

Reasoning about Rationality and Beliefs

Ya'akov Gal and Avi Pfeffer
Division of Engineering and Applied Sciences
Harvard University, Cambridge, MA 02138
{gal,avi}@eecs.harvard.edu

ABSTRACT

In order to succeed, agents playing games must reason about the mechanics of the game, the strategies of other agents, other agents' reasoning about their strategies, and the rationality of agents. This paper presents a compact, natural and highly expressive language for reasoning about the beliefs and rationality of agents' decision-making processes in games. It extends a previous version of the language in a number of important ways. Agents can reason directly about the rationality of other agents; agents' beliefs are allowed to conflict with one another, including situations in which these beliefs form a cyclic structure; agents' play can deviate from the normative game theoretic solution. The paper formalizes the equilibria that holds with respect to agents' models and behavior, and provides algorithms for computing it. It also shows that the language is strictly more expressive than that of Bayesian games.

Categories and Subject Descriptors: I.2 [Computing Methodologies]: Artificial Intelligence

General Terms: Languages

Keywords: Decision-making under uncertainty, Game theory, Opponent modeling

1. INTRODUCTION

Agents playing games reason about a variety of issues, such as the mechanics of the game, other agents' strategies, and other agents reasoning about their own strategies. They may reason about the rationality of other agents, or they might behave irrationally themselves. In addition, agents may be uncertain about the mechanics of the game, or about other agents' beliefs about the mechanics.

In game theory, agents' uncertainty about the preferences and beliefs of other agents has traditionally been captured in the Bayesian game formalism [8]. However, for reasons that will be discussed, Bayesian Games are inadequate as a knowledge representation language for describing agents' reasoning in games. In this paper, we present a knowledge

representation language that provides for a natural, clear and compact model of agents who reason about the rationality of agents in games.

Our language is called *Networks of Influence Diagrams (NID)*. In previous papers, we discuss the expressive power of NIDs [5] and showed how they can be learned from data [6]. This paper extends NIDs in a number of important ways. First, we can model both rational agents, who play best-response, and irrational agents, whose play deviates from the best-response strategy, and allow for other agents to reason about such agents. We can also capture an agent who is *partially* irrational, in the sense that she is mixing between best-response and non best-response strategies; in particular, we can express agent i 's uncertainty whether agent j is about to deviate from her rational strategy, and allow agent i to respond to her model.

Third, we allow agents' beliefs about each other to conflict. This allows us to describe situations in which two agents disagree about the behavior of a third agent. More generally, it allows us to model situations in which agents do not have a common prior probability distribution over the state of the world. The NID language and its extensions will be presented in Section 2.

While these extensions greatly enhance the expressive power of the language, they also raise interesting challenges. When developing a language for describing beliefs and behavior in games, we need a solution concept that specifies what agents *should* do and what we might *expect* them to do. In traditional game theory, the normative (what agents should do), and descriptive (what we expect them to do) aspects have coincided. However, since our language allows agents to behave irrationally, we must be able to distinguish between normative and descriptive behavior in our solution concept. In Section 3, we provide equilibrium conditions that do just that. We introduce *best-response* strategies, in which agents are rational with respect to their beliefs, and *actually-played* strategies, which predict how agents *actually* behave, and show how the two are linked together by a set of equations.

Once we have introduced equilibrium conditions, we need a method for computing equilibria for a model in our language. Here we must show how to integrate different agents' beliefs into a single computational framework, despite the possibility that the agents' beliefs may conflict. We also need a way to solve games with cyclic belief structures. We present solution algorithms in Section 4.

One of the reasons that extending the language to allow cyclic belief structures is so important is that it allows NIDs to encompass that of Bayesian games. Previously, we argued

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'04, July 19–23, 2004, New York NY, USA.

Copyright 2004 ACM 1-58113-683-8/03/0007 ...\$5.00.

that NIDs were more natural and compact than Bayesian games, but that they only formed a subclass of Bayesian games, leading to a trade-off. Now there is no longer a trade-off. We show in Section 5 that NIDs can represent all Bayesian games.

1.1 Bayesian Games

In game theory, agents’ uncertainty about the preferences and beliefs of other agents has traditionally been captured in the Bayesian game formalism [8]. In this formalism, each agent has a set of possible types, and utility functions that depend on the type. Uncertainty is represented by associating for each type a probability distribution over the other agents’ types. It has been shown [11, 2] that Bayesian games are quite powerful and general.

However, Bayesian Games are inadequate as a knowledge representation language for describing agents’ reasoning in games. A knowledge representation language should not only be expressive, but also natural and intuitive, and it should provide compact representations. Bayesian games are deficient on both counts. They are representationally obscure, because all of the uncertainty must be folded into the utility functions and the distribution over the opponents. For example, consider a game in which the payoffs depend on the presence of oil in a location, and a seismic test has been conducted. The agents may have uncertainty about the effectiveness of the test, and therefore on the probability of oil. In order to represent this situation as a Bayesian game, this uncertainty must be folded into the utility functions via expected utility calculations. Thus, the Bayesian game obscures the real story behind the game.

Furthermore, Bayesian games can be unnecessarily large and unwieldy representations [5]. In addition, Bayesian games require all agents to be fully rational, in the sense that, in equilibrium, every type for every player plays a best response to its beliefs. However, actual agents playing games often play irrationally. Alternatively, they may model other agents as being irrational, while being rational themselves, in the sense that they play a best response to their beliefs. A case in point is the well-known game of RoShamBo (rock-paper-scissors). While the game theoretic equilibrium is to randomize uniformly, it always results in an expected payoff of zero. Instead, good players try to model the tendencies of their opponents and react accordingly.

2. NID SYNTAX

Our goal therefore, is to develop a highly expressive and natural knowledge representation language for describing agents’ beliefs and reasoning processes in games. The language should allow the modeling of rational and boundedly rational agents, and provide a seamless integration between them. It should generalize Bayesian games in expressive power, while providing a natural and compact alternative.

Networks of Influence Diagrams was designed with these goals in mind. We begin by reviewing the original version of the language, and then describe the three extensions. The building blocks of NIDs are Bayesian Networks [9], Influence Diagrams [14] and Multi-agent Influence Diagrams (MAID) [10]. A Bayesian network is a directed acyclic graph in which each node represents a random variable. An edge between two nodes X and Y implies that X has a direct influence on the value of Y . Each node X_i contains a conditional probability distribution (CPD) $P(X_i|Parents(X_i))$.

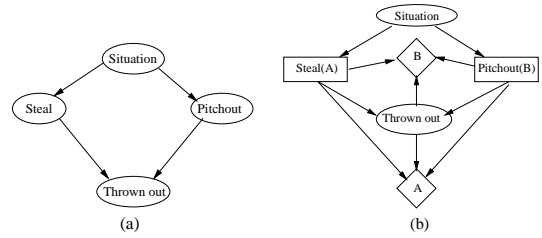


Figure 1: (a) Bayesian network; (b) MAID for baseball example

The topology of the network describes conditional independence relationships that hold in the domain. A Bayesian network defines a complete joint probability distribution over its random variables: $P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i|Parents(X_i))$.

As a running example to illustrate our language, we will use a simple scenario from the game of baseball. Figure 1(a) shows a Bayesian network for this scenario. Consider two baseball team managers A and B whose teams are playing the late innings of a closely tied game. Manager A , whose team is hitting, can attempt to advance a runner by instructing him to steal a base while the next pitch is being delivered. Stealing a base may be successful, resulting in benefit to the hitting team, or it may result in the runner being thrown out, resulting in a large cost to the hitting team. Manager B , whose team is pitching, can instruct his pitcher to throw a pitchout, thereby increasing the probability that a stealing runner will be thrown out. However, throwing a pitchout has a cost to the pitching team. The decision whether to steal and pitchout are taken simultaneously by both team managers. The Bayesian network describes a situation in which the two managers use specific strategies in making their decisions, encoded in the CPD for `STEAL` and `PITCHOUT`. These strategies depend on the current game situation, and affect the result of whether or not the runner is thrown out.

While the Bayesian network can be used to specify that agents play specific strategies, it does not capture the fact that agents are free to choose their own strategies, and it cannot be used to compute the optimal strategies for agents. Influence diagrams address these issues by extending Bayesian networks to decision problems and MAIDs extend influence diagrams to the multi-agent case.

A MAID adds decision nodes and utility nodes associated with particular agents to the chance nodes of Bayesian networks. If a decision node is associated with an agent, that agent gets to choose the value of the node. Edges leading into decision variables represent information that is available to the agent at the time the decision is made; parents of a decision node are called *informational parents*. Utility nodes may have chance nodes and decision nodes as parents. Associated with a utility node is a function mapping values of its parents to utilities. The utility of an agent is the sum of the values of the utility nodes associated with the agent. Given particular choices for all the decisions, a MAID specifies a probability distribution over all the chance nodes, and an expected value for each of the utility nodes. Thus a MAID encodes a game, and solving a MAID means computing Nash-equilibrium strategies for all agents.

Figure 1(b) shows the MAID for our baseball example. First, note that the `STEAL` and `PITCHOUT` nodes, which were

chance variables in the Bayesian network, have been turned into decision nodes, shown as rectangles. Each decision is associated with a manager. We named the manager of the hitting team as A , and the manager of the pitching team as B . Notice that both of the decisions have the SITUATION node as a parent. This is an informational parent, indicating that both managers know the game situation at the time they make their decision. Finally, we have added diamond-shaped utility nodes for each player, which depend on the two decisions, and on the chance variable representing whether or not the runner was thrown out. This MAID is a description of a game, and we can solve it to find a Nash equilibrium. In the Nash equilibrium for this game, both managers will randomize between their two strategies.

While MAIDs capture the structure of a game, they present a monolithic view of the game, describing only the sets of actions available to the agents and the different possible outcomes. They are not capable of describing different beliefs that the agents might have about the game or about each other. They also cannot capture situations in which agents play according to heuristics, or in which agents reason about the strategies other agents might be taking. NIDs build on top of MAIDs to directly address these issues.

To illustrate, let us now modify our baseball example. Suppose there are experts who will dictate whether or not a team should steal or pitchout. There is social pressure on the managers to follow the advice of the experts, because if their decision turns out wrong they can blame it on the experts. Now, suppose the game situation is such that the experts suggest that manager A should call a steal, and manager B should call a pitchout. This advice is common knowledge between the managers. Manager B may be uncertain as to whether A will in fact follow the experts and steal, or whether she will play rationally, meaning that she plays a best-response with respect to her beliefs. To quantify, B believes that with probability 0.7, A will follow the experts, while with probability 0.3, A will play rationally. A 's beliefs about B are symmetric.

It was these types of situations that inspired NIDs. A NID is a network of MAIDs, where each MAID intuitively represents a possible model of the game being played. Agents may use different possible models for making a decision, and they may have uncertainty about which model is used by others. The models may differ in a variety of ways. They may differ simply in the utility functions. They may also differ in aspects of the probability distribution governing possible outcomes. An agent might be replaced by an automaton that plays according to some pre-specified probability distribution. By allowing agents' beliefs and models to differ explicitly, the language provides for a natural encoding of these interesting situations.

More formally, a NID consists of a set of agents \mathbf{G} and a set of MAIDs \mathbf{I} , representing the different mental models the agents might use. Each MAID has a label and is referred to as a *block*. To capture agents' uncertainty about which block is being used by other agents, we introduce a new set of random variables: for each decision D belonging to each agent β in block K , we add a node $\text{Mod}[D]$ to K , whose values range over the set of blocks \mathbf{I} . Intuitively, when $\text{Mod}[D]$ takes value L it means that all agents in K see β as using block L to make her decision. In this case, we say that D is *modeled* by L . When $\text{Mod}[D]$ takes on the value K itself, it means that β is using the *correct* (from the

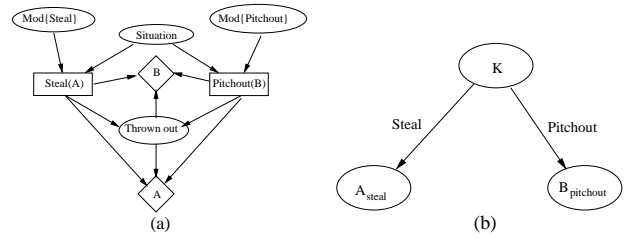


Figure 2: (a) Block K of NID; (b) NID graph for baseball example

point of view of block K) model to make her decision, and we say that decision D is *taken* by β at K .

The node $\text{Mod}[D]$ is a chance node just like any other. Therefore it may influence, or be influenced by, other nodes of K . Although MOD nodes are parents of decisions, they do not represent an agent's observations, and therefore are not considered informational parents.

We can model the modified baseball situation in a NID as follows: There are three blocks. The first block K corresponds to the real-world game. There is also a block A_{steal} , which corresponds to a situation where A follows an expert and instruct her players to steal. In this block, the STEAL decision is replaced with a chance node, which assigns probability 1 to stealing. There is a similar block $B_{pitchout}$ that assigns probability 1 to pitching out. Figure 2(a) shows block K . It is very similar to the MAID in Figure 1(b), but it now has MOD variables. The CPD of $\text{Mod}[Steal]$ assigns probability 0.7 to block A_{steal} and probability 0.3 to block K itself, capturing B 's uncertainty about how A makes her decision. The CPD of $\text{Mod}[Pitchout]$ is symmetric.

There are some technical conditions required for it to make sense to model a decision D at one block K using another block L . First of all, D must appear in L . However, it need not be a decision node in L . It may instead be a chance node. This corresponds to a situation where according to block L , the decision D is being taking by an automaton and not by a decision-making agent. A second requirement is that the informational parents of D in L be a subset of the informational parents of D in K . In other words, agent β cannot have more information when making decision D in K than she did in L .

NIDs are a graphical representation on two levels. In addition to the MAIDs, which are graphical descriptions of the individual mental models, NIDs as a whole naturally form a graph, hence the word “networks” in the name. The nodes of the graph are the blocks in \mathbf{I} . For any decision in K that is modeled at block L , there is an edge (K, L) in the network, labeled with the decision. Leaf nodes in the network represent blocks in which none of the agents are modeling any other agent. We show the NID graph for the baseball example in Figure 2(b).

2.1 Conflicting Beliefs

In the language we have described so far, if a decision D in K is modeled at block L , then all agents using mental model K believe that block L is used to make the decision D . All agents at a block must agree about how other agents make their decisions. While many interesting situations can be captured under this assumption, it is quite limiting. One natural situation that cannot be captured is one in which

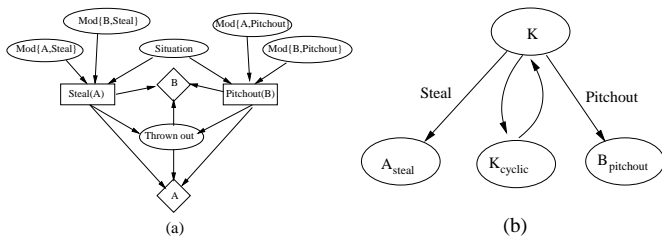


Figure 3: (a) Block K of NID with conflicting beliefs; (b) cyclic NID

there are three agents, and two of the agents have different beliefs about the decision making process of the third.

More generally, the previous language cannot capture situations in which the agents do not have a common prior distribution over the state of the world. In traditional game theory, agents’ beliefs are assumed to be consistent with a common prior distribution, meaning that the beliefs of agents are the posterior distributions resulting from conditioning a common prior on each agent’s information. As a consequence of this assumption, agents’ beliefs can differ only if they have different information [1]. This result gave rise to theoretic work that attempted to relax the common prior assumption [12]. Once we have a language that allows us to talk about different mental models that agents have about the world, and different beliefs that they have about each other and about the structure of the game, it is natural to relax the common prior assumption.

For example, consider another extension to the baseball example, where there is an additional variable *SPEED* corresponding to the speed of the runner. This variable influences the probability that the runner will be thrown out if a steal is attempted. It is natural in NIDs to describe a situation where there are several models, each with a different probability distribution over *SPEED*. Suppose that in the real-world, the runner is actually slow. On the other hand, manager A is positive that the runner is fast, and therefore, she believes that manager B believes so as well.

Thus, in block K , manager A uses a model L to make her decision, in which the CPD of *SPEED* assigns probability 1 to *fast*, and she believes manager B does as well. On the other hand, if B believes that the runner is fast only with probability 0.7, than at block K , manager B uses a model M to make her decision, in which the CPD of *SPEED* assigns probability 0.7 to *fast*. Furthermore, if B thinks that A also believes that, then B believes both managers model their decisions by M . Here we have a case of an uncommon prior, because the two managers have different prior distributions over the *SPEED* variable.

In order to model this kind of situation, we need to distinguish between the different beliefs of different agents about how decisions are taken. For each decision D of agent β in block K , and each agent α , we now introduce a *separate* MOD variable $\text{Mod}[\alpha, D]$. This corresponds to α ’s belief about which mental model β uses to make the decision D . The MOD variables for different agents can have different probability distributions, indicating that agents’ beliefs are different. Note that agent α will also have MOD variables pertaining to her own decisions. Figure 3 illustrates a block with the new MOD variables. Following our example, $\text{Mod}[A, \text{Steal}]$ and $\text{Mod}[A, \text{Pitchout}]$ will both take

on the value L with probability 1, while $\text{Mod}[B, \text{Steal}]$ and $\text{Mod}[B, \text{Pitchout}]$ take the value M with probability 1.

2.2 Agents who Play Irrationally

In our original paper, agents were always rational with respect to their models, meaning that their strategies were always a best response to their (possibly incorrect) models. In particular, this meant that there was no distinction between the normative behavior prescribed for each agent in each block, and the descriptive prediction of how the agent actually *would* play when using that block.

However, in the real-world, we need to model agents who are not completely rational, whose actual behavior, in fact, differs from their best response. Now that we have introduced separate MOD variables for each agent, there is a natural way to express this kind of situation. If D_α is a decision associated with agent α , we can use $\text{Mod}[\alpha, D_\alpha]$ to describe which block α actually uses to make the decision D_α . In block K , if $\text{Mod}[\alpha, D_\alpha]$ is equal to K with probability 1, then it means that within K , α actually is making the decision according to her beliefs in block K , meaning that α will play a best response to the strategies of other agents, given her beliefs. If, however, $\text{Mod}[\alpha, D_\alpha]$ assigns positive probability to some block L other than K , it means that there is some probability that α will not play a best response to his beliefs in K , but rather play according to some other block L . In this case, we say α *self-models* at block K .

We will demonstrate by modifying our example. Let’s suppose that there are now two successive pitches, and on each pitch the managers have an option to steal or pitchout. Clearly each manager is faced with a sequential decision problem, and the strategy for the second pitch is relevant to the strategy for the first pitch. Now, each of the managers could use backward induction to compute optimal strategies for the second pitch, by working backwards to the first pitch. However, this is only a valid procedure if the managers actually will behave rationally on the second pitch. Suppose that manager B is under strong pressure to pitchout on the second pitch. Manager B wishes to take this possibility into account, while making her decision for the first pitch. Therefore, $\text{Mod}[B, \text{Pitch}_2]$ will assign positive probability to B_{pitchout} . However some small probability will be assigned to the real-world block, meaning that manager B thinks she might still behave rationally on the second pitch. Overall, agent B ’s actual-play is the expectation of her strategy at each block. In other words, agent B is self-modeling her decision at the real-world block, to account for the possibility in which her actual play differs from her best response.

2.3 Cyclic Belief Structures

Our final modification to the language is to allow NIDs to describe cyclic belief structures. Cyclic belief structures are important in game theory, where they are used to model agents who are symmetrically modeling each other. They are used to describe an infinite regress of “I think that you think that I think...” reasoning. Furthermore, cyclic belief structures can be expressed in Bayesian games, so it is vital to allow them in NIDs in order for NIDs to encompass Bayesian games.

Allowing cyclic belief structures requires no new syntax. The only difference is that in the original language the NID graph was restricted to be a directed acyclic graph. We now allow it to be cyclic. Cyclic belief structures do require

major changes in defining and computing the solution to a NID, as we will discuss in the next two sections.

To illustrate, suppose manager B believes that with probability 0.2 manager A thinks the probability of throwing out a stealer is very low. However, manager B also believes that manager A can reason about manager B modeling her. We can model this scenario by including a block K_{cyclic} , whose structure is similar to K , except that `THROWN-OUT` is *true* with a very low probability when the runner tries to steal. $\text{Mod}[B, D_A]$ in block K assigns probability 0.2 to K_{cyclic} , and probability 0.8 to K . At block K_{cyclic} , $\text{Mod}[A, A_B]$ assigns probability 1 to K , and A takes her decision at K . There is thus a cycle between K and K_{cyclic} in the NID graph, as described in Figure 3(b).

3. EQUILIBRIUM CONDITIONS

In classical game theory, agents' strategies are in equilibrium when each agent is playing a best response to the other agents' strategies. Because NIDs allow modeling of bounded rationality, the solution concept for NIDS allows two kinds of strategy to appear in equilibrium: *best-response* strategies, which specify the optimal course of action for each agent according to her beliefs regarding the strategies of her opponents; and *actually-played* strategies, which are the model's prediction of how the agents actually play. We say that an agent in a block is *rational with respect to its beliefs* if its actually-played strategy equals its best-response strategy.

We proceed to develop equilibrium conditions for the two kinds of strategy. These conditions specify what it means for a strategy to be a best-response or an actually-played strategy.

Intuitively, A best-response strategy for a particular decision is a strategy that maximizes the utility achieved by the agent, given her beliefs about the actually-played strategies for other decisions. Meanwhile the actually-played strategy is the expectation, over the different blocks the agent might use to make her decision, of the strategy played according to each block. In particular, if an agent in block K takes her decision at K , then the best-response strategy for K is used. Thus best-response strategies are defined in terms of actually-played strategies, and vice versa. We proceed to develop these intuitions step by step. We will denote the best-response for decision D_i in block K by Θ_i^K , and the actually-played strategy by Φ_i^K .

To begin with, suppose an agent α making a decision D_i believes that a particular pure strategy is being used for the other decisions. We will denote such a strategy by s_{-i} . Now, for any pure strategy s_i for D_i , we let $\mathbf{s} = \langle s_i, s_{-i} \rangle$ denote the pure strategy profile in which s_i is used for D_i , while s_{-i} is used for the other decisions. If we specify a particular strategy profile \mathbf{s} for a block, the MAID defining the block can be converted into a Bayesian network, in which the decision nodes are replaced with chance nodes, each implementing the strategy defined by \mathbf{s} . The resulting Bayesian network defines an expected utility to agent α , which we denote by $U_\alpha^{\mathbf{s}}$.

Clearly, a best response for α is a strategy that maximizes this expected utility. First let us consider pure strategies. A pure strategy for D_i must specify what the agent should do, for any situation the agent might encounter while making the decision. In the MAID, the node D_i has informational parents. The possible instantiations of the informational parents are the different possible situations the agent might

encounter. Thus a pure strategy for D_i must specify a choice of action for each possible instantiation of the informational parents, i.e. it is a function from the set of parent instantiations to the domain of D_i . But α is not restricted to pure strategies; he may choose a mixed strategy. If we let S_i denote the set of pure strategies for D_i , then ΔS_i , the set of probability distributions over S_i , is the set of mixed strategies for D_i . Agent α is free to choose any τ in ΔS_i . We must then sum over possible pure strategies s_i to obtain the expected utility to α from playing τ . We can now take our first crack at the best-response equation.

$$\Theta_i^K \in \operatorname{argmax}_{\tau \in \Delta S_i} \sum_{s_i \in S_i} \tau(s_i) \cdot U_\alpha^{\mathbf{s}}$$

Note that the equation uses set membership rather than equality to indicate that there may be more than one τ that maximizes the right hand side. This equation is only valid if α believes the other agents to be playing a particular pure strategy s_{-i} . In general, the other agents may be playing a mixed strategy. In fact, agent α believes that the other agents are playing the mixed strategy defined by their actually-played strategies. Therefore, we need to sum over all possible pure strategies for the other agents, using the actually-played strategies to determine the probability of each pure strategy. For now, let us assume that in block K , agent α believes that all agents take their decisions in K . So the actually-played strategies to use for the other decisions are the actually-played strategies for block K . Let \mathbf{S} denote the set of all pure strategy profiles for all decisions in a block. Then

$$\Theta_i^K \in \operatorname{argmax}_{\tau \in \Delta S_i} \sum_{\mathbf{s} \in \mathbf{S}} \left(\prod_{j=1, j \neq i}^n \Theta_j^K(s_j) \right) \cdot \tau(s_i) \cdot U_\alpha^{\mathbf{s}}$$

This equation only holds when α believes all agents to be taking their decision at block K , but in fact α may model them as using other blocks for the decisions. To handle this situation, we introduce the concept of a block label profile. A block label profile for α represents α 's beliefs about which blocks are used for all decisions in K . It is an assignment of a block to each MOD variable $\text{Mod}[\alpha, D_j]$, $j = \{1, \dots, n\}$ in K . For a given block label profile, element m_j represents the block in which decision D_j is modeled. The actually-played and best-response strategies to use for decision D_j in block m_j is $\Phi_j^{m_j}$ and $\Theta_j^{m_j}$, respectively.

Now, α may be uncertain as to which blocks other agents used. Thus he has a probability distribution over the different possible m_{-i} , where m_{-i} denotes a block label profile for all decisions except D_i . Let $\text{Mod}[\alpha, D_{-i}] = m_{-i}$ denote the event that all of α 's MOD variables for decisions other than D_i are as specified in m_{-i} . As we described above, when a pure strategy profile \mathbf{s} is specified, it induces a Bayesian network. This network defines a probability distribution over block label profiles. We denote this distribution as $P^{\mathbf{s}}(\text{Mod}[\alpha, D_{-i}] = m_{-i})$. We can now take the expectation over the set of all possible block label profiles M_{-i} .

There is one more subtlety before we can present the final equation for best-response strategies. In each component of the summation, we are specifying a particular assignment of values to MOD variables. Because MOD variables function like any other nodes in the network, they can

be parents of other nodes. In particular, they can be parents of utility nodes. Thus the expected utility achieved by α under strategy profile \mathbf{s} now depends on the particular block label profile. To handle this, we introduce the notation $U_\alpha^{\mathbf{s}}(\text{Mod}[\alpha, D_{-i}] = m_{-i})$ to denote the expected utility, defined by the Bayesian network induced by \mathbf{s} , to agent α , conditioned on the event $\text{Mod}[\alpha, D_{-i}] = m_{-i}$. We now present our best-response equation:

$$\Theta_i^K \in \underset{\tau \in \Delta S_i}{\text{argmax}} \sum_{m_{-i} \in M_{-i}} \sum_{\mathbf{s} \in \mathbf{S}} P^{\mathbf{s}}(\text{Mod}[\alpha, D_{-i}] = m_{-i}) \cdot \left(\prod_{j=1, j \neq i}^n \Phi_j^{m_j}(s_j) \right) \cdot \tau(s_i) \cdot U_\alpha^{\mathbf{s}}(\text{Mod}[\alpha, D_{-i}] = m_{-i}) \quad (1)$$

For the actually played strategy for D_i , we look at $\text{Mod}[\alpha, D_i]$ where α is the agent that makes decision D_i . When $\text{Mod}[\alpha, D_i]$ is some block L different from K , it means that α uses model L to determine what to do for D_i , so we simply take the actually-played strategy for L . Otherwise, agent α uses K itself to determine what to do for decision D_i in K . In this case, we stipulate that α must use its best response strategy in K . To write down the equation, we use the notation P^{Φ^K} to denote the probability distribution induced by the actually-played strategy profile Φ^K .

$$\Phi_i^K = \Theta_i^K \cdot P^{\Phi^K}(\text{Mod}[\alpha, D_i] = K) + \sum_{L \in \mathcal{B} \setminus \{K\}} \Phi_i^L \cdot P^{\Phi^K}(\text{Mod}[\alpha, D_i] = L) \quad (2)$$

Note that for any decision D that is *taken* by its agent at K with probability 1, the actually-played and best-response strategies are equal.

4. COMPUTING EQUILIBRIA

Solving a NID corresponds to computing best-response and actually-played strategies for each decision of each agent at each block. Previously, we presented a solution algorithm for an early version of NIDs [5]. However, it was restricted to NIDs whose graphical representations is a DAG, it did not allow for self-modeling, and it assumed that beliefs of all agents were the same at each block.

How can we solve NIDs that *do* include these features? We can try to express every actually-played strategy in terms of best-response strategies through expanding equations (1) and (2). However, the resulting expression would consist of fixed point equations that are non-linear in the unknowns and do not have a closed form solution. Fortunately, we now show, that there do exist procedures for solving a large subclass of these NIDs.

We will begin by describing an algorithm for computing NIDs that include self-modeling and subjective beliefs, but do not contain cyclic beliefs, i.e. the NID structure is a DAG. We traverse the graph in a bottom-up fashion, from the leaves of the graph to the root. The leaves are simply MAIDs, and can be solved using the MAID algorithm [10]. In the case that they only involve decisions of a single agent, they can be solved using a standard influence diagram algorithm [14, 4]. After solving each node, we pass a message to the parent containing the actually-played strategies for each decision that is modeled by the node. For an internal block K , once all of its children have been solved, actually-played strategies exist for all of the decisions in K modeled by one of K 's descendants.

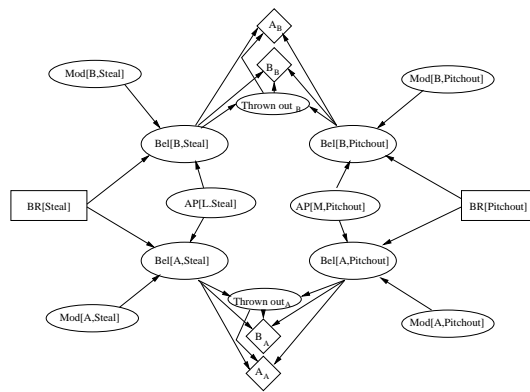


Figure 4: Constructed MAID for baseball example

We then transform K into a new MAID K' . The goal is for K' to include all the information necessary to represent the decision problems faced by all the agents. To achieve this, we introduce the following nodes into K' :

- For each decision D modeled at a child L of K , we introduce the node $\text{AP}[L, D]$. This is a chance node, whose CPD implements the previously computed *actually-played* strategy for D at L .
- For each decision D in K we introduce a $\text{BR}[D]$ node, representing the best response strategy for D . This is a decision node — solving the new MAID K' will determine the *best-response* strategies.
- For each decision D in K and each agent α , we introduce a node $\text{Bel}[\alpha, D]$, to represent the belief of agent α about what strategy is used for D in block K . This is the core of our construction. The beliefs of α about D are determined by the actually-played strategies for D at child blocks, and the best-response strategy for D at this block. The variable $\text{Mod}[\alpha, D]$ is used to determine how strongly to weight each of the actually-played strategies and the best-response strategy. Thus the parents of $\text{Bel}[\alpha, D]$ are $\text{BR}[D]$, $\text{Mod}[\alpha, D]$ and the $\text{AP}[L, D]$ for all child blocks L .
- Each agent will have a different belief about the strategy used for D , i.e. a different $\text{Bel}[\alpha, D]$. Whenever a node X in the original MAID K depended on the decision D , it will now depend on $\text{Bel}[\alpha, D]$. But since each agent has a different $\text{Bel}[\alpha, D]$, each agent will have different beliefs about X . Therefore the node X must be replicated, once for each agent, to describe what each agent believes about X . In particular, this applies to the utility nodes in the original MAID. Each agent will have its own opinion of how the strategies affect the utilities of all agents.

As a result of this construction, the strategy computed for $\text{BR}[D]$ will be the best-response strategy for D as defined in Equation 1. In addition, the probability distribution computed for $\text{Bel}[\alpha, D]$, where α is the agent that decides D , will be the actually-played strategy for D as defined in Equation 2. We demonstrate the algorithm on the block described in Figure 2(a). The result is shown in Figure 4. Notice the central role played by the Bel nodes. For example,

$\text{Bel}[B, \text{Steal}]$ has as parents $\text{AP}[L, \text{Steal}]$, $\text{BR}[\text{Steal}]$, and $\text{Mod}[B, \text{Steal}]$. The CPD of $\text{Bel}[B, \text{Steal}]$ is a multiplexer: it states that when $\text{Mod}[B, \text{Steal}] = K$, then $\text{Bel}[B, \text{Steal}] = \text{BR}[\text{Steal}]$, whereas when $\text{Mod}[B, \text{Steal}] = L$, $\text{Bel}[B, \text{Steal}] = \text{AP}[L, \text{Steal}]$. The node $\text{Bel}[B, \text{Steal}]$ also influences a number of other nodes, including the utilities to the two agents, according to B . The nodes influenced by $\text{Bel}[B, \text{Steal}]$ are duplicated, so that agent A also has a version of them. We can now solve the MAID in Figure 4 and pass decision rules for AP nodes up to the parent.

It turns out that a simple extension of this algorithm can solve a restricted class of cyclic NIDs. To define this class, we need the following definition: We say that agent α employs a *self loop* if there is a cycle of blocks $K_1, K_2, \dots, K_n, K_1$ such that at each block, α self-models by using the successor block to model his own decision. A cyclic NID that has no self loops can be solved using an extension of the algorithm described above. We can find the maximal strongly connected components (MSCC) of the NID in linear time [3].

We traverse the MSCC in a manner consistent with our ordering. If a component C_i consists of a single block, it is not part of any cycle and can be handled just like before. If C_i consists of a set of blocks, we create a MAID M which is the union of all of the MAIDs in C_i , that is, we copy the nodes and edges of each MAID to C with the same CPD. We connect the MAIDs by adding BR, AP and Bel nodes in M for every modeled decision in every block in the component, just like we discussed above. Since no agent employs a self-loop, we are guaranteed that the resulting MAID M will be acyclic. We can then solve the MAID to obtain best-response and actually-played strategies, as we did before.

Lastly, we present a procedure for solving general NIDs. This technique does not make use of the graphical structure of the model. The procedure is a version of a “guess-the-support” strategy [13]. The support of an actually-played strategy profile Φ at block K is the set of all pure-strategy profiles that have positive probability under Φ . The idea of a guess-the-support strategy is that in equilibrium, if two different pure strategies of agent α both have positive probability, then they must give her the same expected payoff.

We will formulate equations for a subclass of NIDs called regular NIDs. We say that a NID is *regular* if every agent $\alpha \in \mathbf{G}$ has a single decision D_α at each block, the graph structure of the MAID at each block is the same, and no decision variable is a parent of any MOD variable. Now, let \mathbf{C}_i be some subset of pure strategies representing our current guess as to which actually-played strategies have positive probability in equilibrium for decision i . If there is an actually-played equilibrium Φ with support $\times_{i \in \mathbf{G}} \mathbf{C}_i$, then there must exist numbers (x_1, \dots, x_n) such that the following equations are satisfied.

$$\begin{aligned} \sum_{m_{-i} \in M^{K_i}} P(\text{Mod}[\alpha, D_{-i}] = m_{-i}) \cdot & \quad \forall i \in \mathbf{N}, \\ \sum_{s_{-i} \in \mathbf{S}_{-i}} \prod_{j=1, j \neq i}^n \Phi_j^{m_j}(s_j) \cdot & \quad \forall c_i \in \mathbf{C}_i \\ \mathcal{U}_\alpha^{(s_{-i}, c_i)}(\text{Mod}[\alpha, D_{-i}] = m_{-i}) = x_i & \\ \Phi_i^K(a) = 0 \quad \forall a \notin \mathbf{C}_i, \forall i \in \mathbf{G} & \\ \sum_{c_i \in \mathbf{C}_i} \Phi_i^K(c_i) = 1 & \end{aligned}$$

The first equation states that the expected utility for agent i is the same for every pure strategy in his set of support. The second equation specifies that any strategy

that is not in the actually-played support set is played with zero probability. The third equation verifies that the sum of probabilities with which every strategy is played is 1. Together, these equations give us $|\sum_{i \in \mathbf{G}} \mathbf{C}_i|$ equations with the same number of unknowns, namely (x_1, \dots, x_n) and $\Phi_i^K(c_i)$. Therefore we might be able to find solutions for them. In particular, if there are only 2 agents in the NID, then the set of equations are linear in the unknowns and we can solve them easily. After extracting the actually-played strategies for the NID, the best-response strategies are computed by solving equation (2) for each decision.

5. NIDS AND BAYESIAN GAMES

In game theory, uncertainty is traditionally handled by the Bayesian game framework [8], in which each agent has a discrete type embodying its private information. Let \mathbf{N} be a set of agents. A *Bayesian game* includes, for each player i a set of possible types \mathbf{T}_i , a set of possible actions \mathbf{C}_i , a conditional distribution p_i and a utility function u_i . Let $\mathbf{T} = \times_{i \in \mathbf{N}} \mathbf{T}_i$ and let $\mathbf{C} = \times_{i \in \mathbf{N}} \mathbf{C}_i$. For each player i , let $\mathbf{T}_{-i} = \times_{j \neq i} \mathbf{T}_j$ denote the set of all possible types other than those of player i . The probability distribution p_i is a function from t_i to $\Delta \mathbf{T}_{-i}$, i.e. $p_i(\cdot | t_i)$ specifies for each type $t_i \in \mathbf{T}_i$ a joint distribution over the types of the other agents. The utility function u_i is a function from $\mathbf{C} \times \mathbf{T}$ to the real numbers. Let $\sigma_i(\cdot | t_i)$ denote the randomized strategy of player i given that her type is t_i . In a *Bayesian Nash equilibrium*, each player chooses, for each type, a randomized strategy that is the best response to the strategies of the other agents, given its beliefs about their type:

$$\sigma_i(\cdot | t_i) \in \text{argmax}_{\tau \in \Delta \mathbf{C}_i} \sum_{t_{-i} \in \mathbf{T}_{-i}} p_i(t_{-i} | t_i) \cdot \sum_{c \in \mathbf{C}} \left(\prod_{j \in \mathbf{N}_{-i}} \sigma_j(c_j | t_j) \right) \tau(c_i) u_i(\mathbf{c}, \mathbf{c})$$

Unlike NIDs, the semantics of Bayesian games does not differentiate between best response and actually played strategies. In fact, the prediction is that the agents will actually play best response strategies. However, we can ask whether the language of NIDs encompasses that of Bayesian games. Given a Bayesian game, can we construct a NID that has essentially the same equilibria, in the sense that both the best-response and actually-played strategies in the NID are the same as the best-response strategies for the Bayesian game? We formalize this question as follows. Let B be a Bayesian game and N a NID. We say that N *represents* B if there exists an injective mapping f from types in B to (block,agent) pairs in N such that the following hold:

1. For any Bayesian Nash equilibrium σ of B , there exists a NID equilibrium of N , such that for every type t_i , if f maps t_i to (K, α) , the best-response and actually-played strategies for α in K are equal to $\sigma_i(\cdot | t_i)$
2. For any NID equilibrium of N , there exists a Bayesian Nash equilibrium σ of B such that for every (K, α) in the image of f , $\sigma_i(\cdot | t_i)$ where $t_i = f^{-1}(K, \alpha)$ is equal to the best-response and actually-played strategies for α in K .

THEOREM 1. *Every Bayesian game can be represented by a NID.*

The proof, which is constructive, calls for constructing a NID in which the set of agents equals the set of agents in the

Bayesian game. Each type t_i of agent i in the Bayesian game has a corresponding block in the NID, labeled by t_i . The MAID of block t_i is designed so that agent i 's distribution over MOD variables corresponds to $p_i(\cdot|t_i)$. In particular, $\text{Mod}[i, C_i]$ takes value t_i with probability 1, so the decision C_i is taken at the block, and actually-played strategies are equal to best-response strategies. The full proof is omitted due to space requirements.

The natural question to ask next is whether NIDs can be captured by Bayesian games. Clearly they cannot be captured in general, since NIDs allow actually-played strategies to differ from best-response strategies, while Bayesian games require them to be equal. However, if we require all decisions to be *taken* at their blocks (disallowing interesting forms of irrationality), can all NIDs now be captured by Bayesian games? To formalize this question we formulate a definition of *representation* of a NID by a Bayesian game that is analogous to representation of a Bayesian game by a NID.

LEMMA 1. *Every NID N can be converted to a regular NID N' such that the equilibrium conditions at N are equal to the equilibrium conditions at N' .*

The proof consists of constructing N' such that the number of agents in N' is equal to the number of decisions in N . In some blocks, there will be “vacuous” decisions that do not affect any of the utilities for any agents. Most of the details are straightforward. However, one interesting point is that N might allow a decision in one block to become a chance variable in another block, thereby turning the agent into an automaton. In order to make this NID regular, we need to convert the automaton chance variable back to a decision in N' , and create a utility variable such that the strategy defined by the equilibrium conditions for this agent follows the automaton. Fortunately, this can always be done. It is obvious that if the automaton plays a pure strategy, one can create a utility function that rewards that strategy enormously, thus making it the dominant strategy for that agent. In fact, one can create an automaton to model any randomized strategy. This can be proved by induction, using a matching-penny like game that has a unique equilibrium equal to the desired randomized strategy.

THEOREM 2. *Any NID, in which agents do not self-model, can be represented by a Bayesian game.*

The proof, which is again by construction, demonstrates how a regular NID in which agents do not self-model can be converted to a Bayesian game, and then uses Lemma 1.

6. CONCLUSION

We have presented a highly expressive language for describing agents' beliefs and decision making processes in games. We showed how to describe agents who behave irrationally, in the sense that their actual play does not correspond to the best possible response given their beliefs about the world and about other agents. We showed how to express situations in which agents have conflicting beliefs, including situations where the agents do not have a common prior distribution over the state of the world. Finally, we showed how to capture cyclic reasoning patterns, in which agents engage in infinite chains of “I think that you think that I think...” reasoning.

In this work, we have developed a solution concept that ties together the possible irrationality of agents with the best response strategies of rational agents. We have developed a series of solution algorithms for our language. We have analyzed the relationship between our language and Bayesian games, showing that Bayesian games are a strict subset of our language, but if one disallows irrational behavior the expressive power of the two formalisms is equal. This result demonstrates that the advantages of our language, namely the compact representation and readability, can be obtained without sacrificing expressive power. We are currently working on formalizing the connection between our language and epistemic logic formalisms.

A vital question that we have only begun to explore is the use of our language to learn about agents' behavior and reasoning processes. In an initial study [6], we showed that our language could be used to learn non-stationary strategies in rock-paper-scissors. Recently, we showed how models that were inspired by NIDs can learn people's play in negotiation games [7]. The focus of our continuing work will be to develop a general method for learning models in NIDs.

Acknowledgments

This work was supported by an NSF Career Award IIS-0091815.

7. REFERENCES

- [1] R. Aumann and A. Brandenburger. Epistemic conditions for nash equilibrium. *Econometrica*, 63, 1995.
- [2] A. Brandenburger and E. Dekel. Hierarchies of beliefs and common knowledge. *Journal of Economic Theory*, 59, 1993.
- [3] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.
- [4] F. F. Jensen and S. L. Dittmer. From influence diagrams to junction trees. In *Proc. 10th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1994.
- [5] Y. Gal and A. Pfeffer. A language for modeling agents' decision making processes in games. In *Proc. 2nd International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 2003.
- [6] Y. Gal and A. Pfeffer. A language for opponent modeling in repeated games. In *AAMAS Game Theory workshop*, 2003.
- [7] Y. Gal, A. Pfeffer, F. Marzo, and B. Grosz. Learning social preferences in games. In *Proc. 19th National Conference on Artificial Intelligence (AAAI)*, 2004.
- [8] J.C. Harsanyi. Games with incomplete information played by 'Bayesian' players. *Management Science*, 14, 1967-68.
- [9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.
- [10] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.
- [11] J.-F. Martens and S. Zamir. Formulation of Bayesian analysis for games with incomplete information. *International Journal of Game Theory*, 14, 1985.
- [12] S. Morris. The common prior assumption in economic theory. *Econ. Philos.*, (53):227-253, 1995.
- [13] R. Myerson. *Game Theory*. Harvard University Press, 1991.
- [14] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871-882, 1986.