well against the predicted class. Besides informing decisions, an agent model can also be used for other purposes. For example, an intelligent tutoring system could use a model of a specific human player in games such as Chess to identify and point out weaknesses in the human's play (Iida et al., 1996).

[1] We will use the term "modelling agent" to refer to the agent which is carrying out the modelling task, and "modelled agent" or "other agent" to refer to the agent which is being modelled.

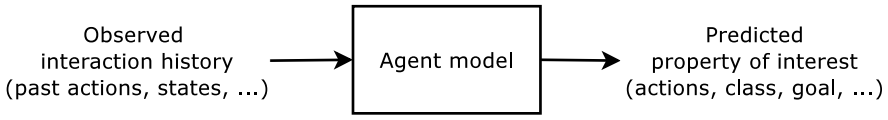


Figure 1: General agent model.

The process of constructing models of other agents, sometimes referred to as *agent modelling* or *opponent modelling*,[2] often involves some form of learning since the model may be based on information observed from the current interaction and possibly data collected in past interactions. For example, an agent may model another agent's decision making as a deterministic finite automaton and learn the parameters of the automaton (e.g. nodes, edges, labels) during the interaction (Carmel and Markovitch, 1998b). Similarly, an agent may attempt to classify the strategy of another agent by using classifiers which were trained with statistical machine learning on data collected from recorded interactions (Weber and Mateas, 2009).

Modelling other agents in complex domains is a challenging task. In the above example in which an agent models another agent's behaviour as a finite automaton, the learning task is known to be NP-complete in both the exact and approximate cases (Pitt, 1989; Gold, 1978). Many other modelling techniques exist, each with their own complexity issues. For example, the task of inferring an agent's goals and plans based on complex action hierarchies often faces an exponential growth in plan hypotheses (Geib, 2004). Yet, despite such difficulties, research in modelling other agents continues to push the boundary, in part driven by innovative applications that necessitate effective modelling capabilities in agents. For example, dialogue systems have to understand and disambiguate the intentions and plans of users (Grosz and Sidner, 1986; Litman and Allen, 1984); intelligent tutor systems must reason about the knowledge and misconceptions of students to facilitate learning progress (McCalla et al., 2000; Anderson et al., 1990); autonomous military and security systems must be able to reason about the decision making, beliefs, and goals of adversaries (Borck et al., 2015; Jarvis et al., 2005; Tambe, 1995); and autonomous vehicles must reason about the behaviours of other vehicles (Buehler et al., 2009). Beyond such applications of "narrow AI", there is also the grand vision of a general AI which is capable of completing tasks, across different domains, that potentially require non-trivial interactions with other agents (including humans). It is evident that such a general AI will require an ability to reason about the goals, beliefs, and decision making of other agents. This is especially true in the absence of coordination and communication protocols, where modelling other agents is a key requirement for effective collaboration (Stone et al., 2010; Rovatsos et al., 2003).

There is a rich history of research on computational agents that model other agents. Some of the earliest work can be traced back to the beginnings of game theory, in which opponent modelling was studied as a means of computing equilibrium solutions for games. The classical example is "fictitious play" (Brown, 1951), in which each player models the other player's strategy as the empirical frequency distribution of their past play. Another example is rational learning (Kalai and Lehrer, 1993), in which players maintain Bayesian beliefs over a space of possible strategies for the other players. In AI research and computational linguistics, methods for recognising the goals and plans of agents (Schmidt et al., 1978) were applied in automated dialogue systems to understand and disambiguate the intentions of users (Pollack, 1986; Litman and Allen, 1984).

[2]Because much of the early work was developed in the context of competitive games such as Chess, the term "opponent modelling" was established to refer to the process of modelling other agents, and is still used by many researchers.

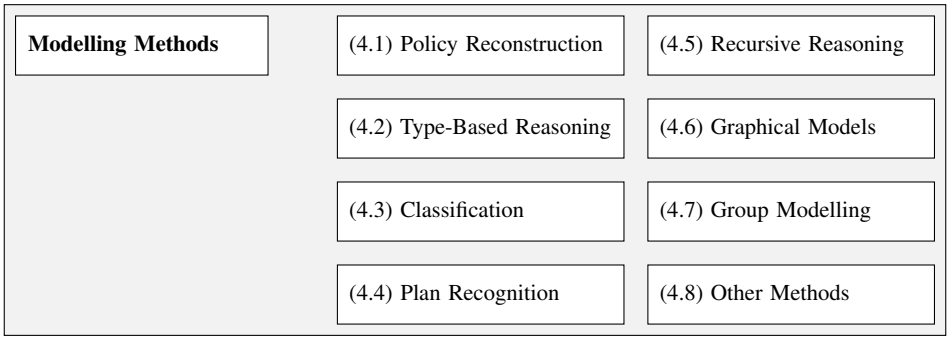


Figure 2: Surveyed modelling methods. Brackets show linked section numbers.

Adversarial games such as Chess were also an important driver of research in opponent modelling. The dominant solution for such games was based on the "minimax" principle, in which agents optimise their decisions against a worst-case, foolproof opponent (Campbell and Marsland, 1983). However, it was recognised that real players often exhibit limitations in their strategic play, e.g. due to cognitive biases or bounded computation, and that knowledge of such limitations could be exploited to obtain superior results to minimax play (Iida et al., 1994, 1993; Carmel and Markovitch, 1993; Reibman and Ballard, 1983). In addition to opponent modelling in game playing, early models of recursive reasoning ("I believe that you believe that I believe...") were formulated (e.g. Gmytrasiewicz et al., 1991; Wilks and Ballim, 1986). Since these early works in game theory and AI, the problem of modelling other agents has been an active area of research in many sub-communities, including classic game playing (Fürnkranz, 2001), computer Poker (Rubin and Watson, 2011), automated negotiation (Baarslag et al., 2016), simulated robot soccer (Kitano et al., 1997), human user modelling (Zukerman and Albrecht, 2001; McTear, 1993), human-robot interaction (Lasota et al., 2014), commercial video games (Bakkes et al., 2012), trust and reputation (Ramchurn et al., 2004), and multiagent learning (Stone and Veloso, 2000).

Many different modelling techniques now exist which vary widely in their underlying assumptions and methodology, largely due to the different needs and constraints of the sub-communities within which they were developed. Assumptions may pertain to aspects of the modelled agent, such as whether the agent makes deterministic or stochastic action choices, and whether its behaviour is fixed or may change over time. They may also pertain to aspects of the environment, such as whether the actions of other agents and environment states are observed fully or only partially with possible uncertainty. Current methodologies include learning detailed models of an agent's decision making as well as reasoning about spaces of such models; inferring an agent's goals and plans based on hierarchical action descriptions; recursive reasoning to predict an agent's state of mind and its higher-order beliefs about other agents; and many other approaches. While some articles have surveyed modelling methods specific to one of the aforementioned sub-communities (see Section 2), there is a gap in the current literature in that there is no unified survey of the principal modelling methods which can be found across the sub-communities. As a result, there has been a missed opportunity to effectively communicate ideas, results, and open problems between these sub-communities.

The purpose of the present article is to provide a comprehensive survey of methods which enable autonomous agents to model other agents, and to highlight important open problems in this area. We identify and describe seven salient modelling methods (plus other relevant methods)

which are shown in Figure 2. Works were included in the survey if a significant part of the work was concerned with the problem of modelling other agents, which in most cases included the proposal of novel algorithms and/or analysis of and experiments with existing algorithms.

After discussing related surveys in Section 2, we begin our survey in Section 3 with a discussion of the different assumptions that modelling methods may be based on, to help the reader gain an understanding of the applicability and limitations of methods. Section 4 then surveys a number of different modelling methods by discussing the general idea underlying each method and surveying the relevant literature. Section 5 concludes with a discussion of open problems which have not been sufficiently addressed in the literature, and which may be fruitful avenues for future research.

## 2. Related Surveys

Several articles survey research on opponent modelling for specific domains. Baarslag et al. (2016) provide a survey of opponent modelling in bilateral negotiation settings, in which two agents negotiate the values of one or more "issues" (e.g. cost, size, and colour of a car) in an exchange. Bakkes et al. (2012) and Karpinskyj et al. (2014) survey methods for player modelling in commercial video games, where the purpose of modelling is to improve the playing strength of game AI as well as player satisfaction. Pourmehr and Dadkhah (2012) provide an overview of modelling methods used in 2D simulated robot soccer, in which two teams of agents compete in a soccer match. Rubin and Watson (2011) survey research in Poker playing agents and dedicate a section to opponent modelling methods. Lasota et al. (2014) survey research in safe human-robot interaction and include a section on methods that predict the motions and actions of humans. Several articles survey work in trust and reputation modelling in multiagent systems (e.g. Pinyol and Sabater-Mir, 2013; Yu et al., 2013; Ramchurn et al., 2004). Other surveys of opponent modelling include van den Herik et al. (2005), Olorunleke and McCalla (2005), and Fürnkranz (2001). The above articles survey modelling methods for specific domains, and their discussions are centred on the particular properties of interest (e.g. offer preferences, team formation, action timing, human motion, trust levels) and constraints (e.g. limited computational resources, extensive form games of imperfect information, modelling from raw data) in these domains.

Our article is a general survey of the major modelling methods that can be found across the literature, including methods which are not or only sparsely addressed in the above surveys, such as type-based reasoning, plan recognition, recursive reasoning, and graphical models. In contrast, the above surveys primarily focus on specific interaction settings which differ significantly in their rules, dynamics, and assumptions, with many of the surveyed methods being domain-specific. While, ultimately, it is useful to exploit specific domain structure to achieve optimal performance, a focus on domain-specific aspects can make it difficult for researchers unfamiliar with the subject to gain an understanding of the general modelling approaches and, thus, contributes to a fragmentation of the community, as evidenced by the fact that the above surveys have little overlap in terms of cited works. Still, one can identify common ideas in methodology between these communities, such as the use of machine learning methods to "classify" other agents and the use of Bayesian beliefs to reason about the relative likelihood of alternative models. Our survey aims to distil the broader context of such methodologies and to provide an overview of the relevant works as well as discuss open problems and avenues for future research, thus documenting the state-of-the-art in agent modelling methods.

In addition to the above surveys, there are also a number of surveys on the topic of multiagent learning (Hernandez-Leal et al., 2017; Bloembergen et al., 2015; Tuyls and Weiss, 2012; Busoniu et al., 2008; Panait and Luke, 2005; Alonso et al., 2001; Stone and Veloso, 2000; Sen and Weiss,

1999). Multiagent learning[3] (MAL) is defined as the application of learning to facilitate interaction between multiple agents, where the learning is typically carried out by the individual agents or some central mechanism that has control over the agents. Modelling other agents often involves some form of learning about the other agents and can, thus, be viewed as a part of MAL. However, MAL may also involve other types of learning, such as learning to coordinate without constructing models of other agents (e.g. Albrecht and Ramamoorthy, 2012; Bowling and Veloso, 2002; Hart and Mas-Colell, 2001) and learning based on communication. Most of the cited MAL surveys provide some discussion of research on modelling other agents, but due to the broader scope the discussions are necessarily limited. Moreover, some of these surveys are somewhat dated now (albeit still useful), and miss out on much of the more recent progress in modelling methods.

A complicating factor in complex domains such as human-robot interaction, simulated robot soccer, and some commercial games is the fact that agents cannot directly observe the chosen actions of other agents, but must instead infer these (with possible uncertainty) from other observations, such as changes in the environment. The task of identifying actions from raw sensor data and changes in states is referred to as *activity recognition*, and it is itself an active research area that has produced a substantial body of work (Sukthankar et al., 2014). Methods for activity recognition are not covered in our survey. We assume that the modelling agent has some means to identify actions during the interaction, e.g. by using domain-specific heuristics as is often done in the robot soccer domain (e.g. Kaminka et al., 2002a), training an action classifier using supervised machine learning (e.g. Ledezma et al., 2009), or reasoning about the probabilities of possible observations (e.g. Panella and Gmytrasiewicz, 2017).

## 3. Assumptions in Modelling Methods

Before surveying the modelling methods, we will discuss some of their possible underlying assumptions. This discussion will be useful for appreciating the applicability and limitations of methods, as well as where some of the current open problems lie. We categorise assumptions into assumptions about the modelled agents and assumptions about the environment within which the agents interact. (For example, in a soccer game, the environment is defined by the soccer field and ball/player positions, and the game dynamics.)

The following is a list of possible assumptions about the modelled agent. To make this discussion a little more precise, we will use $P(a_j|H)$ to denote the probability with which the modelled agent $j$ chooses action $a_j$ after some history $H = \langle o^1, o^2, ..., o^t \rangle$, where $o^\tau$ is an observation at time $\tau$ and $t$ is the current time step. For example, under a fully observable setting, $o^\tau$ may include the environment state at time $\tau$ and the actions of other agents (if any) at time $\tau - 1$.

**Deterministic or stochastic action choices?** An agent makes deterministic action choices if for every history $H$, $P(a_j|H) = 1$ for some action $a_j$. The more general case are stochastic action choices, in which actions may be chosen with any probabilities.[4] Assuming deterministic action choices can greatly simplify the modelling task because we can be sure that the modelled agent will always choose the same action for a given history. This allows us to use deterministic structures such as decision trees and deterministic state automata, for which efficient learning algorithms exist. Besides simplifying the learning of models, assuming

[3] The 2017 International Joint Conference on Artificial Intelligence held a tutorial on "Multiagent Learning: Foundations and Recent Trends". Tutorial slides can be downloaded at: `http://www.cs.utexas.edu/~larg/ijcai17_tutorial`

[4] In the game theory literature, stochastic actions are often referred to as "mixed strategies" (e.g. Myerson, 1991)

deterministic action choices can also simplify the planning of our own agent's actions, because the planning does not have to account for uncertainties in the modelled agent's actions. On the other hand, such an assumption precludes the possibility that the modelled agent may randomise deliberately or that it may make mistakes, as human agents often do. Therefore, modelling methods which allow for stochasticity in action choices can facilitate more robust prediction and planning.

**Fixed or changing behaviour?** An important question in modelling methods is the degree to which the modelled agent is allowed to change its decision making. The precise meaning of change varies in the literature and also depends on the property of interest that is to be predicted (e.g. actions, class, plan). The basic notion is that the modelled agent has some ability to adapt its decision making based on its past observations. An example of a non-changing (sometimes called "fixed", "stationary", or "non-learning") agent often found in the literature is a simple "Markovian" agent which chooses its actions based only on the most recent observation and regardless of what happened before, i.e. $P(a_j|H) = P(a_j|o^t)$. In contrast, an example of an adaptive/learning agent is one which itself tries to learn models of other agents and bases its decisions on these models. Early modelling methods assumed fixed behaviours to avoid the added complexity of tracking and predicting possible changes in behaviours. Today, more methods allow for varying degrees of adaptability in order to allow for greater complexity in modelled agents.

**Decision factors known or unknown?** Agents often make decisions based on some portion of the history (e.g. the most recent $n$ observations), or based on abstract features which were calculated from the history. An example of an abstract feature is the average number of times a particular action was observed in a specific situation. Given such dependencies on factors, an important question in modelling methods is whether the relevant factors in the modelled agent's decision making are known a priori. Many methods assume that this knowledge is available, or that the relevant factors can in principle be derived from the information available in the observed history. In the worst case, the modelling method can work on the entire history and the hope is that the relevant factors are approximately reconstructed in the modelling process. However, if such a reconstruction is not possible and knowledge of relevant factors is not available, then the predictions of the resulting model can be very unreliable. Some methods attempt to solve this problem by reasoning about a space of possible relevant factors (cf. Section 4.1.1).

**Independent or correlated action choices?** If the modelling agent is interacting with more than one other agent, then a possible question is whether the other agents choose their actions independently from each other. Independence means that the joint probability $P(a_j, a_{j'}|H)$ for agents $j$ and $j'$ can be factored into $P(a_j|H)P(a_{j'}|H)$. Otherwise, the agents are said to have correlated action choices. Many modelling methods assume independent action choices, which allows for the independent construction of models for each agent. Note that independence does not mean that the agents ignore each other, since they may observe each others' past actions in the history $H$. However, if agents are correlated in their action choices, e.g. due to joint plans and communication (Stone and Veloso, 1999; Grosz and Kraus, 1996), then it may be important for the modelling method to capture such correlations. For applications in which this is the case, such as robot soccer, researchers have developed methods that model entire teams as opposed to individual agents.

**Common or conflicting goals?** Another possible assumption concerns the agents' goals.[5] A goal may be to reach a specific state in the environment or to optimise a given objective function, such as the payoff/reward functions used in game theory and reinforcement learning. Goals are said to be common if they are identical for all agents. Many modelling methods that attempt to predict an agent's actions are unaffected by the goals of the agents, since such methods primarily work on observed actions (cf. Sections 4.1 and 4.2). However, methods which attempt to predict the intentions and beliefs of other agents can be influenced significantly by assumptions about goals, since an observed action may yield different clues when viewed in the context of common versus conflicting goals. Some modelling methods attempt to learn the payoff functions used by other agents (cf. Section 4.1.4).

In addition to assumptions about the modelled agent, many methods make assumptions about the environment within which the interaction takes place. Some common assumptions concern the order in which agents choose their actions (simultaneous or alternating moves), and the representation of actions and environment states (discrete, continuous, mixed). However, the most important assumptions usually concern the extent to which agents are able to observe what is happening in the environment. Much of the early work in opponent modelling was developed in idealised settings such as Chess, in which the state of the environment and the agents' chosen actions are fully observable by all agents. The domain of Poker added the problem of partial observability of environment states, since no player can see the private cards of other players. In domains such as human-robot interaction and robot soccer, additional complications are that observations about the environment state may be unreliable (e.g. due to noisy sensors), and that actions may no longer be observed directly by the agents but have to be inferred (with some uncertainty) based on other observations, such as changes in the environment. (For example, a soccer player may infer a passing action between two players based on changes in the position, velocity, and direction of the ball.) Such partial observability can make the modelling task significantly more difficult, since agents can make decisions based on private observations and the modelling method must take such possibilities into account.

## 4. Modelling Methods

This section provides a comprehensive survey of the salient modelling methods that can be found in the literature (cf. Figure 2). Specifically, we will survey methods of policy reconstruction (Section 4.1), type-based reasoning (Section 4.2), classification (Section 4.3), plan recognition (Section 4.4), recursive reasoning (Section 4.5), graphical models (Section 4.6), group modelling (Section 4.7), and other relevant methods (Section 4.8). For each modelling method, we provide a table[6] which lists the assumptions in the surveyed papers, organised according to the dimensions identified in Section 3. Table 1 provides a high-level summary of the surveyed modelling methods.

[5] Assumptions about the goals of agents may also be viewed as assumptions about the environment, since the payoff/reward functions are usually part of the task and environment specification. We view them as assumptions about agents to allow for the more general notion of subjective goals, such as intrinsic rewards (Singh et al., 2005).

[6] See Appendix A for further clarifications on assumption tables.

| Method | Summary |
|---|---|
| Policy reconstruction (4.1) | Model predicts action probabilities of modelled agent. Assume specific model structure and learn model parameters based on observed actions.<br>+ Can learn arbitrary model (subject to chosen model structure)<br>+ Models often progressively generated during the interaction<br>− May require many observations to yield useful model<br>− Learning task can be complex (space/time) |
| Type-based reasoning (4.2) | Model predicts action probabilities of modelled agent. Assume agent has one of several known types and compute relative likelihood of types based on observed actions.<br>+ Types can be very general (e.g. blackbox)<br>+ Can lead to fast adaptation if true type of agent (or a similar type) is in type space<br>− Can lead to wrong predictions if type space is wrong<br>− Beliefs not expressive enough to tell if type space is wrong |
| Classification (4.3) | Model predicts class label (or real number, if regression) for modelled agent. Choose model structure and use machine learning to fit model parameters based on various information sources.<br>+ Can learn to predict various kinds of properties<br>+ Many machine learning algorithms available<br>− Learning may require large amount of data to yield useful model<br>− Model is usually computed before interaction and can be difficult to update during interaction |
| Plan recognition (4.4) | Model predicts goal and (to some extent) future actions of modelled agent. Algorithms often use hierarchical plan library or domain model.<br>+ Knowledge of goal and plan extremely useful for long-term planning<br>+ Rich plan library can encode complex plans (e.g. with temporal and applicability conditions)<br>− Specifying plan library can be tedious/impractical; may be incomplete<br>− Most methods assume modelled agent is unaware of observer ("keyhole plan recognition") |
| Recursive reasoning (4.5) | Model predicts next action of modelled agent. Recursively simulate reasoning of modelled agent ("I think that you think that I think..."). <br>+ Account for higher-order beliefs of other agents<br>− Recursion is computationally expensive<br>− Assumes modelled agent is rational |
| Graphical models (4.6) | Model predicts action probabilities of modelled agent. Uses graphical model to represent agent's decision process and preferences.<br>+ Detailed model of agent's domain conceptualisation (causal beliefs) and preferences<br>+ Graphical representation can lead to computational improvements<br>− Does not scale efficiently to sequential decision processes |
| Group modelling (4.7) | Model predicts joint properties of group of agents (e.g. joint action/goal/plan of group).<br>+ Can capture correlations in action choices of group<br>+ Can exploit group structure to improve efficiency and quality of prediction<br>− Reasoning about agent groups is highly complex due to interdependencies among agents |

Table 1: High-level summary of surveyed modelling methods, with an indication of some of their potential strengths (+) and limitations (−). This summary does not apply to all surveyed papers; many variations exist, and not all potential strengths and limitations are listed in this summary (see main text).

### *4.1.* **Policy Reconstruction**

Policy reconstruction methods generate models which make explicit predictions about an agent's actions, by reconstructing the agent's decision making. Most methods begin with some arbitrary or idealised model and "fit" the internals of the model to reflect the agent's observed behaviour. The predictions of such a model can be utilised by a planner to reason about how the modelled agent might react to various courses of actions. For example, Monte-Carlo tree search (Browne et al., 2012) can naturally integrate such models to sample possible interaction trajectories, which are used to find optimal actions with respect to the agent model.

The two central design questions in policy reconstruction methods are (1) what elements of the interaction history should be used to make predictions, and (2) how should these elements be mapped to predictions? The following discussion of methods gradually shifts emphasis from the first question to the second question.

#### *4.1.1. Conditional Action Frequencies*

The archetypal example of a policy reconstruction method is "fictitious play" (Brown, 1951), in which agents model each other as a probability distribution over their possible actions. The probabilities are "fitted" via a maximum-likelihood estimation over the agents' observed actions, which corresponds to simply computing their average frequencies. This simple method has some well-known convergence properties in matrix games (Fudenberg and Levine, 1998) and was adopted early in multiagent reinforcement learning (Claus and Boutilier, 1998). Of course, a single distribution is unable to capture agent behaviours with complex dependencies on the interaction history. The key to making this method more capable is to *condition* the action distribution on elements of the history. For instance, Sen and Arora (1997) and Banerjee and Sen (2007) propose agents that learn the action frequencies of other agents conditioned on the modelling agent's own action, and Davison and Hirsh (1998) propose a user model which learns conditional probabilities of user commands based on the user's previous command. More complex methods may condition distributions on more information from the history, such as the $n$ most recent joint actions of all agents (Powers and Shoham, 2005).

The difficulty with learning conditional action distributions is that we may not know what elements of the history to use. If we condition distributions on too little or the wrong information from the history, then the learned distributions may not produce reliable predictions. If we condition on too much information, then the learning may be too slow and inefficient. To address this issue, methods have been developed which automate the conditioning. Jensen et al. (2005) propose a method which learns action frequencies for each possible subset of the $n$ most recent elements in the history. To manage the combinatorial explosion of subsets, some subsets are removed if the entropy of their conditional distributions is above some threshold, meaning that their predictions are not certain enough. To make a prediction, the method selects the subset with the lowest entropy for the given history (i.e. most certain prediction). Similarly, Chakraborty and Stone (2014) describe a method which learns action frequencies conditioned on the most recent $n, n-1, n-2, ...$ observations and plans its own actions using the "smallest" conditioning which best predicts the modelled agent's actions, in the sense that it is not too dissimilar to the largest conditioning. Essentially the same method can be used to model agents which condition their choices on abstract feature vectors derived from the history (Chakraborty and Stone, 2013).

The idea of monitoring conditional action frequencies of the modelled agents has also been used in the context of extensive form games with imperfect information, such as Poker (Mealing and Shapiro, 2017; Ganzfried and Sandholm, 2011; Southey et al., 2005; Billings et al., 2004).

Such games are characterised by the fact that agents may have private information (e.g. cards in own hand) in addition to public information (e.g. cards on the table). Hence, agents make decisions in "information sets", which are sets of decision nodes that cannot be distinguished with the available information. The decision making of other agents can be modelled as the observed frequency with which they chose actions in the various information sets. For example, Southey et al. (2005) associate an independent Dirichlet distribution for each information set and update the corresponding distribution after each observed action. Dirichlet distributions are a natural way to model uncertainty over finite probability distributions and can be updated efficiently. Rather than learning such distributions from scratch, it is also possible to initialise the distributions to some reasonable baselines. For example, Ganzfried and Sandholm (2011) first compute a Nash equilibrium solution for the game which specifies action distributions for each information set and agent. This solution can be used to initialise the agent models. During play, the distributions in the models are gradually shifted toward the observed action frequencies of the modelled agents, to reflect their true behaviours. The advantage of this method is that the modelling agent can initially plan its actions against a rational (Nash) opponent model, rather than starting with an arbitrary model. Billings et al. (2004) propose a method which learns action frequencies conditioned on entire action sequences. To generalise observed actions more quickly, the method employs a sequence of increasingly coarse abstractions over action sequences. Moreover, to allow for changing behaviours, the method uses a decay factor such that more recent observations have greater weight in the calculation of action frequencies.

#### *4.1.2. Case-Based Reasoning*

A limitation in the above methods is that they may lack a mechanism to extrapolate (or "generalise") past observations to previously unseen situations. Abstraction methods such as those used by Billings et al. (2004) can achieve some level of generalisation by defining equivalence relations over observations. Case-based reasoning (e.g. Kolodner, 2014; Veloso, 1994; Hammond, 1986) is a related method which uses similarity functions to relate observations. In essence, this method maintains a set of "cases" along with the observed actions of the modelled agent in each encountered case. To extrapolate between cases, a similarity function must be specified which measures how similar two given cases are. For example, in simulated robot soccer, a case may be defined by the state of the soccer field, and the similarity could measure the respective differences of ball and players positions in two given cases. When presented with a new case, the method searches for the most similar known cases and predicts an action as a function of these cases.

Albrecht and Ramamoorthy (2013) propose a method which stores observed cases (defined as environment states) and the observed actions of the modelled agent in each case. When queried with a new case, the method generates a prediction by searching for similar cases and aggregating their predictions based on the relative similarity to the queried case and the recency of the observed actions to allow for changing behaviours. Similar case-based methods for modelling the behaviour of other agents were proposed by Borck et al. (2015) and Hsieh and Sun (2008). In all of the above methods, a case is represented as a multi-attribute vector and similarity between vectors is measured using domain-specific difference calculations. An interesting question in case-based methods is whether the similarity function can be optimised automatically with respect to the modelled agent (Steffens, 2005, 2004a; Ahmadi et al., 2003). For example, Steffens (2004a) proposes a method in which the similarity function is defined as a linear weighting of differences in the attributes of two given cases. The weighting is learned based on the goal of the modelled agent and a "Goal Dependency Network" which specifies dependencies between sub-goals and case attributes. Another important question in case-based methods is how to store and retrieve

cases efficiently. For example, Denzinger and Hamdan (2004) propose a retrieval method based on tree search, and Borck et al. (2015) prune cases to reduce the number of the stored cases.

#### *4.1.3. Compact Model Representations*

Methods based on frequency distributions and case-based reasoning are general, since the conditioning and cases can be based on any observable information. However, this generality comes at the cost of exponential space complexity. For example, if action distributions of the modelled agent are conditioned on the past $n$ observations which each can assume $m$ possible values (or, equivalently, if a case consists of $n$ different attributes with $m$ possible values), then there are (up to) $m^n$ distributions to be stored. An alternative method is to use more compact model representations such as those found in the machine learning literature. For example, one may attempt to model an agent's decision making as a deterministic finite automaton (DFA) (Carmel and Markovitch, 1998b, 1996c; Mor et al., 1995). Carmel and Markovitch (1996c) show how such a model can be learned from observed actions. Essentially, each time the method observes a new action, it checks if the current model is consistent with the observation in the sense that it would have predicted the action, given the current state of the DFA. If it is not, the DFA model is modified to account for the new observation, e.g. by adding new nodes and edges between nodes. A useful property of this method is that it searches for the smallest DFA that is consistent with the observations. Other representations that have been used to model agents include decision trees (Barrett et al., 2013) and artificial neural networks (Silver et al., 2016; Davidson et al., 2000).

Machine learning methods can also be used to infer missing information from the observed interaction. For example, in robot soccer an agent cannot directly observe what actions other agents took; it only observes (if at all) the changes in the environment as a result of the agents' actions. Ledezma et al. (2009) propose a method which trains multiple decision/regression tree classifiers on recordings from past plays. One classifier is trained to predict the action that the modelled agent took, given two consecutive environment states. Another classifier is trained to predict the next action that the agent will take, given the current state and past action predicted by the first classifier. Additional classifiers are trained to predict the continuous parameters of the predicted actions. Panella and Gmytrasiewicz (2017) propose to use probabilistic DFAs (PDFAs) to model the stochastic action choices of agents in domains in which neither the state of the environment nor the other agents' actions are observed with certainty. The proposed method uses a Bayesian nonparametric prior over the space of all PDFAs, and updates the prior after new observations to find a model which captures the behaviour of the modelled agent. Mealing and Shapiro (2017) use an expectation-maximisation algorithm to infer the current information set of the modelled agent in extensive form games.

#### *4.1.4. Utility Reconstruction*

One characteristic which is shared by all of the above methods is that they do not model the preferences of the modelled agent, which are often expressed as some kind of utility function. However, it can be difficult to generalise the observed actions from the modelled agent if its preferences are unknown. An alternative is to assume that the modelled agent maximises some utility function which is unknown to the modelling agent. This *rationality* assumption allows the modelling agent to reason about the possible utility function of the modelled agent, given its observed actions. Once an estimate of a utility function is obtained, one can predict the actions of the modelled agent by maximising the utility function from the perspective of the modelled agent.

Based on this idea, Carmel and Markovitch (1996b, 1993) consider opponent modelling in extensive form games (e.g. Checkers) and define a model as the search depth and utility

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Mealing and Shapiro, 2017) | yes | no | yes | yes | no | altern. | discrete | partial |
| (Panella and Gmytrasiewicz, 2017) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Silver et al., 2016) | yes | no | yes | yes | no | altern. | discrete | full |
| (Borck et al., 2015) | no | no | yes | yes | no | simult. | mixed | partial |
| (Chakraborty and Stone, 2014, 2013) | yes | no | no | yes | no | simult. | discrete | full |
| (Albrecht and Ramamoorthy, 2013) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Barrett et al., 2013) | no | no | yes | yes | yes | simult. | discrete | full |
| (Ganzfried and Sandholm, 2011) | yes | no | yes | yes | no | altern. | discrete | partial |
| (Ledezma et al., 2009) | no | no | yes | yes | no | simult. | mixed | partial |
| (Hindriks and Tykhonov, 2008) | no | no | yes | yes | no | altern. | discrete | full |
| (Banerjee and Sen, 2007) | yes | no | – | yes | no | simult. | discrete | full |
| (Powers and Shoham, 2005) | yes | no | no | yes | no | simult. | discrete | full |
| (Jensen et al., 2005) | yes | no | no | yes | no | simult. | discrete | full |
| (Southey et al., 2005) | yes | no | yes | yes | no | altern. | discrete | partial |
| (Steffens, 2005, 2004a) | no | no | no | yes | no | simult. | mixed/disc. | full |
| (Billings et al., 2004) | yes | yes | yes | yes | no | altern. | discrete | partial |
| (Coehoorn and Jennings, 2004) | no | no | yes | yes | no | altern. | discrete | full |
| (Denzinger and Hamdan, 2004) | no | no | yes | yes | no | simult. | discrete | full |
| (Gal et al., 2004) | yes | no | yes | yes | no | simult. | discrete | full |
| (Ahmadi et al., 2003) | no | no | no | yes | no | simult. | mixed/disc. | full |
| (Chajewska et al., 2001) | no | no | yes | yes | no | altern. | discrete | full |
| (Davidson et al., 2000) | yes | no | yes | yes | no | altern. | discrete | partial |
| (Claus and Boutilier, 1998) | yes | no | – | yes | yes | simult. | discrete | full |
| (Davison and Hirsh, 1998) | yes | no | – | yes | no | –* | discrete | full |
| (Sen and Arora, 1997) | yes | no | – | yes | no | simult. | discrete | full |
| (Carmel and Markovitch, 1998b, 1996c) | no | no | yes | yes | no | simult. | discrete | full |
| (Carmel and Markovitch, 1996b, 1993) | no | no | yes | yes | no | altern. | discrete | full |
| (Mor et al., 1995) | no | no | yes | yes | no | simult. | discrete | full |
| (Brown, 1951) | yes | no | – | yes | no | simult. | discrete | full |

Table 2: Assumptions in papers for policy reconstruction methods. *Does not specify move order.

function used by the opponent. The utility function is assumed to be a linear combination of features in the game state, and the goal is to learn the weights in the combination. Given a set of examples which consist of game states and the opponent's chosen action in each state, the proposed method learns multiple candidate models (one for each search depth) using hill-climbing search to iteratively improve the weight estimates until no further improvement is possible. The model which best describes the opponent's moves is then used in the search routine of the modelling agent. Chajewska et al. (2001) consider a similar setting and assume that the modelled agent's utility function is a linear weighting of "subutilities". Here, the weighting is known and the goal is to learn the subutilities. Given observed play trajectories, the proposed method generates linear constraints on the space of possible utility functions, similar to methods of inverse reinforcement learning (Ng and Russell, 2000). To select a utility function from the space of possible functions, the authors propose to use a Bayesian prior which is conditioned on observed actions, and the resulting posterior is used to sample a utility function. Gal et al. (2004) consider single-shot normal-form games and model a human player's utilities as a linear combination of social factors such as social welfare and fairness. Data is collected from human play and utility weight profiles

are learned using expectation-maximisation and gradient ascent algorithms. A prior distribution over the different profiles is used to compute expected payoffs for actions.

Learning the utility function, or preferences, of other agents is also a major line of research in automated negotiation agents (see Baarslag et al. (2016) for a detailed description of many domain-specific methods). For instance, Hindriks and Tykhonov (2008) consider a bilateral multi-issue negotiation and define utility functions as weighted sums of issue evaluation functions. To learn the weights and evaluation functions ascribed by the opponent to each issue, the authors discretise the space of possible weights and evaluation functions by assuming special functional forms. This results in a finite hypothesis space of utility functions over which a Bayesian prior is defined and updated after new bids are received. The resulting posterior can be used to estimate the opponent's utility function. Coehoorn and Jennings (2004) also consider linearly additive utility functions and learn the weights using kernel density estimation. (See also Section 4.6 for utility reconstruction methods in graphical models.)

### *4.2.* Type-Based Reasoning

Learning new models from scratch via policy reconstruction can be a slow process, since many observations may be needed before the modelling process yields a useful model. This can be a problem in applications in which an agent does not have the time or opportunity to collect many observations about another agent. In such cases, it is useful if the agent is able to reuse models learned in previous interactions with other agents, such that it only needs to find the model which most closely resembles the observed behaviour of the modelled agent in the current interaction. In fact, there may be cases in which we know a priori that the modelled agent has one of several known behaviours, and we can provide specifications of those behaviours to the modelling agent.

Based on the above intuition, type-based reasoning methods assume that the modelled agent has one of several known *types*. Each type is a complete specification (a model) of the agent's behaviour, taking as input the observed interaction history and assigning probabilities to the actions available to the modelled agent. Types may be obtained in different ways: they may be specified manually by a domain expert; they may have been learned in previous interactions or generated from a corpus of historical data (e.g. Barrett et al., 2013); or they may be hypothesised automatically from the domain and task to be completed (e.g. Albrecht et al., 2015b). Given a specification of possible types, type-based reasoning begins with a prior belief which specifies the expected probabilities of types before any actions are observed. During the interaction, each time a new action is observed, the belief is updated according to the probability with which the types predicted the observed action. The modelling agent can then use the updated belief and the types in a planning procedure to compute optimal actions with respect to the types and belief. A useful property of this method is that, if the true type of the modelled agent (or a sufficiently similar type) is in the set of considered types, then the beliefs can often point to this type after only a few observations, leading to fast adaptation. Moreover, since types are essentially blackbox mappings, they can encapsulate policy reconstruction methods to learn new types during the interaction (Albrecht and Ramamoorthy, 2013; Barrett et al., 2011).

Type-based reasoning was first studied by game theorists, who considered games in which all players maintain beliefs about the possible types of the other players (Harsanyi, 1967). The principal questions studied in this context are the degree to which players can learn to make correct predictions through repeated interactions, and whether the interaction process converges to solutions such as Nash equilibria (Nash, 1950). A well-known result by Kalai and Lehrer (1993) states that, under a certain "absolute continuity" assumption regarding players' beliefs, their prediction of future play will get arbitrarily close to the true future play and convergence to

Nash equilibrium emerges. (The assumption states that every event with true positive probability is assigned positive probability under the players' beliefs.) Subsequent works studied the impact of prior beliefs on equilibrium convergence and showed that if players have different prior beliefs, their play may converge to a subjective equilibrium which is not a Nash equilibrium (Dekel et al., 2004; Nyarko, 1998). Lastly, for certain games and conditions, there are results which show that players cannot simultaneously have correct beliefs and play optimally with respect to their beliefs (Nachbar, 2005; Foster and Young, 2001).

In AI research, type-based reasoning[7] found popularity in problems of multiagent interaction without prior coordination (Albrecht et al., 2017; Stone et al., 2010), in which the controlled agent interacts with other agents whose behaviours are initially unknown. Albrecht et al. (2016) provide a concise and compact definition of a type-based reasoning method via a recursive combination of the Bayes-Nash equilibrium (Harsanyi, 1968a) and Bellman optimality equation (Bellman, 1957). This combination results in a tree of all possible interaction trajectories as well as their predicted probabilities and payoffs, where the probabilities take into account changes in beliefs along the trajectories. The authors define different belief formulations and analyse their convergence properties (Albrecht and Ramamoorthy, 2014). They also show empirically that prior beliefs can have a significant long-term impact on payoff maximisation, and that they can be computed automatically with consistent performance effects (Albrecht et al., 2015b). Barrett et al. (2011) modify the sampling-based planner UCT (Kocsis and Szepesvári, 2006) such that each rollout in UCT samples a type for each other agent based the current belief over types. The algorithm is evaluated in the "pursuit" grid-world domain where it could perform well even if the true types of other agents were not in the set of considered types, so long as sufficiently similar types were known. In subsequent work, Barrett et al. (2013) show how transfer learning can be used to adapt decision-tree types learned in previous interactions. Rovatsos et al. (2003) propose a method which dynamically learns up to a certain number of types which are represented as deterministic finite automata. When interacting with a new agent, the method finds the closest known type or adds a new type for future reference. Optimal actions against a type are computed using reinforcement learning methods such as Q-learning (Watkins and Dayan, 1992). Takahashi et al. (2002) propose a "multi-module" reinforcement learning method where each module corresponds to a possible agent type and a "gating signal" is used to determine how closely each module matches the current agent. Type-based reasoning has also been studied under partial-observability conditions. In Interactive POMDPs (Gmytrasiewicz and Doshi, 2005), agents have possible uncertainty about the state of the environment, the types of other agents, and their chosen actions. (We defer a more detailed discussion of this model to Section 4.5).

The above methods all use Bayes' law or some modification thereof to determine the relative likelihood of types, given the observed actions of the modelled agent. An alternative to Bayes' law is to use machine learning methods such as artificial neural networks, which can learn to predict "mixtures" of types (represented as weight vectors) given the observed actions. For example, Lockett et al. (2007) propose a method which consists of two neural networks: one network is trained to predict a mixture of types, taking as input the observed actions of the modelled agent; another network is trained to make decisions by assigning probabilities to available actions, taking as input the observed actions and the predicted mixture from the first network. Similarly, He et al. (2016) train a "gating network" which combines the predicted Q-values of several "expert networks" corresponding to different agent types.

[7] The 2016 AAAI Conference on Artificial Intelligence held a tutorial on "Type-Based Methods for Interaction in Multiagent Systems". Tutorial slides can be downloaded at: `http://thinc.cs.uga.edu/tutorials/aaai-16.html`

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Albrecht and Stone, 2017) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Sadigh et al., 2016) | no | no | yes | yes | no | simult. | continuous | full |
| (He et al., 2016) | yes | no** | yes | no | no | simult. | mixed | full |
| (Albrecht et al., 2016, 2015b) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Albrecht and Ramamoorthy, 2014, 2013) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Barrett et al., 2013, 2011) | yes | no | yes | yes | yes | simult. | discrete | full |
| (Lockett et al., 2007) | yes | no | yes | yes | no | altern. | discrete | partial/full |
| (Gmytrasiewicz and Doshi, 2005) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Nachbar, 2005) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Southey et al., 2005) | yes | no | yes | yes | no | altern. | discrete | partial/full |
| (Dekel et al., 2004) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Chalkiadakis and Boutilier, 2003) | yes | no | yes | yes | no | simult. | discrete | full |
| (Rovatsos et al., 2003) | no | no | yes | yes | no | simult. | discrete | full |
| (Takahashi et al., 2002) | yes | partial* | yes | yes | no | simult. | mixed | partial |
| (Foster and Young, 2001) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Carmel and Markovitch, 1999) | no | no | yes | yes | no | simult. | discrete | full |
| (Nyarko, 1998) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Kalai and Lehrer, 1993) | yes | yes | yes | yes | no | simult. | discrete | full |

Table 3: Assumptions in papers for type-based reasoning methods. *Types are Markov (non-changing) but modelled agent is assumed to change between types periodically. **Modelled agent may change types between episodes but not during episode.

Most type-based reasoning methods use discrete (usually finite) type spaces, where each type is a different decision function. Even inherently continuous hypothesis spaces can be discretised to obtain discrete type spaces (e.g. Hindriks and Tykhonov, 2008). However, one may also reason directly about continuous type spaces: essentially, we now have a single decision function which has some number of continuous parameters, and the beliefs quantify the relative likelihood of parameter values. A specific parameter setting can then be viewed as one type. For example, Southey et al. (2005) maintain Gaussian beliefs over the continuous parameters of a specified player function for Poker (cf. Table 1 in their paper). It is also possible to combine discrete and continuous type spaces. Albrecht and Stone (2017) propose a method which reasons simultaneously about both the relative likelihood of a finite set of types *and* the values of any bounded continuous parameters within these types. The method begins with an initial parameter estimate for each discrete type. After new actions are observed, a subset of the types is selected and their parameter estimates updated using methods such as approximate Bayesian updating and exact global optimisation (Horst et al., 2000).

An interesting aspect of type-based reasoning is the possibility of deliberately choosing actions to elicit information about an agent's type. While it is possible to use schemes such as occasional randomisation in action selection, such schemes ignore the risk that the exploratory actions may influence the modelled agent in unintended ways (Carmel and Markovitch, 1999). In this regard, type-based reasoning can naturally integrate a decision-theoretic "value of information" (Howard, 1966) into the evaluation of actions. For example, the methods proposed by Carmel and Markovitch (1999) and Albrecht et al. (2016) recursively take into account the potential information that actions may reveal about the type of the modelled agent and how this in turn may affect the future interaction. Chalkiadakis and Boutilier (2003) propose a "myopic" approximation of this kind

of reasoning which considers only one recursion of belief change, after which beliefs are held constant for the evaluation of actions. Sadigh et al. (2016) use a form of model predictive control to optimise a heuristic tradeoff between minimising uncertainty in the modelled agent's type and maximising a given reward function. In the related context of goal recognition (cf. Section 4.4), the "Proactive Execution Module" of Schmid et al. (2007) incorporates several criteria in the selection of actions, including uncertainty minimisation, expected success, and minimising risk values assigned to actions.

### *4.3.* Classification

While policy reconstruction (Section 4.1) and type-based reasoning (Section 4.2) attempt to predict the future actions of the modelled agent, there may be other properties or quantities of interest which an agent model could predict. For example, an agent model may make predictions about more abstract properties such as whether the play style of the modelled agent is "aggressive" or "defensive" (e.g. Schadd et al., 2007), or it may predict quantities such as the expected times at which the modelled agent will take certain actions (e.g. Weber and Mateas, 2009). The former task of assigning one of a finite number of labels is referred to as *classification*, whereas the latter task of predicting continuous values is referred to as *regression*. There are different ways in which such predictions can be utilised by a modelling agent. For instance, an assigned class label can be naturally incorporated into the decision procedure of the modelling agent using if-then-else rules or decision trees. Alternatively, given a class label, the agent may employ a precomputed strategy which is expected to be effective against that particular class label.

Classification methods[8] produce models which assign class labels to the modelled agent (e.g. "play-style = aggressive") based on information from the observed interaction. Similarly to policy reconstruction methods, there are two central design questions in classification methods: (1) what observations from the interaction should be used and how should they be represented to facilitate the classification, and (2) how should the classification be performed given the data representation? The second question often includes a learning phase which is carried out prior to the current interaction, using data collected from past interactions.

Several classification methods have been proposed to model players in complex strategy games. Weber and Mateas (2009) propose methods to predict a player's strategy and build times in the game Starcraft. The models are trained on collected replay data from expert human players. Each replay is tagged as one of six strategies and transformed into a feature vector which contains the initial build times for the various unit types in the game. A number of machine learning algorithms (e.g. decision trees, nearest neighbours) are tested on the data and the results show that the learned models can successfully predict player strategies and build times. Using the same collected replay data, Synnaeve and Bessiere (2011) propose methods to classify the opening strategy of Starcraft players from a finite set of strategies, using expectation-maximisation and k-means algorithms. Schadd et al. (2007) propose domain-specific classifiers to predict the play style (e.g. "aggressive", "defensive") of players in the game Spring. To account for possible changes in play style, the model prioritises recent observations over past observations. Spronck and den Teuling (2010) use support vector machines (SVMs) (Cortes and Vapnik, 1995) to predict the "preferences" of players in the game Civilization IV. Each player is characterised by integer-valued preferences

[8] We focus on classification methods since many of the surveyed papers in this section are in this category. Note also that regression problems can be transformed into classification problems via a finite discretisation of values, albeit with an exponential growth of class labels if multiple regression variables are jointly discretised.

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Synnaeve and Bessiere, 2011) | yes | no | –* | no | no | simult. | mixed | partial |
| (Bombini et al., 2010) | no | no | –* | no | no | simult. | mixed | partial |
| (Iglesias et al., 2010) | yes | yes | –* | yes | no | simult. | mixed | partial |
| (Spronck and den Teuling, 2010) | yes | no | –* | no | no | simult. | mixed | partial |
| (Laviers et al., 2009) | no | no | yes | no | no | simult. | mixed | full |
| (Weber and Mateas, 2009) | yes | no | –* | no | no | simult. | mixed | partial |
| (Iglesias et al., 2008) | yes | no | –* | no | no | simult. | mixed | partial |
| (Schadd et al., 2007) | yes | yes | –* | no | no | simult. | mixed | partial |
| (Sukthankar and Sycara, 2007) | no | no | yes | yes | no | simult. | discrete | full |
| (Huynh et al., 2006) | yes | no | –* | –*** | no | –*** | mixed | partial |
| (Steffens, 2004b) | yes | no | yes** | no | no | simult. | mixed | partial |
| (Mui et al., 2002) | yes | no | –* | yes | no | simult. | discrete | full |
| (Visser and Weland, 2002) | yes | no | –* | yes | no | simult. | mixed | partial |
| (Sabater and Sierra, 2001) | yes | no | –* | –*** | no | –*** | mixed | partial |
| (Abdul-Rahman and Hailes, 2000) | yes | no | –* | –*** | no | –*** | –*** | –*** |
| (Riley and Veloso, 2000) | yes | no | –* | no | no | simult. | mixed | partial |
| (Schillo et al., 2000) | yes | no | –* | yes | no | simult. | discrete | full |

Table 4: Assumptions in papers for classification methods. *This assumption does not apply here since the goal is not to predict the actions of agents. **Method is in principle based on action prediction and requires specification of decision factors (state descriptions). ***Not specified.

in areas such as military, cultural, and scientific development. Training data are generated by pitting predefined AI players with different preference settings against each other. The collected data consist of game states which are transformed into feature vectors with attributes such as the number of cities and units. Using the data, one SVM classifier is trained for each preference. Laviers et al. (2009) use SVMs to classify the defensive play of opponent teams in the football game Rush 2008. The game specifies finite sets of team formations and plays for offense and defense. Using game data generated from all combinations of these team formations and plays, a series of multi-label SVM classifiers is trained corresponding to increasing lengths in observation sequences. Sukthankar and Sycara (2007) consider turn-based strategy games such as Dungeons & Dragons and train SVMs to classify players into a finite set of roles (e.g. "scout", "medic") using simulated game data for the various roles.

Another complex domain in which classification methods have been studied is simulated robot soccer. Two notable differences to the above methods are that the models now predict the identities of players or entire teams, and the (partial) use of symbolic methods in addition to statistical machine learning methods. Steffens (2004b) proposes the "Feature-Based Declarative" classification method. Therein, each model consists of a number of features which are defined as pairs of logical state descriptions and the actions of one or more opponent players expected to be seen in the described states. Compactness of models is achieved by limiting models to features which are highly distinctive (relative to other models) and stable, meaning that they occur frequently for the model. Given an observation of the game, consisting of the game state and player actions, different symbolic approaches and a Bayesian approach can be used to match features to observations. A successful match to the features of a model means that the opponent has been identified. Bombini et al. (2010) propose a relational procedure which works on temporal

sequences of game events for a given team. Each sequence consists of high-level actions such as passing and dribbling, which in turn consist of low-level (primitive) actions such as kicking and turning. Inductive logic programming (Muggleton, 1991) is used to automatically select a feature representation from these sequences. Given the feature vectors, the method uses a k-nearest neighbour algorithm with a specified distance function between feature vectors to classify teams. Similarly, Iglesias et al. (2008) extract symbolic sequences of game events from which subsequences of a certain length are extracted and their frequencies represented in a "trie" structure (Fredkin, 1960), which is compared to known models using statistical hypothesis testing. This approach has been extended to allow for evolving agent behaviours, essentially by adding new models when the existing ones are found to be insufficient (Iglesias et al., 2010). Other methods proposed for simulated robot soccer include Riley and Veloso (2000), who classify teams based on a grid discretisation of the playing field which is used to count the occurrence of certain events (such as ball/player positions and pass/dribble events) in specific geographic areas, and Visser and Weland (2002) who learn decision trees to classify the behaviour of the goal keeper (e.g. "leaving goal", "returning to goal") and the passing behaviour of opponent players.

Trust and reputation in multiagent systems is an area of research which uses classification and regression methods to model the trustworthiness of agents (see Pinyol and Sabater-Mir (2013), Yu et al. (2013), and Ramchurn et al. (2004) for useful surveys). One definition of trust is the expectation with which an agent will realise its terms of a contract in a given context (many other definitions exist, e.g. Dasgupta, 2000). Trust can be based on a multitude of information, including own experiences from interactions with the modelled agent, communicated experiences from other agents in the system, as well as the roles of the modelled agent and its social relations to other agents. For example, Abdul-Rahman and Hailes (2000) classify agents as very trustworthy, trustworthy, untrustworthy, or very untrustworthy based on direct experiences and reported experiences about agents. Many other proposed methods quantify trust as a continuous value which aggregates various information sources using relative importance weights, confidence values, time discounting, etc. (e.g. Huynh et al., 2006; Mui et al., 2002; Sabater and Sierra, 2001; Schillo et al., 2000). Such qualitative or quantitative predictions of trust levels can be used by the modelling agent to tailor its interaction with the modelled agent, and, importantly, trust levels can be used to decide which agents to interact with in the first place.

### *4.4.* Plan Recognition

Plan recognition is the task of identifying the possible goals and plans of an agent, based on the agent's observed actions (Carberry, 2001). The focus is on predicting the intended end-product (goal) of the actions that have been observed so far, as well as the sequence of steps (plan) with which the agent intends to achieve its goal.[9] Knowledge of the goals and plans of other agents can be extremely useful in interactions with them. For example, an adaptive user interface may suggest certain actions and display other relevant information if it knows what the human user intends to accomplish (Oh et al., 2011; McTear, 1993), and an intrusion detection system may take certain counter measures if it detects the goals and plan of an attacker (Geib and Goldman, 2001).

Many plan recognition methods employ a *plan library* which describes the possible plans and goals that the observed agent may pursue. The representation of plans is a key element in plan recognition methods, and many methods use a hierarchical[10] representation in which "top-level"

[9] "Goal recognition design" is a closely related problem in which the goal is to modify the environment such that any agent acting in it reveals its goal as early as possible (Wayllace et al., 2017; Keren et al., 2016, 2015, 2014).

[10] Two examples of hierarchical plan libraries are the network security domain of Geib and Goldman (2009) and the pasta-making domain of Kautz and Allen (1986).

goals are decomposed into sub-plans which may be further decomposable. The leaves in this plan hierarchy are the primitive (non-decomposable) actions that can potentially be observed. Plan libraries may also include additional rules such as temporal orderings between the steps in plans, and preconditions on the environment state which must hold in order to perform certain plan steps. Given such a plan library and a set of observed actions, the plan recognition task is to generate possible plan hypotheses that respect the rules of the plan library and explain (i.e. contain) all observed actions. If multiple plan hypotheses exist that explain the observed actions, they may be distinguished by additional factors such as how plausible or probable they are.

Plan recognition differs from policy reconstruction (Section 4.1) and type-based reasoning (Section 4.2) in that the latter predict actions for given situations, but they do not predict the intended end-product of these actions, such as that the modelled agent seeks to reach a certain goal state in the environment. On the other hand, while plan recognition can also be used to predict future *actions*, the resulting predictions are often less precise than predictions of models produced by policy reconstruction and type-based reasoning (with some notable exceptions, e.g. Bui et al. (2002)). For example, plans often specify a partial temporal order of actions, such as that some actions have to occur before some other actions. While this flexibility is useful for planning, it leaves open the precise order and probability of actions in a plan execution. Hence, a plan may predict a set of possible actions but not necessarily which action will be taken next.

Plan recognition methods are sometimes categorised into "keyhole" and "intended" methods (Cohen et al., 1981). The difference is in whether the modelled agent is assumed to be aware of the modelling agent. The vast majority of current methods are designed for keyhole plan recognition, in which the modelled agent is assumed to be unaware of the modelling agent.

#### *4.4.1. Plan Recognition in Hierarchical Plan Libraries*

Kautz and Allen (1986) propose a symbolic theory of plan recognition in which plans are represented using complex hierarchical actions that decompose into other complex and primitive actions. This results in a graph representation in which edges denote plan decomposition, and root nodes in the graph correspond to "top-level plans" which can be interpreted as goals. The recognition problem is then framed as a problem of graph covering given the observed (primitive) actions, which the authors formulate using the concept of circumscription (McCarthy, 1980). Tambe and Rosenbloom (1995) use a hierarchical plan hierarchy in which plan steps are conditioned on environment states. The proposed method commits early to a single plan hypothesis and evaluates new observations in the context of this hypothesis. If the current plan hypothesis is inconsistent with new observations, the method attempts to repair the hypothesis via limited backtracking in the plan hierarchy. Avrahami-Zilberbrand and Kaminka (2005) represent the plan library as a directed acyclic graph which specifies decomposition, temporal orderings, and applicability conditions of plan steps. The plan recognition is carried out via a "lazy" procedure which time-stamps complete paths in the plan graph that match new observations and respect the temporal orderings and applicability conditions. A complete set of plan hypotheses can then be extracted when needed (hence "lazy"). Several extensions to this method have been proposed: one which allows for action duration, interleaved plan execution, and missing observations (Avrahami-Zilberbrand et al., 2005); an extension to rank plan hypotheses by their expected utility to the modelling agent (Avrahami-Zilberbrand and Kaminka, 2007); and an extension which incorporates timing constraints on the plan recognition task (Fagundes et al., 2014).

Charniak and Goldman (1993) frame plan recognition as a problem of probabilistic inference in Bayesian networks (Pearl, 1988). The plan library is represented as a set of decomposable actions, based on which a set of Bayesian networks can be constructed. The root of each network

corresponds to a high-level plan for which prior probabilities must be specified, and the child nodes correspond to plan decomposition. The "belief" in this plan hypothesis is expressed by the probability that the value of the root node is true, which can be computed using standard inference algorithms (Pearl, 1988). Bui et al. (2002) represent plans as a $K$-depth hierarchy of abstract policies, where a policy at depth $k$ selects a policy at depth $k-1$, and policies at depth $k=0$ are the primitive actions. A notable difference from other formulations is that the policies are defined over environment states, which is similar to models learned in policy reconstruction (Section 4.1) and type-based reasoning (Section 4.2). The authors show how the recognition process can be framed using dynamic Bayesian networks and they perform inference using the Rao-Blackwellised particle filter (Doucet et al., 2000). A related method is based on probabilistic state-dependent grammars which allow the plan production rules to depend on state information (Pynadath and Wellman, 2000). Geib and Goldman (2009) represent plans based on AND/OR trees, in which AND children are required steps in plans with possible temporal constraints and OR children are alternative (choice) steps in plans of which one must be performed. Their method uses a generative model of plan execution which specifies probabilities for how an agent decides on a particular plan and how the steps in the plan are executed. This plan execution model can be simulated and the authors show how the model can be used to infer plans based on observations.

#### *4.4.2. Plan Recognition by Planning in Domain Models*

Two potential drawbacks of using plan libraries are that their specification can be tedious, and that they may be incomplete (i.e. the observed agent may use a plan that cannot be constructed with the plan library). Ramírez and Geffner (2009) propose an alternative formulation of plan recognition as a problem of planning in a domain model which is specified in the STRIPS planning language (Fikes and Nilsson, 1971). Given a set of possible goals, the idea is that the potential goals of the observed agent are those goals for which the optimal plans that achieve the goals contain the observed actions in the order in which they were observed. This idea assumes that the modelled agent is "rational" in that it only executes optimal plans with respect to a known cost definition (similar to methods of utility reconstruction; cf. Section 4.1.4). The authors show how existing exact and approximate planning methods can be adopted to compute this set of goals, essentially by solving the planning problem for the modelled agent such that the solution is consistent with the observed actions. This work is subsequently extended to compute Bayesian probabilities over plan hypotheses (Ramírez and Geffner, 2010). Each possible goal now has a specified prior probability, and the likelihood of the observed actions given a goal is defined as the cost difference between the plan that optimally achieves the goal and the plan that optimally achieves the goal and is consistent with the observed actions. This likelihood definition encodes the assumption that an agent is more likely to pursue optimal plans than suboptimal ones. (See also the work of Sohrabi et al. (2016) for an alternative probabilistic extension which allows for unreliable observations, and the work of Vered and Kaminka (2017) for a heuristic extension that works with continuous domains.) Baker et al. (2009, 2005) propose a very similar idea to Ramírez and Geffner (2009) but formulate it within Markov decision processes (MDPs) (Bellman, 1957). Since MDPs allow for stochasticity in state transitions and action choices, any optimal policy for an MDP that achieves a specific goal induces a likelihood of the observed actions given the goal, which can be used to compute Bayesian posteriors over the alternative goals. Similar goal recognition methods using MDPs were proposed by Nguyen et al. (2011) and Fern and Tadepalli (2010). In subsequent work, both Baker et al. (2011) and Ramırez and Geffner (2011) propose planning-based methods to infer the goals (and beliefs) of an agent in partially observable MDPs (Kaelbling et al., 1998). Lesh and Etzioni (1995) and Hong (2001, 2000) propose symbolic

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Vered and Kaminka, 2017) | yes | no | yes | yes | no | –** | continuous | full |
| (Sohrabi et al., 2016) | yes | no | yes | yes | no | –** | discrete | partial |
| (Tian et al., 2016) | yes | yes | –* | yes | no | –** | –*/disc. | partial |
| (Fagundes et al., 2014) | no | no | yes | yes | no | –** | mixed/disc. | full |
| (Baker et al., 2011) | yes | yes | yes | yes | no | –** | discrete | partial |
| (Ramırez and Geffner, 2011) | yes | no | yes | yes | no | –** | discrete | partial |
| (Nguyen et al., 2011) | yes | no | yes | yes | yes | simult. | discrete | full |
| (Fern and Tadepalli, 2010) | yes | no | yes | yes | yes | simult. | discrete | full |
| (Ramírez and Geffner, 2010) | yes | no | yes | yes | no | –** | discrete | full |
| (Gold, 2010) | yes | no | yes | yes | no | simult. | discrete | full |
| (Geib and Goldman, 2009) | yes | no | –* | yes | no | –** | discrete | partial |
| (Ramírez and Geffner, 2009) | no | no | yes | yes | no | –** | discrete | full |
| (Baker et al., 2009) | yes | yes | yes | yes | no | –** | discrete | full |
| (Avrahami-Zilberbrand and Kaminka, 2007) | yes | no | yes | yes | no | –** | mixed/disc. | partial |
| (Blaylock and Allen, 2006) | yes | no | –* | yes | no | –** | discrete | full |
| (Avrahami-Zilberbrand and Kaminka, 2005) | no | no | yes | yes | no | –** | mixed/disc. | full |
| (Avrahami-Zilberbrand et al., 2005) | no | no | yes | yes | no | –** | mixed/disc. | partial |
| (Baker et al., 2005) | yes | yes | yes | yes | no | –** | discrete | full |
| (Blaylock and Allen, 2004, 2003) | yes | no | –* | yes | no | –** | discrete | full |
| (Fagan and Cunningham, 2003) | no | yes | yes | yes | no | –** | discrete | full |
| (Kerkez and Cox, 2003) | no | yes | yes | yes | no | –** | discrete | full |
| (Bui et al., 2002) | yes | no | yes | yes | no | –** | discrete | partial |
| (Hong, 2001, 2000) | no | no | yes | yes | no | –** | discrete | full |
| (Pynadath and Wellman, 2000) | yes | no | yes | yes | no | –** | discrete | partial |
| (Albrecht et al., 1998, 1997) | yes | yes | yes | yes | no | –** | discrete | partial |
| (Lesh and Etzioni, 1995) | no | no | yes | yes | no | –** | discrete | full |
| (Tambe and Rosenbloom, 1995) | no | no | yes | yes | no | –** | mixed/disc. | full |
| (Baré et al., 1994) | yes | no | yes | yes | no | –** | mixed | partial |
| (Charniak and Goldman, 1993) | no | no | –* | yes | no | –** | –*/disc. | full |
| (Kautz and Allen, 1986) | no | no | –* | yes | no | –** | –*/disc. | full |

Table 5: Assumptions in papers for plan recognition methods. *Does not model environment states. **Does not define move order between agents.

graph-based methods for domains specified in extensions of the STRIPS language. Both methods construct graph structures based on the domain model and observed actions, and utilise this structure to find a subset of goals which are consistent with the observed actions.

### *4.4.3. Plan Recognition by Similarity to Past Plans*

Plan hypotheses may also be generated based on similarity to past observed plans. This idea was explored in the context of case-based reasoning methods for plan recognition (Kerkez and Cox, 2003; Fagan and Cunningham, 2003; Baré et al., 1994). For example, Kerkez and Cox (2003) represent a plan as a sequence of environment states and actions in each state. Given the current state, a history of observed actions, and a case base consisting of previously observed plans, the recognition task is to retrieve plans from the case base which are similar to the current situation. One way to define similarity is by using state abstractions whereby states that share certain properties are grouped together. A useful property of this approach is that the plan library (case base) does not need to be fully specified ahead of time and can be expanded after new

plans have been observed. (See also Section 4.1.2 for case-based reasoning methods for policy reconstruction.) Tian et al. (2016) formulate plan recognition as a problem of sentence completion in natural language processing. A sentence (plan) is a sequence of words (actions), and the corpus (plan library) consists of previously seen sentences. Based on the corpus, natural language processing methods are used to learn probability distributions for how words may surround other words. An incomplete sentence (plan) can then be completed by filling the missing words such that the overall probability of the resulting sentence is maximised. (See also Geib and Steedman (2007) for a discussion of the connections between plan recognition and natural language processing.) Albrecht et al. (1998, 1997) seek to recognise what "quest" a player is pursuing in an online adventure game, for which they use a dynamic Bayesian network (Dean and Kanazawa, 1989) whose parameters are learned using a corpus of historical play data. Similarly, Gold (2010) trains an Input-Output Hidden Markov Model (Bengio and Frasconi, 1995) to predict a player's goal in an action-adventure game. Closely related is the work of Blaylock and Allen (2004, 2003), who compute goal probabilities as a product of conditional action probabilities which are learned using a corpus of observed plan executions. This work was later extended to recognise hierarchical sub-goals (Blaylock and Allen, 2006).

### *4.5.* Recursive Reasoning

Autonomous agents often base their decisions on explicit beliefs about the state of the environment and, possibly, the mental states of other agents. The mental states of other agents may, in turn, also contain beliefs about the environment and mental states of other agents. This nesting of beliefs leads to a possibly infinite reasoning process of the form "I believe that you believe that I believe...". While the modelling methods discussed in the previous sections do not model such nested beliefs, methods of *recursive reasoning* use explicit representations of nested beliefs and "simulate" the reasoning processes of other agents to predict their actions.

Game theorists first addressed infinitely nested beliefs in the context of incomplete information games, in which some components of the game (such as players' payoff functions) are not common knowledge (Harsanyi, 1962). In Bayesian games (Harsanyi, 1967), an early precursor of type-based reasoning (see Section 4.2), the infinite regress is resolved by assuming that the private elements of players are drawn from a distribution that is common knowledge. While this assumption allows for an elegant equilibrium analysis (Harsanyi, 1968b), creating such a setting is rather impractical when designing an autonomous agent that is interacting with unknown other agents. Recursive reasoning methods follow a more direct approach by *approximating* the belief nesting down to a fixed recursion depth. As a prototypical example, assume agent A is modelling another agent B. In order to choose an action, A predicts the next action of B by simulating the decision making of B given what A believes about B. This requires a prediction of A's next action from B's perspective, given what A believes B to believe about A, and so on. The recursion is terminated at some predetermined depth by fixing the action prediction to some probability distribution, e.g. uniform probabilities. The prediction at the bottom of this recursion is passed up to the above recursion level to choose an optimal action at that level, which in turn is passed to the next higher level, and so on, until agent A can make its *actual* choice at the beginning of the recursion. Note that the recursion assumes that each agent believes to have more sophisticated (deeper) beliefs than the other agent. Another central assumption is that each agent assumes the other agent to be *rational*[11] in that it will choose optimal actions with respect to its beliefs.

[11] We already saw instances of this rationality assumption in utility reconstruction (Section 4.1.4) and some approaches for plan recognition (Section 4.4).

The method proposed by Carmel and Markovitch (1996a) implements the recursion outlined above for game tree search in games with alternating moves. Here, an agent model specifies the agent's evaluation function for game states as well as the evaluation function the agent believes its opponent to use, and so on. As the authors point out, the well-known minimax algorithm for zero-sum games (Campbell and Marsland, 1983) is a special case of this method in which the evaluation function of the opponent is simply the negative of one's own function. The "Recursive Modeling Method" (RMM) (Gmytrasiewicz and Durfee, 2000, 1995; Gmytrasiewicz et al., 1991) also implements the above recursion, with the added complexity that agents may be uncertain about the exact model of other agents, such as their payoff function and recursion depth. In the above example, agent A has additional probabilistic beliefs about the possible models of agent B. During the recursion, A has to predict B's action under each possible model, adding an extra branching factor to the recursion. The resulting predictions are then weighted by the probabilities in A's beliefs about B's models. Gmytrasiewicz et al. (1998) also show how these beliefs can be updated after new observations, which involves the recursive updating of the beliefs of other agents, such that A updates its own belief about B's models, and B's expected belief about A's possible models, and so on. Vidal and Durfee (1995) show how the recursion in RMM can be made more efficient by pruning branches in the recursion tree which are expected to have no or minimal influence on the final choice of the agent.

RMM is the precursor of the Interactive POMDP (I-POMDP) (Gmytrasiewicz and Doshi, 2005). In a POMDP (Sondik, 1971), an agent makes sequential decisions based on its belief about the state of the environment, which is represented as a probability distribution over possible states and updated based on incomplete and uncertain observations. I-POMDPs modify POMDPs by adding model spaces to the environment state, such that an agent has beliefs about the environment state *and* the models of other agents. Agent models are categorised into "sub-intentional" and "intentional" models. A sub-intentional model defines any non-recursive mapping from observation histories to action probabilities, such as the finite state automata used in the work of Panella and Gmytrasiewicz (2017). In contrast, intentional models are themselves defined as I-POMDPs with beliefs about the environment and models of other agents. I-POMDPs are solved via a finite recursion as outlined above: To choose an optimal action, agent A has to solve the I-POMDP of agent B for each of its intentional models, which in turn requires solving the I-POMDP of agent A for each model ascribed to A by B, and so on, down to some fixed recursion depth. At the bottom of the recursion are standard POMDPs in which other agents are treated as "noise" in the transition and observation dynamics. These POMDPs can be solved directly using existing methods (Kaelbling et al., 1998) and their solutions are passed up the recursion tree. Several exact and approximate solution methods for I-POMDPs have been proposed, including methods based on model equivalence (Rathnasabapathy et al., 2006), particle filtering (Doshi and Gmytrasiewicz, 2009), value iteration (Doshi and Perez, 2008), policy iteration (Sonu and Doshi, 2015), and structural problem reduction (Hoang and Low, 2013). Ng et al. (2012) propose an even more complex modification of I-POMDPs in which agents are also uncertain about the transition and observation models of the environment.

An alternative to quantitative (probabilistic) representations of uncertainty (as used in RMM and I-POMDPs) are qualitative belief representations based on logics, such as dynamic epistemic logic (DEL) (Bolander and Andersen, 2011; Löwe et al., 2010). Epistemic logics are characterised by a knowledge operator $K_i\phi$ (or $B_i\phi$) which expresses that agent $i$ "knows" (or "believes") the formula $\phi$. For example, $K_iK_jK_i\phi$ corresponds to "agent $i$ knows that agent $j$ knows that agent $i$ knows $\phi$". The semantics of $K_i\phi$ are defined such that it holds true if $\phi$ is true in all world states that agent $i$ believes the world may be in. The dynamic aspect of DEL is given by event operators

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (de Weerd et al., 2017) | yes | no | yes | yes | no | altern. | discrete | full |
| (Kominis and Geffner, 2015) | no | no | yes | yes | no | simult. | discrete | partial |
| (Muise et al., 2015) | no | no | yes | yes | no | simult. | discrete | partial |
| (Sonu and Doshi, 2015) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (de Weerd et al., 2013) | yes | no | yes | yes | no | simult. | discrete | full |
| (Hoang and Low, 2013) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Ng et al., 2012) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Bolander and Andersen, 2011) | no | no | yes | yes | no | simult. | discrete | partial |
| (Löwe et al., 2010) | no | no | yes | yes | no | simult. | discrete | partial |
| (Doshi and Gmytrasiewicz, 2009) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Doshi and Perez, 2008) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Ghaderi et al., 2007) | no | no | yes | yes | yes | altern. | discrete | partial/full |
| (Rathnasabapathy et al., 2006) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Gmytrasiewicz and Doshi, 2005) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Camerer et al., 2004) | yes | no | yes | yes | no | simult. | discrete | full |
| (Van Der Hoek and Wooldridge, 2002) | no | no | yes | yes | no | simult. | discrete | partial |
| (Gmytrasiewicz and Durfee, 2000, 1995) | yes | no | yes | yes | no | –* | discrete | –** |
| (Gmytrasiewicz et al., 1998) | yes | no | yes | yes | no | –* | discrete | partial |
| (Carmel and Markovitch, 1996a) | no | no | yes | yes | no | altern. | discrete | full |
| (Vidal and Durfee, 1995) | yes | no | yes | yes | no | –* | discrete | –** |
| (Gmytrasiewicz et al., 1991) | yes | no | yes | yes | no | –* | discrete | –** |

Table 6: Assumptions in papers for recursive reasoning methods. *No explicit move order defined. **No explicit observation model used.

(actions) that can modify ontic and epistemic facts in the world via pre/post-conditions, similar to other planning languages such as STRIPS (Fikes and Nilsson, 1971). Several planning methods have been proposed that use such epistemic logics. Muise et al. (2015) and Kominis and Geffner (2015) both propose methods that solve epistemic planning problems using classical planning algorithms. Van Der Hoek and Wooldridge (2002) solve epistemic planning problems using model checking algorithms. Ghaderi et al. (2007) propose a framework based on the situation calculus (McCarthy and Hayes, 1969) for reasoning about beliefs and coordination in agent teams.

Given the belief nesting, an important question is how deep the recursion should be to achieve a robust interaction with humans and other agents. This question has been addressed extensively by researchers in behavioural game theory and experimental psychology (Camerer et al., 2015; Goodie et al., 2012; Wright and Leyton-Brown, 2010; Yoshida et al., 2008; Camerer et al., 2004; Hedden and Zhang, 2002). For example, Camerer et al. (2004) develop a simple recursive reasoning model in which an agent at recursion level $k$ has probabilistic beliefs regarding what level $k' < k$ the other agent uses. The beliefs are assumed to be correct, in that they are derived from a population distribution over recursion depths which is represented as a Poisson distribution. After "fitting" the model based on a large corpus of human play data, the authors find that humans reason on average at depth 1.5, i.e. one or two levels down the recursion. In addition to experiments with humans, some research pitted artificial recursive reasoning agents against each other to see what reasoning depths are most useful. For example, de Weerd et al. (2017, 2013) test their specific agents in domains such as repeated rock-paper-scissors and sequential negotiation, and find that reasoning levels deeper than 2 do not provide significant benefits in their setting.

### *4.6.* Graphical Models

The modelling methods discussed in the previous sections are based on rather abstract formulations of multiagent systems, in which much of the system's structure is left implicit. For example, a common formulation describes an environment which at any time is in some abstract state $s$, and transition probabilities between states are specified by some function $T(s, a, s')$ where $a$ is a tuple containing the agents' actions. In addition, an agent's utility is commonly defined as a general function $u(s, a)$ that depends on the state and joint action. What is left implicit in such formulations are the precise relations between the state components $s = (s_1, ..., s_m)$ (e.g. some components may depend on other components); how state components interact with the agents' decisions $a = (a_1, ..., a_n)$ (e.g. some agents may disregard certain components in their decisions); and the precise dependencies of utilities on state components and actions (e.g. an agent's utility may depend on the actions of some agents but not on others).

*Graphical models* make such dependencies explicit by using graph representations of multiagent systems. The advantage of making this structure explicit is that, if the interaction is only over a short horizon,[12] it can lead to compact models and more efficient algorithms, similarly to how Bayesian networks exploit conditional independence relations for compactness and efficient inference (Koller and Friedman, 2009; Pearl, 1988). Moreover, graphical models can be used as detailed mental models of how other agents may view the interaction.

The basic building block of many graphical models is the "Influence Diagram" (ID) (Howard and Matheson, 2005, 1984). An ID is a graphical representation of a single-agent decision problem. IDs use three types of nodes: chance nodes, which describe the components in the environment state; decision nodes, whose values the agent has to choose; and utility nodes, which determine the agent's utilities. Directed edges between nodes indicate dependence relations, e.g. the parent nodes of a decision node constitute the information that is used by the agent for that particular decision. A solution to an ID is a set of optimal decision rules, one for each decision node, which specify action probabilities for each input to the decision nodes (Shachter, 1986). Given a set of decision rules, an ID can be reduced to a normal Bayesian network by replacing each decision node with a chance node whose conditional probabilities are specified by the corresponding decision rule. One can then use standard inference algorithms (Pearl, 1988) to compute a variety of queries, such as expected utilities and the probability of certain events. The "Multi-Agent Influence Diagram" (MAID) (Koller and Milch, 2003) extends IDs by assigning each decision and utility node to one of several agents. Graphical games (Vickrey and Koller, 2002; Kearns et al., 2001; La Mura, 2000) can be viewed as a special type of MAID that have only decision and utility nodes. These works on MAID and graphical games show how the graph structure can be exploited for efficient computation of Nash equilibrium solutions (Nash, 1950).

Graphical models, such as IDs and MAIDs, can be used by an agent to model the decision making and *domain conceptualisation* of other agents. For example, an existing parent relation between a chance node $X$ and a decision node $D$ encodes the belief that the modelled agent incorporates $X$ in its decision for $D$; conversely, the absence of such a relation encodes the belief that the modelled agent does not account for $X$ in its decision for $D$ (or not directly). Several works have used graphical models for such mental representations of other agents. Suryadi and Gmytrasiewicz (1999) use IDs to model the capabilities, beliefs, and preferences of other agents.

[12] Graphical models can represent sequential interactions by adding additional nodes for each time step in the interaction, as well as dependencies between nodes in different time steps (Jensen and Nielsen, 2011). Unfortunately, this approach does not scale efficiently with the number of time steps (e.g. Doshi et al., 2009; Gal and Pfeffer, 2003b).

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Cadilhac et al., 2013) | yes | no | yes | yes | no | altern. | discrete | partial/full |
| (Zeng and Doshi, 2012) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Doshi et al., 2010, 2009) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Gal and Pfeffer, 2008) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Nielsen and Jensen, 2004) | yes | yes | yes | yes | no | –*** | discrete | full |
| (Gal and Pfeffer, 2003a) | yes | yes | yes | yes | no | simult. | discrete | partial |
| (Koller and Milch, 2003) | yes | no* | yes | yes | no | simult. | discrete | partial/–* |
| (Vickrey and Koller, 2002) | yes | no* | yes | yes | no | simult. | –**/disc. | –**/–* |
| (Kearns et al., 2001) | yes | no* | yes | yes | no | simult. | –**/disc. | –**/–* |
| (La Mura, 2000) | yes | no* | yes | yes | no | simult. | –**/disc. | –**/–* |
| (Milch and Koller, 2000) | yes | no | yes | yes | no | –*** | discrete | partial |
| (Chajewska et al., 2000) | yes | no | yes | yes | no | altern. | mixed | partial/full |
| (Suryadi and Gmytrasiewicz, 1999) | yes | yes | yes | yes | no | simult. | discrete | full |

Table 7: Assumptions in papers for graphical methods. *Does not model repeated interactions. **Does not model environment states. ***Does not define move order.

They show how the parameters of an ID may be modified to reflect the observed behaviour of an agent, focusing on learning the agent's preferences by modifying the utility nodes in the ID. Nielsen and Jensen (2004) also propose methods to learn the utility function in an ID for an observed agent. They relax the usual rationality assumption, which requires that the agent choose actions to strictly optimise its utilities, by allowing for random deviations from optimality. Milch and Koller (2000) define a probabilistic epistemic logic (cf. Section 4.5) to represent and infer the beliefs of agents, and use IDs to derive an agent's decision rules given its inferred beliefs and assuming the agent is rational. Cadilhac et al. (2013) use conditional preference (CP) networks (Boutilier et al., 2004) to model the preferences of players based on their negotiation dialogues. The resulting CP-nets are used to predict the players' actions by computing an equilibrium solution over the preferences encoded by the CP-nets. Chajewska et al. (2000) use IDs to represent the preferences of patients in a clinical trial and propose an algorithm for effective preference elicitation, which is the problem of deciding what questions to ask patients to obtain additional information about their preferences.

Graphical models can also represent uncertainty over multiple hypothesised models of other agents (as in type-based reasoning; see Section 4.2) and nested beliefs (as in recursive reasoning; see Section 4.5). "Networks of Influence Diagrams" (NIDs) (Gal and Pfeffer, 2008, 2003a) achieve this as follows: A NID is a single-rooted graphical model in which each node is a MAID. The root node of a NID represents the perspective of the modelling agent, and directed edges $A \rightarrow_{j,D} B$ indicate that the agent whose view is represented by the MAID in node $A$ believes that agent $j$ uses the MAID in node $B$ to make some decision $D$. If multiple such edges exist for the same agent $j$ and decision $D$, then the MAID in $A$ may contain a new chance node specifying the probabilistic belief of the modelling agent for each edge. The MAID in node $B$ may contain beliefs about other agents, and cycles in a NID are used to represent recursive reasoning. NIDs are solved by first solving the leaves of the NID, which are normal MAIDs that can be solved with existing methods (Koller and Milch, 2003). The solutions are decision rules for the decision nodes, which are passed to the parents in the NID, transforming them into MAIDs that can be solved, and so forth. A

related model is the "Interactive Dynamic ID" (I-DID) (Doshi et al., 2009) which was designed as a graphical representation of I-POMDPs (Gmytrasiewicz and Doshi, 2005) (cf. Section 4.5). In contrast to NIDs, which compute equilibrium solutions for a set of agents, I-DIDs are designed for subjective decision making of a single agent in a system containing multiple agents. This means that I-DIDs do not represent the decisions of other agents as decision nodes (as in MAIDs) but rather as chance nodes whose conditional probabilities are governed by the possible models ascribed to the agents, which may themselves be I-DIDs. Models and uncertainties over models are represented in a new "model node". I-DIDs represent temporal relations between nodes by "unrolling" the network for each time step in the interaction, such that edges between nodes in successive time steps indicate temporal dependencies (similar to dynamic Bayesian networks; Dean and Kanazawa (1989)). To manage the exponential growth of possible agent models after new observations, methods have been proposed which cluster behaviourally similar models (Zeng and Doshi, 2012; Doshi et al., 2010, 2009).

### *4.7.* Group Modelling

Most methods surveyed in the earlier sections use models that make predictions about a *single* agent, following the agent model shown in Figure 1. For methods that predict an agent's actions, such as policy reconstruction (Section 4.1), type-based reasoning (Section 4.2), and recursive reasoning (Section 4.5), modelling single agents is predicated on the assumption that agents choose actions independently from each other, as defined in Section 3. Thus, many papers proceed by explaining their methods for a single agent, with the underlying idea that the same method can be used to maintain separate models for each other agent. Note that this separation does not mean that agents ignore each other, since the models may base their predictions on the observed actions of other agents. Nonetheless, there are important cases in which it may be preferable to use *group models* which make joint predictions about a group of agents.

One such case is when agents have significant randomisation and *correlation* in their action choices (cf. Section 3), which cannot be captured by independent models. An example of this case is the concept of correlated equilibrium (Aumann, 1974), which generalises the Nash equilibrium by defining the equilibrium as a joint distribution over agents' actions rather than independent distributions. Many of the existing methods for policy reconstruction and type-based reasoning can be used to learn such action correlations, essentially by combining all other agents into a single agent whose action space is the Cartesian product of the agents' actions. This approach allows a model to capture action correlations by making predictions about the joint probability of actions. However, this approach may scale poorly since the action space of the "combined agent" grows exponentially in the number of combined agents and actions. A middle-path is to partition the other agents into smaller groups such that there is high expected correlation within groups but only little or no correlation between groups (an approach commonly used in probabilistic state estimation, e.g. Albrecht and Ramamoorthy, 2016; Boyen and Koller, 1998). The modelling agent can then use separate group models for each group.

Even when there is no significant randomisation in action choices, group models can often be more efficient and accurate by exploiting additional structure in the group. In particular, agent groups may act as *teams* which utilise structure such as roles within teams, dynamic formation of subteams, "divide-and-conquer" division of goals into sub-goals, as well as predefined joint plans and communication protocols (Stone and Veloso, 1999; Tambe, 1997; Grosz and Kraus, 1996; Cohen and Levesque, 1991). Knowledge of such structure can be used by group models to effectively limit the search space. For example, the behaviours of agents in a coordinated team, when observed in isolation, may not be very informative (and even possibly misleading) as to the

intended goals of the agents. However, when the same behaviours are interpreted in the context of a team, they may give important clues as to the goal and plan of the team (Tambe, 1996). In this spirit, a number of methods have been proposed which model teams rather than individual agents.

Section 4.3 already surveyed several works which use classification methods to identify teams and team strategies (Bombini et al., 2010; Laviers et al., 2009; Iglesias et al., 2008; Sukthankar and Sycara, 2007; Steffens, 2004b; Riley and Veloso, 2000). In addition, methods have been developed which model the physical formation and movement patterns of teams. Erdogan and Veloso (2011) use a hierarchical clustering method to extract clusters of similar movement trajectories from log data in the small size multi-robot league of RoboCup. During a game, the method observes an incomplete trajectory from the opponent team and classifies it into one of the extracted clusters, which allows it to predict future movements and compute counter-strategies. (Riley and Veloso, 2002) propose a method for simulated robot soccer which uses a predefined set of opponent models that specify probabilities of field positions for each player in the opponent team, given their initial positions and ball movements. Starting with a prior distribution over models, Bayesian updates are performed after new movement observations and the most probable model is used in the planning stage. Lattner et al. (2005) also consider simulated robot soccer and use unsupervised symbolic learning to extract movement patterns from observations. Kuhlmann et al. (2006) propose a method for the RoboCup simulated coach competition which can classify "patterns" (defined as exploitable weaknesses in an opponent team's strategy) by extracting feature vectors that include formation statistics, and comparing them to previously learned models from log data.

While the above methods learn and use models of *opponent* teams, an agent may also need to model its *own* team. This is important in problems of ad hoc (or impromptu) teamwork (Stone et al., 2010; Bowling and McCracken, 2005), in which an agent has to collaborate "on the fly" with an established but previously unknown team, without opportunities for prior coordination with the team members. Bowling and McCracken (2005) consider such a setting in the context of robot soccer, in which the team uses "plays" from a set of predefined plays, called the playbook. Each play specifies roles for the agents in the team along with sequences of synchronised actions for each role, as well as applicability and termination conditions for the play. A pickup player joins an established team but is not informed about the currently used plays nor its role in the plays. Assuming that the pickup player has access to a playbook, its task is to find the correct plays and its role within the plays. One proposed method to achieve this task is to compute a matching score for each play based on how well the play matches the observed actions in the team, and to select the play that has the highest matching score. Barrett and Stone (2015) consider a similar setting in the Half-Field Offense domain (Hausknecht et al., 2016) and use reinforcement learning to learn optimal collaboration policies for the pickup player in a range of previously encountered teams. During a new game, the pickup player uses the optimal policy for the past team which is most similar to the new team. Bayesian probabilities are calculated to quantify similarity between past teams and the new team, using models of past teams which predict transition probabilities between observed game states.

In addition to a large body of work on plan recognition for single agents (cf. Section 4.4), there is a growing body of work on multiagent plan recognition in which the modelling agent attempts to infer the goals and plans of an entire team of agents. Thus, plan libraries specify team plans that utilise additional structure such as roles within teams and division into subteams. Tambe (1996) extends a previous method (Tambe and Rosenbloom, 1995, cf. Section 4.4.1) by using a hierarchical team plan library. Teams can be divided into subteams which must be assigned to exactly one role in the team. Similar to the original method, the new method quickly commits to a single plan hypothesis and repairs inconsistencies via backtracking in the plan hierarchy.

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Barrett and Stone, 2015) | yes | no | yes | no | yes | simult. | contin./disc. | full/–* |
| (Zhuo et al., 2012) | no | no | yes | no | no | simult. | discrete | partial |
| (Banerjee and Kraemer, 2011) | no | no | yes | no | no | simult. | –**/discrete | partial |
| (Erdogan and Veloso, 2011) | yes | no | yes | yes | no | simult. | contin. | full/–* |
| (Zhuo and Li, 2011) | no | no | yes | no | no | simult. | –**/discrete | partial |
| (Banerjee et al., 2010) | no | no | yes | no | no | simult. | –**/discrete | full |
| (Sukthankar and Sycara, 2008) | no | no | yes | no | no | simult. | –**/discrete | partial |
| (Kuhlmann et al., 2006) | yes | no | yes | no | no | simult. | contin. | full/–* |
| (Bowling and McCracken, 2005) | no | yes | yes | no | yes | simult. | contin. | full/–* |
| (Lattner et al., 2005) | yes | no | yes | no | no | simult. | contin. | full/–* |
| (Saria and Mahadevan, 2004) | yes | no | yes | mixed*** | yes | simult. | discrete | partial |
| (Kaminka et al., 2002b) | yes | no | yes | mixed | yes | simult. | –**/discrete | partial |
| (Riley and Veloso, 2002) | yes | no | yes | yes | no | simult. | discrete | full/–* |
| (Tambe, 1996) | no | no | yes | no | no | simult. | mixed/disc. | full |

Table 8: Assumptions in papers for group modelling. *Actions are not directly observed. **Does not define environment states. ***Actions may be correlated in team (joint) policies and independent in the lower (individual) policies.

Saria and Mahadevan (2004) propose an extension of the abstract hidden Markov model (Bui et al., 2002, cf. Section 4.4.1) in which top-level joint policies for the team select lower-level policies for individual agents which are executed in a decentralised way. The proposed method proceeds similarly to the original work by defining the plan inference based on dynamic Bayesian networks and using particle filtering to perform the inference. Sukthankar and Sycara (2008) use a hierarchical plan library specified with AND/OR trees similar to the model of Geib and Goldman (2009) (cf. Section 4.4.1), with extra elements to specify the number of agents needed to commence a plan and special nodes in plan trees to generate and resolve subteams. The authors show how this additional structure can be utilised to prune the search space in the recognition task. Kaminka et al. (2002b) propose a method which infers a team's current plan based on overheard communications between team members, using plan and team hierarchies. Banerjee et al. (2010) show NP-completeness in a restricted version of multiagent plan recognition, in which team plans are defined as matrices that specify a sequence of synchronised actions for a subset of agents. This work was subsequently extended to allow for interleaved plan execution and incomplete observation traces (Banerjee and Kraemer, 2011). Zhuo and Li (2011) consider a similar formulation to Banerjee et al. (2010) but allow for partial observations. The proposed method frames the plan recognition problem as a satisfiability problem by automatically generating a set of constraints from the plan library and observations, which are solved using a MAX-SAT solver. In later work, Zhuo et al. (2012) propose a similar SAT-based recognition approach using action specifications in the STRIPS planning language rather than matrix-based plan libraries.

### *4.8.* Other Relevant Methods

In this section we briefly discuss several other relevant methods, namely implicit modelling, hypothesis testing for agent models, and safe best-response methods.

#### 4.8.1. Implicit Modelling

This survey focused on *explicit* modelling of other agents, in which agent models implement the mapping shown in Figure 1. In contrast, *implicit* modelling does not produce explicit models of other agents, but implicitly encodes aspects of other agents (such as their behaviours) in other structures or reasoning processes. For example, "expert" algorithms, which learn to follow the best expert policy from a given set of such policies (e.g. Crandall, 2014; de Farias and Megiddo, 2004), can be viewed as implicit modelling in that each expert policy may be optimal against a particular opponent and, thus, implicitly encode the opponent's behaviour without making explicit predictions about that opponent. Implicit modelling based on expert algorithms has been shown to be effective in variants of Poker (Bard et al., 2013; Hoehn et al., 2005). Other examples of implicit modelling include learning logical action descriptions in the context of other agents (Illobre et al., 2010; Guerra-Hernández et al., 2004); modelling other agents as part of the MDP transition dynamics (Hernandez-Leal et al., 2017); and using opponent features in a neural network to learn expected action utilities (He et al., 2016). A potential advantage of implicit modelling is that it may more naturally exploit synergies between modelling and planning by merging the two processes. Advantages of explicit modelling are that the models are decoupled from the planning and may thus be used by different planning algorithms, and that explicit models are more amenable to direct inspection. It is also possible to combine these two forms of modelling, e.g. Albrecht et al. (2015a) combine expert algorithms with type-based reasoning (cf. Section 4.2).

#### 4.8.2. Hypothesis Testing for Agent Models

Agent models may make incorrect or inaccurate predictions. This is one of the main motivations of type-based reasoning methods (Section 4.2), which consider a set of alternative models and compute Bayesian posteriors to find the most accurate model. However, such Bayesian methods generally cannot tell us about the *correctness* of models, since the posteriors quantify a relative likelihood of models but not absolute truth. Thus, even if all probability points to one model, that model may still be almost arbitrarily incorrect in that it merely has to support the observations, i.e. assign non-zero probabilities. An alternative approach is to view a model as a *hypothesis* and to decide, based on the observations, whether or not to reject the model. For agent models that predict actions, this question can be decided using methods for statistical hypothesis testing. For example, agents have been proposed which maintain models of action frequencies of other agents and conduct hypothesis tests over these models by comparing their predicted action probabilities with the average action frequencies over some window of past actions (Chakraborty and Stone, 2014; Conitzer and Sandholm, 2007; Foster and Young, 2003). Albrecht and Ramamoorthy (2015) propose an efficient sampling-based algorithm which uses "score functions" to compute test statistics from observations and learns the test distribution during the interaction, based on which a frequentist hypothesis test is performed. Given such methods, if an agent persistently rejects a model, it may decide to change the model (e.g. by using a different learning method) or to resort to some kind of default policy such as a minimax strategy (Von Neumann and Morgenstern, 1944).

#### 4.8.3. Using Models Safely

An agent can utilise models of other agents by incorporating the models' predictions into the agent's planning process. For example, if a model predicts the actions of another agent, then these predictions can be used directly by a planner to evaluate different courses of actions, resulting in an action policy that is strictly optimised with respect to the model. A potential problem with this approach is that the computed policy may be exploitable by other agents if the used agent models are inaccurate. To address this issue, several methods have been proposed which compute

| Paper | Agents | | | | | Environment | | |
|---|---|---|---|---|---|---|---|---|
| | Stochastic actions? | Changing behaviour? | Factors known? | Independent agents? | Common goals? | Move order | State/action representation | State/action observability |
| (Hernandez-Leal et al., 2017) | yes | yes | yes | yes | no | simult. | discrete | full |
| (He et al., 2016) | yes | no*** | yes | no | no | simult. | mixed | full |
| (Albrecht and Ramamoorthy, 2015) | yes | yes | no | yes | no | –* | mixed/disc. | full |
| (Albrecht et al., 2015a) | yes | no | yes | yes | no | simult. | discrete | full |
| (Bard et al., 2013) | yes | no | no | yes | no | altern. | discrete | partial/full |
| (Wang et al., 2011) | yes | yes | yes | yes | no | simult. | discrete | full |
| (Illobre et al., 2010) | no | no | yes | yes | no | simult. | mixed/disc. | full |
| (Johanson and Bowling, 2009) | yes | no | no | yes | no | altern. | discrete | partial/full |
| (Johanson et al., 2008) | yes | no | no | yes | no | altern. | discrete | partial/full |
| (Conitzer and Sandholm, 2007) | yes | yes | yes | yes | no | simult. | –**/discrete | full |
| (Hoehn et al., 2005) | yes | no | no | yes | no | altern. | discrete | partial/full |
| (Markovitch and Reger, 2005) | no | no | no | yes | no | simult. | discrete | full |
| (McCracken and Bowling, 2004) | yes | yes | no | yes | no | simult. | –**/discrete | full |
| (Guerra-Hernández et al., 2004) | no | yes | no | yes | no | –* | discrete | full |
| (Foster and Young, 2003) | yes | yes | no | yes | no | simult. | –**/discrete | full |
| (Stone et al., 2000) | yes | no | no | yes | no | simult. | mixed/disc. | partial |
| (Carmel and Markovitch, 1996b) | no | no | yes | yes | no | altern. | discrete | full |

Table 9: Assumptions in papers for other relevant methods. *Does not define move order. **Does not define environment states. ***Modelled agent may change behaviour between episodes but not during episode.

"safe" (or "robust") best-response policies to models. These methods often use a parameter of the form $\delta \in [0, 1]$ which regulates a tradeoff between safety and exploitability, such that one extreme corresponds to strict optimisation with respect to the agent models (optimal if models correct, but exploitable otherwise) and the other extreme corresponds to choosing a safe policy which may not achieve optimal performance but is less exploitable (e.g. minimax). For example, Wang et al. (2011) model an opponent as a space of models in the proximity of the empirical frequency model, with distance bounded by $\delta$, and compute a best-response against the worst-case model from this space. Other examples of safe/robust best-response methods include the works of Johanson and Bowling (2009); Johanson et al. (2008); McCracken and Bowling (2004); Carmel and Markovitch (1996b). A related idea is the use of "ideal" agent models (Stone et al., 2000). For example, Markovitch and Reger (2005) propose to learn the weaknesses of an opponent, which are defined as states in which the opponent deviates from some ideal "teacher" policy.

## 5. Open Problems

We conclude our survey by discussing nine open problems which we believe have not been sufficiently addressed in the literature and may provide fruitful avenues of future research.

### *5.1. Synergistic Combination of Modelling Methods*

This survey has outlined a landscape of methodologies, each with their individual purposes, strengths, and weaknesses. An interesting and relatively unexplored question is how these methods might be combined to complement their strengths and weaknesses. As an example, type-based reasoning methods have been combined with policy reconstruction methods, where the former allow for fast initial adaptation while the latter generate new types during the interaction (Albrecht

and Ramamoorthy, 2013; Barrett et al., 2011). These examples use a modular combination, by encapsulating the policy reconstruction methods into a special kind of type. In the long-term, an important question is whether we can find a single representation and approach that can naturally generate various modelling capabilities, including the ones discussed in this survey, such that the modelling processes synergistically inform one another. We believe there is much ground for fertile research investigating such combinations and approaches.

### *5.2. Policy Reconstruction under Partial Observability*

Many domains are characterised by partial observability, in which agents receive incomplete and uncertain observations about the environment and the actions of other agents (cf. Section 3). The existence of partial observability can make the modelling task significantly more difficult, since a modelling agent now has to take into account the possibility of incorrect and/or missing information. Different symbolic and probabilistic approaches have been proposed to deal with partial observability, especially in methods for classification, plan recognition, recursive/epistemic reasoning, and graphical models. However, as can be seen in Table 2, relatively little work exists on the problem of learning models of agent behaviours (i.e. policy reconstruction) under partial observability conditions, with most efforts focusing on extensive form games with incomplete information (e.g. Poker). Moreover, existing methods often assume that observation probabilities can be derived via provided domain knowledge (e.g. Panella and Gmytrasiewicz, 2017; Southey et al., 2005). Thus, additional research is needed for the development of methods which can effectively reconstruct behaviour models under partial observability, and methods which can deal with partial observability in the absence of domain knowledge.

### *5.3. Safe and Efficient Model Exploration*

Agents that model other agents can consider the possibility of taking actions so as to explore certain aspects of the other agents' behaviours, and in the process gain new information which may lead to better model predictions. However, such actions may carry a risk in that they may modify the behaviour of the modelled agents in unintended ways. Although the importance of safe model exploration was recognised almost 20 years ago (Carmel and Markovitch, 1998a), it has since received relatively little attention in the community.[13] Current solutions are based on look-ahead exploration to estimate the value of information of available actions (Albrecht et al., 2016; Chalkiadakis and Boutilier, 2003; Carmel and Markovitch, 1999). However, the exponential complexity of such methods makes them intractable in complex settings, indicating the need for new, more efficient approaches for safe model exploration. Closely related areas are active learning (Settles, 2012), preference elicitation (Boutilier, 2002; Chajewska et al., 2000), and Bayesian experimental design (Chaloner and Verdinelli, 1995). However, these problems usually assume that the cost of experiments/queries and their possible outcomes are known beforehand, while in our case the (long-term) cost of exploratory actions are initially unknown and there may be no crisp definition of "outcomes".

### *5.4. Efficient Discovery of Decision Factors*

Closely related to safe model exploration, it remains a significant open question how to efficiently and effectively discover the relevant factors in an agent's decision making (cf. Section 3).

[13] Indeed, the vast majority of current plan recognition methods assume that the modelling agent does not interact at all with the modelled agents (cf. Section 4.4).

Current methods either assume that this knowledge is given, include all possible decision factors in the model, or engage in an exhaustive combinatorial search to identify the relevant factors (cf. Section 4.1.1). However, these approaches are bound to be intractable or inefficient in complex, realistic applications that involve large numbers of decision factors (such as long interaction histories and high-dimensional state descriptions). Hence, more research is needed to develop methods which can efficiently discover the relevant decision factors in an agent's decision making.

### *5.5. Computationally Efficient Implementations*

Modelling methods are part of a larger agent architecture which may include many other elements, such as modules for perception (e.g. vision, natural language), communication, and planning. In domains such as commercial video games, the system will in addition have to graphically render the game world and simulate its physics (Millington and Funge, 2009). All of these additional elements can be computationally expensive. As a result, the task of modelling other agents will usually be allocated only a small fraction of the available computational resources. Therefore, to be useful in practice, modelling methods need highly efficient implementations, similar to other recent applications (Silver et al., 2016; Bowling et al., 2015). Efficient implementations may include the use of efficient data structures, parallel computing architectures, and iterative model updates which process only new observations rather than re-processing past observations. Such implementation issues have received relatively little attention in the literature, thus additional research is needed to develop efficient implementations.

### *5.6. Modelling Changing Behaviours*

A common assumption still found in many modelling methods is that the modelled agent, in particular its behaviour, will not change during the course of the interaction (cf. Section 3). However, such an assumption is easily violated in applications in which other agents may learn and adapt, and especially in interactions with humans. Modelling changing behaviours is notoriously difficult due to the essentially unconstrained nature of what other agents may do in the future. Some methods attempt to address this issue by allowing for varying degrees of changing behaviours, such as that behaviours must converge in the limit (Conitzer and Sandholm, 2007), that agents may switch periodically between different stationary behaviours (Hernandez-Leal et al., 2017; Bard and Bowling, 2007), by defining behaviours as blackbox mappings over the entire interaction history (Albrecht et al., 2016), or by prioritising recent observations over past ones (Albrecht and Ramamoorthy, 2013; Billings et al., 2004). Still, many methods are unable to deal with changing behaviours, especially methods for classification, plan recognition, and recursive reasoning. Hence, the design of methods which can effectively learn to identify, track, and predict changing behaviours remains a significant open problem, one which will be a crucial element in the quest for full autonomy.

### *5.7. Modelling with Action Duration*

The vast majority of surveyed methods (with the exception of some plan recognition methods; cf. Section 4.4) assume that actions have instant effects, meaning that actions are completed immediately after they are taken. Even in domains such as robot soccer, where actions such as passing the ball from one player to another have durations, current modelling methods work at a level of abstraction that renders such actions as though they have instant effects (e.g. Bombini et al., 2010; Kaminka et al., 2002a). It is not clear if existing modelling methods require non-trivial modification to handle actions with durations, or if this can be addressed sufficiently via such

action abstractions. In fact, it is unclear if the notion of action duration may be better viewed as an issue of activity recognition, which is the task of inferring action labels from state data and usually takes place at a lower abstraction level than the modelling methods surveyed in this article (cf. Section 3). Given that many realistic applications involve actions with durations, we believe that such questions will require further research and clarification.

### *5.8. Modelling in Open Multiagent Systems*

Virtually all of the surveyed works in this article assume *closed* multiagent systems, in which the number of agents in the system remains constant throughout the interaction, and all agents begin the interaction at the same time. This is in contrast to *open* multiagent systems, in which agents may enter and leave the system at any time during the interaction, without necessarily notifying other agents. Many important applications are characterised by such openness, such as ad-hoc wireless networks (Royer and Toh, 1999) and web-based systems for collaborative computing (Miorandi et al., 2014). In addition, a fully autonomous agent engaged in lifelong learning (Hawasly and Ramamoorthy, 2013) may itself enter and leave many multiagent systems. While some works investigated modelling other agents in open multiagent systems (Chandrasekaran et al., 2016; Huynh et al., 2006; Rovatsos et al., 2003), it remains a significant open challenge to develop efficient modelling methods for such systems. Transfer learning, which is the process of reusing past experiences to improve learning in new tasks, could be a useful element in such methods (e.g. Barrett et al., 2013).

### *5.9. Autonomous Model Contemplation and Revision*

While the methods discussed in this survey enable an autonomous agent to reason about other agents in highly sophisticated ways, they do not generally tell the agent if the used methods are the right ones in any given setting. As a result, it is possible that the agent may use inadequate and possibly misleading models of other agents, without ever realising it. For example, learning-based methods for policy reconstruction are usually restricted by the structure of the model (e.g. decision trees, finite state automata) but do not tell the modelling agent if the model structure is even capable of capturing an agent's behaviour. Type-based reasoning can utilise a space of models, but the Bayesian beliefs do not generally tell an agent if the model space is sufficient. Methods for plan recognition that use plan libraries suffer from essentially the same limitation (cf. Section 4.4.2). To detect such insufficiencies, a modelling agent requires the ability to introspectively reason about the adequacy and correctness of its modelling processes, and ultimately the ability to autonomously revise its model structures and modelling processes. Statistical hypothesis testing can be used to reason about the incorrectness of models (cf. Section 4.8.2), but such methods do not tell us *why* a model is incorrect and *how* it may be revised. In fact, it is likely that the conventional notion of correctness is too strict, and that different notions of adequacy (such as the degree to which a model allows the modelling agent to complete its task) may be needed. The current generation of intelligent agents fall short of full autonomy in part because they lack the ability to contemplate such questions, and we believe there is much research to be done to address these issues.

## 6. Conclusion

This survey identified seven major methodologies for agents modelling other agents. Surveyed methods include policy reconstruction, which seeks to reconstruct an agent's decision making based on its observed actions; type-based reasoning, which maintains beliefs over a space of

alternative decision-making models to identify the most likely models based on observed actions; classification methods, which use machine learning to predict additional properties of interest such as behaviour classes and agent identities; plan recognition, which seeks to identify an agent's goals and plans using hierarchical action descriptions or domain models; recursive reasoning, which predicts an agent's actions by modelling its beliefs and the beliefs it ascribes to other agents; graphical models, which utilise graph structures to represent detailed dependence relations in an agent's decision making; and group modelling, in which models make joint predictions about a group of agents rather than single agents. We also covered other relevant methods, including implicit modelling, hypothesis testing for agent models, and safe best-response methods. Finally, we identified a number of open problems which can provide fertile grounds for future research. Our survey of the literature shows that there exists a very large body of work on the topic of agents modelling other agents, broadly addressing questions of algorithmic design, experimental evaluation, theoretical guarantees, computational complexity, and observational constraints. As research in artificial intelligence continues to pursue the goal of creating autonomous agents that interact with other agents to accomplish tasks in complex dynamic domains, we expect to see continued development towards addressing these questions. Our hope is that this survey will contribute to this continued development by summarising the current state of research and exposing important open problems.

**Acknowledgements**

This survey benefited from comments and suggestions of many colleagues, which we would like to thank here: Michael Rovatsos, Nolan Bard, Michael Littman, Karl Tuyls, Christopher Geib, Subramanian Ramamoorthy, Alex Lascarides, Gal Kaminka, and three anonymous reviewers. This work took place in the Learning Agents Research Group (LARG) at The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-1637736, IIS-1651089, IIS-1724157), Intel, Raytheon, and Lockheed Martin. Stefano Albrecht is supported by a Feodor Lynen Research Fellowship from the Alexander von Humboldt Foundation. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by The University of Texas at Austin in accordance with its policy on objectivity in research.

## Appendix A. Clarification for Assumption Tables

Tables 2–9 list assumptions for each surveyed paper in the corresponding sections. Assumptions are in the order in which they are discussed in Section 3. The first five assumptions concern the agents to be modelled and include:

(1) whether they make stochastic or deterministic action choices

(2) whether they have changing or non-changing behaviours

(3) whether their relevant decision factors are a priori known

(4) whether they make independent or correlated action choices

(5) whether they have common or conflicting goals

The last three assumptions concern the environment within which the interaction takes place and include:

(6) the order in which agents take actions (simultaneous, alternating)

(7) the representation used for environment states and actions (discrete, continuous, mixed)

(8) the observability of environment states and actions (full, partial)

For assumptions (7) and (8), we may distinguish between states and actions by using a "state/action" notation. Additional comments are provided in the table captions.

We note that while many works state all or most of the above assumptions explicitly, there are also many works which are rather vague about some assumptions. In vague cases, we tried to infer assumptions based on our understanding of the provided descriptions.

## References


Abdul-Rahman, A., Hailes, S., 2000. Supporting trust in virtual communities. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. IEEE.

Ahmadi, M., Lamjiri, A., Nevisi, M., Habibi, J., Badie, K., 2003. Using a two-layered case-based reasoning for prediction in soccer coach. In: Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications. pp. 181–185.

Albrecht, D., Zukerman, I., Nicholson, A., 1998. Bayesian models for keyhole plan recognition in an adventure game. User Modeling and User-Adapted Interaction 8 (1), 5–47.

Albrecht, D., Zukerman, I., Nicholson, A., Bud, A., 1997. Towards a Bayesian model for keyhole plan recognition in large domains. In: User Modeling: Proceedings of the Sixth International Conference. Springer, pp. 365–376.

Albrecht, S., Crandall, J., Ramamoorthy, S., 2015a. E-HBA: Using action policies for expert advice and agent typification. In: AAAI'15 Workshop on Multiagent Interaction without Prior Coordination.

Albrecht, S., Crandall, J., Ramamoorthy, S., 2015b. An empirical study on the practical impact of prior beliefs over policy types. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. pp. 1988–1994.

Albrecht, S., Crandall, J., Ramamoorthy, S., 2016. Belief and truth in hypothesised behaviours. Artificial Intelligence 235, 63–94.

Albrecht, S., Liemhetcharat, S., Stone, P., 2017. Special issue on multiagent interaction without prior coordination: Guest editorial. Autonomous Agents and Multi-Agent Systems 31 (4), 765–766.
URL http://dx.doi.org/10.1007/s10458-016-9358-0

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

Albrecht, S., Ramamoorthy, S., 2013. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. Tech. rep., School of Informatics, The University of Edinburgh.
URL http://arxiv.org/abs/1506.01170

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

Albrecht, S., Ramamoorthy, S., 2015. Are you doing what I think you are doing? Criticising uncertain agent models. In: Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence. pp. 52–61.

Albrecht, S., Stone, P., 2017. Reasoning about hypothetical agent behaviours and their parameters. In: Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems. pp. 547–555.

Albrecht, S. V., Ramamoorthy, S., 2016. Exploiting causality for selective belief filtering in dynamic Bayesian networks. Journal of Artificial Intelligence Research 55, 1135–1178.

Alonso, E., D'Inverno, M., Kudenko, D., Luck, M., Noble, J., 2001. Learning in multi-agent systems. The Knowledge Engineering Review 16 (3), 277–284.

Anderson, J., Boyle, C., Corbett, A., Lewis, M., 1990. Cognitive modeling and intelligent tutoring. Artificial Intelligence 42 (1), 7–49.

Aumann, R., 1974. Subjectivity and correlation in randomized strategies. Journal of mathematical Economics 1, 67–96.

Avrahami-Zilberbrand, D., Kaminka, G., 2005. Fast and complete symbolic plan recognition. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. pp. 653–658.

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

Avrahami-Zilberbrand, D., Kaminka, G., Zarosim, H., 2005. Fast and complete symbolic plan recognition: Allowing for duration, interleaved execution, and lossy observations. In: IJCAI'05 Workshop on Modeling Others from Observations.

Baarslag, T., Hendrikx, M., Hindriks, K., Jonker, C., 2016. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. Autonomous Agents and Multi-Agent Systems 30 (5), 849–898.

Baker, C., Saxe, R., Tenenbaum, J., 2009. Action understanding as inverse planning. Cognition 113 (3), 329–349.

Baker, C., Saxe, R., Tenenbaum, J., 2011. Bayesian theory of mind: Modeling joint belief-desire attribution. In: Proceedings of the Cognitive Science Society. pp. 2469–2474.

Baker, C., Tenenbaum, J., Saxe, R., 2005. Bayesian models of human action understanding. In: Proceedings of the 18th International Conference on Neural Information Processing Systems. pp. 99–106.

Bakkes, S., Spronck, P., van Lankveld, G., 2012. Player behavioural modelling for video games. Entertainment Computing 3 (3), 71–79.

Banerjee, B., Kraemer, L., 2011. Branch and price for multi-agent plan recognition. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence. pp. 601–607.

Banerjee, B., Kraemer, L., Lyle, J., 2010. Multi-agent plan recognition: Formalization and algorithms. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence. pp. 1059–1064.

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

Bard, N., Bowling, M., 2007. Particle filtering for dynamic agent modelling in simplified poker. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence. pp. 515–521.

Bard, N., Johanson, M., Burch, N., Bowling, M., 2013. Online implicit agent modelling. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems. pp. 255–262.

Baré, M., Canamero, D., Delannoy, J., Kodratoff, Y., 1994. XPlans: Case-based reasoning for plan recognition. Applied Artificial Intelligence 8 (4), 617–643.

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

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

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

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

Bengio, Y., Frasconi, P., 1995. An input output HMM architecture. In: Advances in Neural Information Processing Systems 8. pp. 427–434.

Billings, D., Davidson, A., Schauenberg, T., Burch, N., Bowling, M., Holte, R., Schaeffer, J., Szafron, D., 2004. Game-tree search with adaptation in stochastic imperfect-information games. Proceedings of the 4th International Conference on Computers and Games, 21–34.

Blaylock, N., Allen, J., 2003. Corpus-based, statistical goal recognition. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence. pp. 1303–1308.

Blaylock, N., Allen, J., 2004. Statistical goal parameter recognition. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling. pp. 297–304.

Blaylock, N., Allen, J., 2006. Fast hierarchical goal schema recognition. In: Proceedings of the 21st AAAI National Conference on Artificial Intelligence. pp. 796–801.

Bloembergen, D., Tuyls, K., Hennes, D., Kaisers, M., 2015. Evolutionary dynamics of multi-agent learning: A survey. Journal of Artificial Intelligence Research 53, 659–697.

Bolander, T., Andersen, M., 2011. Epistemic planning for single- and multi-agent systems. Journal of Applied Non-Classical Logics 21 (1), 9–33.

Bombini, G., Di Mauro, N., Ferilli, S., Esposito, F., 2010. Classifying agent behaviour through relational sequential patterns. Agent and Multi-Agent Systems: Technologies and Applications, 273–282.

Borck, H., Karneeb, J., Alford, R., Aha, D., 2015. Case-based behavior recognition in beyond visual range air combat. In: Proceedings of the 28th International Florida Artificial Intelligence Research Society Conference. pp. 379–384.

Boutilier, C., 2002. A POMDP formulation of preference elicitation problems. In: Proceedings of the 18th National Conference on Artificial Intelligence. pp. 239–246.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., Poole, D., 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research 21, 135–191.

Bowling, M., Burch, N., Johanson, M., Tammelin, O., 2015. Heads-up limit hold'em poker is solved. Science 347 (6218), 145–149.

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

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

Boyen, X., Koller, D., 1998. Tractable inference for complex stochastic processes. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. pp. 33–42.

Brown, G., 1951. Iterative solution of games by fictitious play. In: Proceedings of the Conference on Activity Analysis of Production and Allocation, Cowles Commission Monograph 13. pp. 374–376.
Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S., 2012. A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games 4 (1), 1–43.
Buehler, M., Iagnemma, K., Singh, S., 2009. The DARPA urban challenge: autonomous vehicles in city traffic. In: Springer Tracts in Advanced Robotics 56. Springer.
Bui, H., Venkatesh, S., West, G., 2002. Policy recognition in the abstract hidden Markov model. Journal of Artificial Intelligence Research 17, 451–499.
Busoniu, L., Babuska, R., De Schutter, B., 2008. A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C 38 (2).
Cadilhac, A., Asher, N., Benamara, F., Lascarides, A., 2013. Grounding strategic conversation: Using negotiation dialogues to predict trades in a win-lose game. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 357–368.
Camerer, C., Ho, T., Chong, J., 2004. A cognitive hierarchy model of games. The Quarterly Journal of Economics 119 (3), 861–898.
Camerer, C., Ho, T., Chong, J., 2015. A psychological approach to strategic thinking in games. Current Opinion in Behavioral Sciences 3, 157–162.
Campbell, M., Marsland, T., 1983. A comparison of minimax tree search algorithms. Artificial Intelligence 20 (4), 347–367.
Carberry, S., 2001. Techniques for plan recognition. User Modeling and User-Adapted Interaction 11 (1-2), 31–48.
Carmel, D., Markovitch, S., 1993. Learning models of opponent's strategy in game playing. In: Proceedings of the AAAI Fall Symposium Series. Games: Planning and Learning. pp. 140–147.
Carmel, D., Markovitch, S., 1996a. Incorporating opponent models into adversary search. In: Proceedings of the 13th National Conference on Artificial Intelligence. pp. 120–125.
Carmel, D., Markovitch, S., 1996b. Learning and using opponent models in adversary search. Tech. rep., Computer Science Department, Technion. Technical Report CIS9606.
Carmel, D., Markovitch, S., 1996c. Learning models of intelligent agents. In: Proceedings of the 13th AAAI National Conference on Artificial Intelligence. pp. 62–67.
Carmel, D., Markovitch, S., 1998a. How to explore your opponent's strategy (almost) optimally. In: Proceedings of the International Conference on Multi Agent Systems. IEEE, pp. 64–71.
Carmel, D., Markovitch, S., 1998b. Model-based learning of interaction strategies in multi-agent systems. Journal of Experimental & Theoretical Artificial Intelligence 10 (3), 309–332.
Carmel, D., Markovitch, S., 1999. Exploration strategies for model-based learning in multi-agent systems. Autonomous Agents and Multi-Agent Systems 2 (2), 141–172.
Chajewska, U., Koller, D., Ormoneit, D., 2001. Learning an agent's utility function by observing behavior. In: Proceedings of the 18th International Conference on Machine Learning. pp. 35–42.
Chajewska, U., Koller, D., Parr, R., 2000. Making rational decisions using adaptive utility elicitation. In: Proceedings of the 17th National Conference on Artificial Intelligence. pp. 363–369.
Chakraborty, D., Stone, P., 2013. Cooperating with a Markovian ad hoc teammate. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems. pp. 1085–1092.
Chakraborty, D., Stone, P., 2014. Multiagent learning in the presence of memory-bounded agents. Autonomous Agents and Multi-Agent Systems 28 (2), 182–213.
Chalkiadakis, G., Boutilier, C., 2003. Coordination in multiagent reinforcement learning: a Bayesian approach. In: Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems. pp. 709–716.
Chaloner, K., Verdinelli, I., 1995. Bayesian experimental design: A review. Statistical Science, 273–304.
Chandrasekaran, M., Eck, A., Doshi, P., Soh, L., 2016. Individual planning in open and typed agent systems. In: Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence. pp. 82–91.
Charniak, E., Goldman, R., 1993. A Bayesian model of plan recognition. Artificial Intelligence 64 (1), 53–79.
Claus, C., Boutilier, C., 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the 15th National Conference on Artificial Intelligence. pp. 746–752.
Coehoorn, R., Jennings, N., 2004. Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In: Proceedings of the 6th International Conference on Electronic Commerce. ACM, pp. 59–68.
Cohen, P., Levesque, H., 1991. Teamwork. Nous 25 (4), 487–512.
Cohen, P., Perrault, C., Allen, J., 1981. Beyond question answering. In: Lehnert, W., Ringle, M. (Eds.), Strategies for Natural Language Processing. Taylor & Fancis Group, pp. 245–274.
Conitzer, V., Sandholm, T., 2007. AWESOME: a general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. Machine Learning 67 (1-2), 23–43.
Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine Learning 20 (3), 273–297.
Crandall, J., 2014. Towards minimizing disappointment in repeated games. Journal of Artificial Intelligence Research 49,

111–142.
Dasgupta, P., 2000. Trust as a commodity. Trust: Making and Breaking Cooperative Relations 4, 49–72.
Davidson, A., Billings, D., Schaeffer, J., Szafron, D., 2000. Improved opponent modeling in poker. In: Proceedings of the International Conference on Artificial Intelligence. pp. 1467–1473.
Davison, B., Hirsh, H., 1998. Predicting sequences of user actions. In: AAAI/ICML'98 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis.
de Farias, D., Megiddo, N., 2004. Exploration-exploitation tradeoffs for experts algorithms in reactive environments. In: Advances in Neural Information Processing Systems 17. pp. 409–416.
de Weerd, H., Verbrugge, R., Verheij, B., 2013. How much does it help to know what she knows you know? an agent-based simulation study. Artificial Intelligence 199, 67–92.
de Weerd, H., Verbrugge, R., Verheij, B., 2017. Negotiating with other minds: the role of recursive theory of mind in negotiation with incomplete information. Autonomous Agents and Multi-Agent Systems 31 (2), 250–287.
Dean, T., Kanazawa, K., 1989. A model for reasoning about persistence and causation. Computational Intelligence 5, 142–150.
Dekel, E., Fudenberg, D., Levine, D., 2004. Learning to play Bayesian games. Games and Economic Behavior 46 (2), 282–303.
Denzinger, J., Hamdan, J., 2004. Improving modeling of other agents using tentative stereotypes and compactification of observations. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology. pp. 106–112.
Doshi, P., Chandrasekaran, M., Zeng, Y., 2010. Epsilon-subjective equivalence of models for interactive dynamic influence diagrams. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. Vol. 2. IEEE, pp. 165–172.
Doshi, P., Gmytrasiewicz, P., 2009. Monte carlo sampling methods for approximating interactive POMDPs. Journal of Artificial Intelligence Research, 297–337.
Doshi, P., Perez, D., 2008. Generalized point based value iteration for interactive POMDPs. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence. pp. 63–68.
Doshi, P., Zeng, Y., Chen, Q., 2009. Graphical models for interactive POMDPs: representations and solutions. Autonomous Agents and Multi-Agent Systems 18 (3), 376–416.
Doucet, A., De Freitas, N., Murphy, K., Russell, S., 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. pp. 176–183.
Erdogan, C., Veloso, M., 2011. Action selection via learning behavior patterns in multi-robot domains. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 192–197.
Fagan, M., Cunningham, P., 2003. Case-based plan recognition in computer games. In: International Conference on Case-Based Reasoning. Springer, pp. 161–170.
Fagundes, M., Meneguzzi, F., Bordini, R., Vieira, R., 2014. Dealing with ambiguity in plan recognition under time constraints. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems. pp. 389–396.
Fern, A., Tadepalli, P., 2010. A computational decision theory for interactive assistants. In: Advances in Neural Information Processing Systems. pp. 577–585.
Fikes, R., Nilsson, N., 1971. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial intelligence 2 (3-4), 189–208.
Foster, D., Young, H., 2001. On the impossibility of predicting the behavior of rational agents. Proceedings of the National Academy of Sciences 98 (22), 12848–12853.
Foster, D., Young, H., 2003. Learning, hypothesis testing, and Nash equilibrium. Games and Economic Behavior 45 (1), 73–96.
Fredkin, E., 1960. Trie memory. Communications of the ACM 3 (9), 490–499.
Fudenberg, D., Levine, D., 1998. The Theory of Learning in Games. Vol. 2. MIT Press.
Fürnkranz, J., 2001. Machine learning in games: A survey. In: Fürnkranz, J., Kubat, M. (Eds.), Machines That Learn to Play Games. Nova Science Publishers, Ch. 2, pp. 11–59.
Gal, Y., Pfeffer, A., 2003a. A language for modeling agents' decision making processes in games. In: Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems. ACM, pp. 265–272.
Gal, Y., Pfeffer, A., 2003b. A language for opponent modeling in repeated games. In: AAMAS'03 Workshop on Game Theory and Decision Theory.
Gal, Y., Pfeffer, A., 2008. Networks of influence diagrams: A formalism for representing agents' beliefs and decision-making processes. Journal of Artificial Intelligence Research 33 (1), 109–147.
Gal, Y., Pfeffer, A., Marzo, F., Grosz, B., 2004. Learning social preferences in games. In: Proceedings of the 19th AAAI National Conference on Artificial Intelligence. pp. 226–231.
Ganzfried, S., Sandholm, T., 2011. Game theory-based opponent modeling in large imperfect-information games. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems. pp. 533–540.

Geib, C., 2004. Assessing the complexity of plan recognition. In: Proceedings of the 19th AAAI National Conference on Artificial Intelligence. pp. 507–512.
Geib, C., Goldman, R., 2001. Plan recognition in intrusion detection systems. In: Proceedings of the 2nd DARPA Information Survivability Conference and Exposition. pp. 329–342.
Geib, C., Goldman, R., 2009. A probabilistic plan recognition algorithm based on plan tree grammars. Artificial Intelligence 173 (11), 1101–1132.
Geib, C., Steedman, M., 2007. On natural language processing and plan recognition. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence. pp. 1612–1617.
Ghaderi, H., Levesque, H., Lespérance, Y., 2007. A logical theory of coordination and joint ability. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence. pp. 421–426.
Gmytrasiewicz, P., Doshi, P., 2005. A framework for sequential planning in multiagent settings. Journal of Artificial Intelligence Research 24 (1), 49–79.
Gmytrasiewicz, P., Durfee, E., 1995. A rigorous, operational formalization of recursive modeling. In: Proceedings of the 1st International Conference on Multiagent Systems. pp. 125–132.
Gmytrasiewicz, P., Durfee, E., 2000. Rational coordination in multi-agent environments. Autonomous Agents and Multi-Agent Systems 3 (4), 319–350.
Gmytrasiewicz, P., Durfee, E., Wehe, D., 1991. A decision-theoretic approach to coordinating multi-agent interactions. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence. pp. 63–68.
Gmytrasiewicz, P., Noh, S., Kellogg, T., 1998. Bayesian update of recursive agent models. User Modeling and User-Adapted Interaction 8 (1), 49–69.
Gold, E., 1978. Complexity of automaton identification from given data. Information and Control 37 (3), 302–320.
Gold, K., 2010. Training goal recognition online from low-level inputs in an action-adventure game. In: Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. pp. 21–26.
Goodie, A., Doshi, P., Young, D., 2012. Levels of theory-of-mind reasoning in competitive games. Journal of Behavioral Decision Making 25 (1), 95–108.
Grosz, B., Kraus, S., 1996. Collaborative plans for complex group action. Artificial Intelligence 86 (2), 269–357.
Grosz, B., Sidner, C., 1986. Attention, intentions, and the structure of discourse. Computational Linguistics 12 (3), 175–204.
Guerra-Hernández, A., El Fallah-Seghrouchni, A., Soldano, H., 2004. Learning in BDI multi-agent systems. In: Computational Logic in Multi-Agent Systems. Springer, pp. 218–233.
Hammond, K., 1986. CHEF: A model of case-based planning. In: Proceedings of the 5th AAAI National Conference on Artificial Intelligence. pp. 267–271.
Harsanyi, J., 1962. Bargaining in ignorance of the opponent's utility function. Journal of Conflict Resolution 6 (1), 29–38.
Harsanyi, J., 1967. Games with incomplete information played by "Bayesian" players. Part I. The basic model. Management Science 14 (3), 159–182.
Harsanyi, J., 1968a. Games with incomplete information played by "Bayesian" players. Part II. Bayesian equilibrium points. Management Science 14 (5), 320–334.
Harsanyi, J., 1968b. Games with incomplete information played by "Bayesian" players. Part III. The basic probability distribution of the game. Management Science 14 (7), 486–502.
Hart, S., Mas-Colell, A., 2001. A reinforcement procedure leading to correlated equilibrium. Economic Essays: A Festschrift for Werner Hildenbrand, 181–200.
Hausknecht, M., Mupparaju, P., Subramanian, S., Kalyanakrishnan, S., Stone, P., 2016. Half field offense: An environment for multiagent learning and ad hoc teamwork. In: AAMAS'16 Workshop on Adaptive Learning Agents.
Hawasly, M., Ramamoorthy, S., 2013. Lifelong transfer learning with an option hierarchy. In: International Conference on Intelligent Robots and Systems. IEEE, pp. 1341–1346.
He, H., Boyd-Graber, J., Kwok, K., Daumé III, H., 2016. Opponent modeling in deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning. pp. 1804–1813.
Hedden, T., Zhang, J., 2002. What do you think i think you think?: Strategic reasoning in matrix games. Cognition 85 (1), 1–36.
Hernandez-Leal, P., Kaisers, M., Baarslag, T., de Cote, E. M., 2017. A survey of learning in multiagent environments: Dealing with non-stationarity. CoRR https://arxiv.org/abs/1707.09183.
Hernandez-Leal, P., Zhan, Y., Taylor, M., Sucar, L., de Cote, E., 2017. Efficiently detecting switches against non-stationary opponents. Autonomous Agents and Multi-Agent Systems 31 (4), 767–789.
Hindriks, K., Tykhonov, D., 2008. Opponent modelling in automated multi-issue negotiation using Bayesian learning. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems. pp. 331–338.
Hoang, T., Low, K., 2013. Interactive POMDP lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence. pp. 2298–2305.
Hoehn, B., Southey, F., Holte, R., Bulitko, V., 2005. Effective short-term opponent exploitation in simplified poker. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. pp. 783–788.

Hong, J., 2000. Graph construction and analysis as a paradigm for plan recognition. In: Proceedings of the 17th National Conference on Artificial Intelligence. pp. 774–779.
Hong, J., 2001. Goal recognition through goal graph analysis. Journal of Artificial Intelligence Research 15, 1–30.
Horst, R., Pardalos, P., Thoai, N., 2000. Introduction to Global Optimization. Kluwer Academic Publishers.
Howard, R., 1966. Information value theory. IEEE Transactions on Systems Science and Cybernetics 2 (1), 22–26.
Howard, R., Matheson, J., 1984. Influence diagrams. In: Howard, R., Matheson, J. (Eds.), Readings on the Principles and Applications of Decision Analysis. Vol. 2. Strategic Decisions Group, pp. 719–762.
Howard, R., Matheson, J., 2005. Influence diagrams. Decision Analysis 2 (3), 127–143.
Hsieh, J., Sun, C., 2008. Building a player strategy model by analyzing replays of real-time strategy games. In: IEEE International Joint Conference on Neural Networks. pp. 3106–3111.
Huynh, T., Jennings, N., Shadbolt, N., 2006. An integrated trust and reputation model for open multi-agent systems. Autonomous Agents and Multi-Agent Systems 13 (2), 119–154.
Iglesias, J., Angelov, P., Ledezma, A., Sanchis, A., 2010. Evolving classification of agents' behaviors: a general approach. Evolving Systems 1 (3), 161–171.
Iglesias, J., Ledezma, A., Sanchis, A., Kaminka, G., 2008. Classifying efficiently the behavior of a soccer team. Intelligent Autonomous Systems 10, 316–323.
Iida, H., Kotani, Y., Uiterwijk, J., 1996. Tutoring strategies in game-tree search. Games of No Chance 29, 433–435.
Iida, H., Uiterwijk, J., van den Herik, H., Herschberg, I., 1993. Potential applications of opponent-model search. part 1: The domain of applicability. ICCA Journal 16, 201–208.
Iida, H., Uiterwijk, J., van den Herik, H., Herschberg, I., 1994. Potential applications of opponent-model search. part 2: Risks and strategies. ICCA Journal 17, 10–14.
Illobre, A., Gonzalez, J., Otero, R., Santos, J., 2010. Learning action descriptions of opponent behaviour in the Robocup 2D simulation environment. In: Proceedings of the 20th International Conference on Inductive Logic Programming. Springer, pp. 105–113.
Jarvis, P., Lunt, T., Myers, K., 2005. Identifying terrorist activity with AI plan recognition technology. AI Magazine 26 (3), 73.
Jensen, F., Nielsen, T., 2011. Probabilistic decision graphs for optimization under uncertainty. 4OR: A Quarterly Journal of Operations Research 9 (1), 1–28.
Jensen, S., Boley, D., Gini, M., Schrater, P., 2005. Rapid on-line temporal sequence prediction by an adaptive agent. In: Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems. pp. 67–73.
Johanson, M., Bowling, M., 2009. Data biased robust counter strategies. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics. pp. 264–271.
Johanson, M., Zinkevich, M., Bowling, M., 2008. Computing robust counter-strategies. In: Advances in Neural Information Processing Systems 20. pp. 721–728.
Kaelbling, L., Littman, M., Cassandra, A., 1998. Planning and acting in partially observable stochastic domains. Artificial intelligence 101 (1), 99–134.
Kalai, E., Lehrer, E., 1993. Rational learning leads to Nash equilibrium. Econometrica 61 (5), 1019–1045.
Kaminka, G., Fidanboylu, M., Chang, A., Veloso, M., 2002a. Learning the sequential coordinated behavior of teams from observations. In: RoboCup 2002: Robot Soccer World Cup VI. Springer, pp. 111–125.
Kaminka, G., Pynadath, D., Tambe, M., 2002b. Monitoring teams by overhearing: A multi-agent plan-recognition approach. Journal of Artificial Intelligence Research 17 (1), 83–135.
Karpinskyj, S., Zambetta, F., Cavedon, L., 2014. Video game personalisation techniques: A comprehensive survey. Entertainment Computing 5 (4), 211–218.
Kautz, H., Allen, J., 1986. Generalized plan recognition. In: Proceedings of the 5th National Conference on Artificial Intelligence. pp. 32–37.
Kearns, M., Littman, M., Singh, S., 2001. Graphical models for game theory. In: Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence. pp. 253–260.
Keren, S., Gal, A., Karpas, E., 2014. Goal recognition design. In: Proceedings of the 24th International Conference on Automated Planning and Scheduling. pp. 154–162.
Keren, S., Gal, A., Karpas, E., 2015. Goal recognition design for non-optimal agents. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. pp. 3298–3304.
Keren, S., Gal, A., Karpas, E., 2016. Goal recognition design with non-observable actions. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence. pp. 3152–3158.
Kerkez, B., Cox, M., 2003. Incremental case-based plan recognition with local predictions. International Journal on Artificial Intelligence Tools 12 (4), 413–463.
Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., Asada, M., 1997. The RoboCup synthetic agent challenge 97. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 24–29.
Kocsis, L., Szepesvári, C., 2006. Bandit based Monte-Carlo planning. In: Proceedings of the 17th European Conference

on Machine Learning. Springer, pp. 282–293.
Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press.
Koller, D., Milch, B., 2003. Multi-agent influence diagrams for representing and solving games. Games and Economic Behavior 45 (1), 181–221.
Kolodner, J., 2014. Case-Based Reasoning. Morgan Kaufmann.
Kominis, F., Geffner, H., 2015. Beliefs in multiagent planning: From one agent to many. In: Proceedings of the 25th International Conference on Automated Planning and Scheduling. pp. 147–155.
Kuhlmann, G., Knox, W., Stone, P., 2006. Know thine enemy: A champion RoboCup coach agent. In: Proceedings of the 21st National Conference on Artificial Intelligence. pp. 1463–1468.
La Mura, P., 2000. Game networks. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. pp. 335–342.
Lasota, P., Fong, T., Shah, J., 2014. A survey of methods for safe human-robot interaction. Foundations and Trends in Robotics 5 (4), 261–349.
Lattner, A., Miene, A., Visser, U., Herzog, O., 2005. Sequential pattern mining for situation and behavior prediction in simulated robotic soccer. In: RoboCup 2005, LNAI 4020. Springer, pp. 118–129.
Laviers, K., Sukthankar, G., Molineaux, M., Aha, D., 2009. Improving offensive performance through opponent modeling. In: Proceedings of the 5th Artificial Intelligence for Interactive Digital Entertainment Conference. pp. 58–63.
Ledezma, A., Aler, R., Sanchis, A., Borrajo, D., 2009. OMBO: an opponent modeling approach. AI Communications 22 (1), 21–35.
Lesh, N., Etzioni, O., 1995. A sound and fast goal recognizer. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. pp. 1704–1710.
Litman, D., Allen, J., 1984. A plan recognition model for clarification subdialogues. In: Proceedings of the 10th International Conference on Computational Linguistics. pp. 302–311.
Lockett, A., Chen, C., Miikkulainen, R., 2007. Evolving explicit opponent models in game playing. In: Proceedings of the 9th Conference on Genetic and Evolutionary Computation. pp. 2106–2113.
Löwe, B., Pacuit, E., Witzel, A., 2010. Planning based on dynamic epistemic logic. Tech. rep., Technical Report PP-2010-14, Institute for logic, Language and Computation, Universiteit van Amsterdam.
Markovitch, S., Reger, R., 2005. Learning and exploiting relative weaknesses of opponent agents. Autonomous Agents and Multi-Agent Systems 10 (2), 103–130.
McCalla, G., Vassileva, J., Greer, J., Bull, S., 2000. Active learner modelling. In: Proceedings of the 5th International Conference on Intelligent Tutoring Systems. pp. 53–62.
McCarthy, J., 1980. Circumscription — a form of non-monotonic reasoning. Artificial intelligence 13 (1), 27–39.
McCarthy, J., Hayes, P., 1969. Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502.
McCracken, P., Bowling, M., 2004. Safe strategies for agent modelling in games. In: AAAI Fall Symposium on Artificial Multi-agent Learning. pp. 103–110.
McTear, M., 1993. User modelling for adaptive computer systems: a survey of recent developments. Artificial Intelligence Review 7 (3), 157–184.
Mealing, R., Shapiro, J., 2017. Opponent modelling by expectation-maximisation and sequence prediction in simplified poker. IEEE Transactions on Computational Intelligence and AI in Games 9.
Milch, B., Koller, D., 2000. Probabilistic models for agents' beliefs and decisions. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. pp. 389–396.
Millington, I., Funge, J., 2009. Artificial Intelligence for Games, second edition Edition. CRC Press.
Miorandi, D., Maltese, V., Rovatsos, M., Nijholt, A., Stewart, J., 2014. Social collective intelligence: combining the powers of humans and machines to build a smarter society. Springer.
Mor, Y., Goldman, C., Rosenschein, J., 1995. Learn your opponent's strategy (in polynomial time)! In: IJCAI'95 Workshop on Adaption and Learning in Multi-Agent Systems.
Muggleton, S., 1991. Inductive logic programming. New Generation Computing 8 (4), 295–318.
Mui, L., Mohtashemi, M., Halberstadt, A., 2002. A computational model of trust and reputation. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences. IEEE, pp. 2431–2439.
Muise, C., Belle, V., Felli, P., McIlraith, S., Miller, T., Pearce, A., Sonenberg, L., 2015. Planning over multi-agent epistemic states: A classical planning approach. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. pp. 3327–3334.
Myerson, R., 1991. Game Theory: Analysis of Conflict. Harvard University Press.
Nachbar, J., 2005. Beliefs in repeated games. Econometrica 73 (2), 459–480.
Nash, J., 1950. Equilibrium points in n-person games. Proceedings of the National Academy of Sciences 36 (1), 48–49.
Ng, A., Russell, S., 2000. Algorithms for inverse reinforcement learning. In: Proceedings of the 17th International Conference on Machine Learning. pp. 663–670.
Ng, B., Boakye, K., Meyers, C., Wang, A., 2012. Bayes-adaptive interactive POMDPs. In: Proceedings of the 26th AAAI

Conference on Artificial Intelligence. pp. 1408–1414.
Nguyen, T.-H. D., Hsu, D., Lee, W. S., Leong, T.-Y., Kaelbling, L. P., Lozano-Perez, T., Grant, A. H., 2011. CAPIR: Collaborative action planning with intention recognition. In: Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. pp. 61–66.
Nielsen, T., Jensen, F., 2004. Learning a decision maker's utility function from (possibly) inconsistent behavior. Artificial Intelligence 160 (1-2), 53–78.
Nyarko, Y., 1998. Bayesian learning and convergence to Nash equilibria without common priors. Economic Theory 11 (3), 643–655.
Oh, J., Meneguzzi, F., Sycara, K., Norman, T., 2011. An agent architecture for prognostic reasoning assistance. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 2513–2518.
Olorunleke, O., McCalla, G., 2005. A condensed roadmap of agents-modelling-agents research. In: IJCAI'05 Workshop on Modeling Other Agents From Observation.
Panait, L., Luke, S., 2005. Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11 (3), 387–434.
Panella, A., Gmytrasiewicz, P., 2017. Interactive POMDPs with finite-state models of other agents. Autonomous Agents and Multi-Agent Systems.
Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann.
Pinyol, I., Sabater-Mir, J., 2013. Computational trust and reputation models for open multi-agent systems: a review. Artificial Intelligence Review 40 (1), 1–25.
Pitt, L., 1989. Inductive inference, DFAs, and computational complexity. In: International Workshop on Analogical and Inductive Inference. Springer, pp. 18–44.
Pollack, M., 1986. A model of plan inference that distinguishes between the beliefs of actors and observers. In: Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics. pp. 207–214.
Pourmehr, S., Dadkhah, C., 2012. An overview on opponent modeling in RoboCup soccer simulation 2D. In: RoboCup 2011, LNCS 7416. Springer, pp. 402–414.
Powers, R., Shoham, Y., 2005. Learning against opponents with bounded memory. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. pp. 817–822.
Pynadath, D., Wellman, M., 2000. Probabilistic state-dependent grammars for plan recognition. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. pp. 507–514.
Ramchurn, S., Huynh, D., Jennings, N., 2004. Trust in multi-agent systems. The Knowledge Engineering Review 19 (1), 1–25.
Ramírez, M., Geffner, H., 2009. Plan recognition as planning. In: Proceedings of the 21st International Joint Conference on Artifical Intelligence. pp. 1778–1783.
Ramírez, M., Geffner, H., 2010. Probabilistic plan recognition using off-the-shelf classical planners. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence. pp. 1121–1126.
Ramırez, M., Geffner, H., 2011. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 2009–2014.
Rathnasabapathy, B., Doshi, P., Gmytrasiewicz, P., 2006. Exact solutions of interactive POMDPs using behavioral equivalence. In: Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems. pp. 1025–1032.
Reibman, A., Ballard, B., 1983. Non-minimax search strategies for use against fallible opponents. In: Proceedings of the 3rd AAAI National Conference on Artificial Intelligence. pp. 338–342.
Riley, P., Veloso, M., 2000. On behavior classification in adversarial environments. In: Distributed Autonomous Robotic Systems 4. Springer, pp. 371–380.
Riley, P., Veloso, M., 2002. Recognizing probabilistic opponent movement models. In: RoboCup 2001, LNAI 2377. Springer, pp. 453–458.
Rovatsos, M., Weiß, G., Wolf, M., 2003. Multiagent learning for open systems: A study in opponent classification. In: Adaptive Agents and Multi-Agent Systems, LNAI 2636. Springer, pp. 66–87.
Royer, E., Toh, C., 1999. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Personal Communications 6 (2), 46–55.
Rubin, J., Watson, I., 2011. Computer poker: A review. Artificial Intelligence 175 (5), 958–987.
Sabater, J., Sierra, C., 2001. Regret: A reputation model for gregarious societies. In: Fourth Workshop on Deception Fraud and Trust in Agent Societies. Vol. 70. pp. 61–69.
Sadigh, D., Sastry, S., Seshia, S., Dragan, A., 2016. Information gathering actions over human internal state. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems. pp. 66–73.
Saria, S., Mahadevan, S., 2004. Probabilistic plan recognition in multiagent systems. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling. pp. 287–296.
Schadd, F., Bakkes, S., Spronck, P., 2007. Opponent modeling in real-time strategy games. In: Proceedings of the 8th Annual European GAMEON Conference. pp. 61–70.

Schillo, M., Funk, P., Rovatsos, M., 2000. Using trust for detecting deceitful agents in artificial societies. Applied Artificial Intelligence 14 (8), 825–848.
Schmid, A., Weede, O., Wörn, H., 2007. Proactive robot task selection given a human intention estimate. In: Proceedings of the 16th IEEE International Symposium on Robot and Human Interactive Communication. pp. 726–731.
Schmidt, C., Sridharan, N., Goodson, J., 1978. The plan recognition problem: An intersection of psychology and artificial intelligence. Artificial Intelligence 11 (1-2), 45–83.
Sen, S., Arora, N., 1997. Learning to take risks. In: AAAI'97 Workshop on Multiagent Learning. pp. 59–64.
Sen, S., Weiss, G., 1999. Learning in multiagent systems. In: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Ch. 6, pp. 259–298.
Settles, B., 2012. Active Learning. Morgan & Claypool Publishers.
Shachter, R., 1986. Evaluating influence diagrams. Operations Research 34 (6), 871–882.
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), 484–489.
Singh, S., Barto, A., Chentanez, N., 2005. Intrinsically motivated reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 1281–1288.
Sohrabi, S., Riabov, A., Udrea, O., 2016. Plan recognition as planning revisited. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence. pp. 3258–3264.
Sondik, E., 1971. The optimal control of partially observable Markov processes. Ph.D. thesis, Stanford University.
Sonu, E., Doshi, P., 2015. Scalable solutions of interactive POMDPs using generalized and bounded policy iteration. Autonomous Agents and Multi-Agent Systems 29 (3), 455–494.
Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., Rayner, C., 2005. Bayes' bluff: opponent modelling in poker. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence. pp. 550–558.
Spronck, P., den Teuling, F., 2010. Player modeling in Civilization IV. In: Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. pp. 180–185.
Steffens, T., 2004a. Adapting similarity measures to agent types in opponent modelling. In: AAMAS'04 Workshop on Modeling Other Agents from Observations. pp. 125–128.
Steffens, T., 2004b. Feature-based declarative opponent-modelling. In: RoboCup 2003, LNAI 3020. Springer, pp. 125–136.
Steffens, T., 2005. Similarity-based opponent modelling using imperfect domain theories. In: Proceedings of the 1st IEEE Symposium on Computational Intelligence and Games. pp. 285–291.
Stone, P., Kaminka, G., Kraus, S., Rosenschein, J., 2010. Ad hoc autonomous agent teams: collaboration without pre-coordination. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence. pp. 1504–1509.
Stone, P., Riley, P., Veloso, M., 2000. Defining and using ideal teammate and opponent agent models. In: Proceedings of the 12th Conference on Innovative Applications of Artificial Intelligence. pp. 441–442.
Stone, P., Veloso, M., 1999. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence 110 (2), 241–273.
Stone, P., Veloso, M., 2000. Multiagent systems: A survey from a machine learning perspective. Autonomous Robots 8 (3), 345–383.
Sukthankar, G., Goldman, R., Geib, C., Pynadath, D., Bui, H., 2014. Plan, Activity, and Intent Recognition: Theory and Practice. Morgan Kaufmann.
Sukthankar, G., Sycara, K., 2007. Policy recognition for multi-player tactical scenarios. In: Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems. pp. 58–65.
Sukthankar, G., Sycara, K., 2008. Hypothesis pruning and ranking for large plan recognition problems. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence. pp. 998–1003.
Suryadi, D., Gmytrasiewicz, P., 1999. Learning models of other agents using influence diagrams. Proceedings of the 7th International Conference on User Modeling, 223–234.
Synnaeve, G., Bessiere, P., 2011. A Bayesian model for opening prediction in RTS games with application to Starcraft. In: IEEE Conference on Computational Intelligence and Games. pp. 281–288.
Takahashi, Y., Edazawa, K., Asada, M., 2002. Multi-module learning system for behavior acquisition in multi-agent environment. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 1. IEEE, pp. 927–931.
Tambe, M., 1995. Recursive agent and agent-group tracking in a real-time dynamic environment. In: Proceedings of the 1st International Conference on Multi-Agent Systems. pp. 368–375.
Tambe, M., 1996. Tracking dynamic team activity. In: Proceedings of the 13th National Conference on Artificial Intelligence. pp. 80–87.
Tambe, M., 1997. Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124.
Tambe, M., Rosenbloom, P., 1995. RESC: an approach for real-time, dynamic agent tracking. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. pp. 103–110.
Tian, X., Zhuo, H., Kambhampati, S., 2016. Discovering underlying plans based on distributed representations of actions. In: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems. pp. 1135–1143.

Tuyls, K., Weiss, G., 2012. Multiagent learning: Basics, challenges, and prospects. AI Magazine 33 (3), 41.
van den Herik, H., Donkers, H., Spronck, P., 2005. Opponent modelling and commercial games. In: Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games. pp. 15–25.
Van Der Hoek, W., Wooldridge, M., 2002. Tractable multiagent planning for epistemic goals. In: Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Aystems. ACM, pp. 1167–1174.
Veloso, M., 1994. Planning and Learning by Analogical Reasoning. LNAI 886. Springer-Verlag.
Vered, M., Kaminka, G., 2017. Heuristic online goal recognition in continuous domains. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 4447–4454.
Vickrey, D., Koller, D., 2002. Multi-agent algorithms for solving graphical games. In: Proceedings of the 18th National Conference on Artificial Intelligence. AAAI, pp. 345–351.
Vidal, J., Durfee, E., 1995. Recursive agent modeling using limited rationality. In: Proceedings of the 1st International Conference on Multi-Agent Systems. pp. 376–383.
Visser, U., Weland, H., 2002. Using online learning to analyze the opponent's behavior. In: RoboCup 2002: Robot Soccer World Cup VI. Springer, pp. 78–93.
Von Neumann, J., Morgenstern, O., 1944. Theory of Games and Economic Behavior. Princeton University Press.
Wang, Z., Boularias, A., Mülling, K., Peters, J., 2011. Balancing safety and exploitability in opponent modeling. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence. pp. 1515–1520.
Watkins, C., Dayan, P., 1992. Q-learning. Machine learning 8 (3), 279–292.
Wayllace, C., Hou, P., Yeoh, W., 2017. New metrics and algorithms for stochastic goal recognition design problems. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 4455–4462.
Weber, B., Mateas, M., 2009. A data mining approach to strategy prediction. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games. pp. 140–147.
Wilks, Y., Ballim, A., 1986. Multiple agents and the heuristic ascription of belief. In: Proceedings of the 10th International Joint Conference on Artificial Intelligence. pp. 118–124.
Wright, J., Leyton-Brown, K., 2010. Beyond equilibrium: Predicting human behavior in normal-form games. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence. pp. 901–907.
Yoshida, W., Dolan, R., Friston, K., 2008. Game theory of mind. PLoS Computational Biology 4 (12).
Yu, H., Shen, Z., Leung, C., Miao, C., Lesser, V., 2013. A survey of multi-agent trust management systems. IEEE Access 1, 35–50.
Zeng, Y., Doshi, P., 2012. Exploiting model equivalences for solving interactive dynamic influence diagrams. Journal of Artificial Intelligence Research 43, 211–255.
Zhuo, H., Li, L., 2011. Multi-agent plan recognition with partial team traces and plan libraries. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 484–489.
Zhuo, H., Yang, Q., Kambhampati, S., 2012. Action-model based multi-agent plan recognition. In: Advances in Neural Information Processing Systems. pp. 368–376.
Zukerman, I., Albrecht, D., 2001. Predictive statistical models for user modeling. User Modeling and User-Adapted Interaction 11 (1), 5–18.