

A Language for Modeling Agents' Decision Making Processes in Games

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

ABSTRACT

Multi-agent systems that use game-theoretic analysis for decision making traditionally take a normative approach, in which agents' decisions are derived rationally from the game description. This approach is insufficient to capture the decision making processes of real-life agents. Such agents may be partially irrational, they may use models other than the real world to make decisions, and they may be uncertain about their opponents' decision making processes. We present Networks of Influence Diagrams (NIDs), a language for descriptive decision and game theory that is based on graphical models. NIDs provide a framework for computing optimal decisions for agents that operate in an environment characterized by uncertainty, not only over states of knowledge but also over game mechanics and others' decision processes. NIDs allow the modeling of situations in which an agent has an incorrect model of the way the world works, or in which a modeler has uncertainty about the agent's model. One can also recursively model agents' uncertain beliefs about other agents' decision-making models. We present an algorithm that computes the actions of agents under the assumption that they are rational with respect to their own model, but not necessarily with respect to the real world. Applications of our language include determining the cost to an agent of using an incorrect model, opponent modeling in games, and modeling bounded rationality.

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

General Terms: Languages

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

1. INTRODUCTION

In recent years, decision theory and game theory have had a profound impact on artificial intelligence. On a fundamental level, the decision-theoretic approach provides a definition of what it means to build an intelligent agent, by

equating intelligence with utility maximization. Meanwhile, game theory has been adopted by many as the basis for building multi-agent systems. More concretely, a wide variety of representations and algorithms have been developed to determine decision-theoretic and game-theoretic solutions to problems. There is therefore a need for good knowledge representation languages for representing games and the agents that play them.

The focus in AI so far has been on the classical, normative approach to decision and game theory. In the classical approach, a game specifies the actions available to the players, and the utility to each player associated with each possible set of actions. The game is then analyzed to determine rational strategies for each of the players. Fundamental to this approach are the assumptions that the structure of the game, including the payoff functions and strategies available to each player is known to all of the players, and that all of the players' reasoning about the game is captured in the game structure itself. These assumptions do not hold generally in the real world. A player may be mistaken about the structure of the game, or may have uncertainty about the beliefs of other players about the structure of the game. Also, agents typically use various data structures and reasoning methods not present in the game itself, such as heuristics, to determine how to play. A clear distinction must be made between the structure of the game, which determines how the actions of the agents produce effects in the real world, and the mental models used by the agents to make their decisions.

This paper presents a knowledge representation language for games that makes this distinction. In our framework, called *Network of Influence Diagrams (NIDs)*, there is an explicit model of the real-world game being played, as well as additional mental models of agents playing the game. The language allows for the possibility that an agent's model is different from the real-world model. The language also allows multiple possible mental models for an agent, with uncertainty over which model the agent actually uses. The language is recursive, so that the mental model for an agent may itself contain models of the mental models of other agents, with associated uncertainty.

NIDs can be used both descriptively and normatively. In the descriptive approach, they can be used as a tool for describing, in a clear and explicit manner, the reasoning of agents. NID models can be solved to determine the behavior that results from the agents' reasoning processes. The models can thereby be used to predict agents' behavior, or to explain how certain behavior might have arisen. When used

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'03, July 14–18, 2003, Melbourne, Australia.
Copyright 2003 ACM 1-58113-683-8/03/0007 ...\$5.00.

to predict the behavior of other agents, NIDs can form the basis for an opponent modeling approach to game playing. In the normative approach, NIDs are used to model a game, and the beliefs of the agents involved in the game. Solving the NID amounts to computing behavior that is rational *with respect to the beliefs of the agents*. In other words, it is the best that the agents can do, given their beliefs. We present a variety of examples in this paper demonstrating the expressive power of NIDs.

The NID framework is based on *influence diagrams (IDs)* [10], a representation language based on graphical models for single-agent decision problems, and their recent extension to *multi-agent influence diagrams (MAIDs)* [14]. Graphical models, such as Bayesian networks, provide for natural representations of complex situations by explicitly describing the variables involved and the relationships between them. The resulting representations are much more compact than they would be otherwise, and often lead to exponential savings in inference time. MAIDs extended the benefits of graphical models to game theory, but only for the classical approach in which only the real-world model is described. Our work provides the benefits of graphical models to modeling agents’ decision making processes.

NIDs share a close relationship with the classic game theoretic formalism of *Bayesian games* [11]. Any Bayesian game can be reduced to a NID, and computing the Nash equilibrium [12] of the Bayesian game reduces to the same strategies that are obtained by solving the NID. In a separate paper [7], we show that in fact, NIDs are a superset of Bayesian games. In this paper, we show that NIDs are better suited to knowledge representation than Bayesian games, providing representations that are more compact, descriptive and explicit, thereby making them preferable for AI systems.

2. NID SYNTAX

We begin the description of NIDs by describing the basic building blocks, *influence diagrams (IDs)* [10]. An ID consists of a directed graph with three types of nodes: chance nodes, drawn as circles, represent random variables; decision nodes, drawn as rectangles, represent decision points; and value nodes, drawn as a diamonds, represent the agent’s utility which is to be maximized. There are two kinds of edges in the graph. Edges leading to chance and value nodes represent probabilistic dependence, in the same manner as edges in a Bayesian network. Edges leading into decision variables represent information that is available to the agent at the time the decision is made.

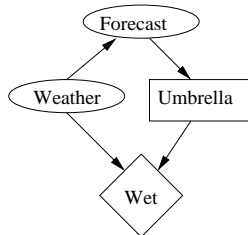


Figure 1: real-world ID of umbrella scenario

For example, consider the following scenario: Waldo is about to leave his house, and needs to decide whether or

not to take an umbrella. Waldo’s objective is to remain dry, but carrying an umbrella around is annoying. The ID in Figure 1 describes Waldo’s decision problem as we see it. There is a prior distribution over the weather, for which the forecast is a noisy sensor. Waldo gets to observe the forecast, and decides whether or not to take an umbrella, which, depending on the weather, influences his final utility.

Solving an ID requires computing an optimal strategy for the agent. Such a strategy specifies what the agent should do, given her available information. In our case, a strategy will specify whether or not the agent should take an umbrella, given that it is or is not raining. Note that in some cases an influence diagram may contain several sequential decisions to be made by an agent. In such cases, a strategy specifies what the agent should do at each of her decisions. It is standard to assume that all information that was known for earlier decisions is available to later decisions, so the strategy will specify what an agent should do for each decision, conditioned on all prior information. After the strategies for an ID have been computed, a Bayesian network can be produced in which the decision nodes are replaced by chance nodes representing the optimal strategies for the agent. This network can then be used to predict what might actually happen in the situation. In our example, if we have determined that Waldo’s strategy is to take an umbrella if the forecast says rain, we can then query the probability that Waldo will get wet.

Multi-agent influence diagrams (MAIDs) [14] extend the framework of ID to game-theoretic situations. Syntactically, MAIDs are almost the same as IDs. The only difference is that each decision and utility node is now associated with a particular agent. A MAID represents a game in which each agent gets to choose the decisions associated with it. Once all decisions have been fixed, values of all utility nodes are determined, and each agent’s utility is the sum of the values of the utility nodes associated with it. Solving a MAID requires computing a Nash equilibrium of strategies for each of the players.

To motivate the definition of NIDs, let us return for now to the previous single-agent example. Suppose now that Waldo’s view of the world is slightly different from what we believe to be the correct model, in that Waldo is more trusting of weather forecasters, and uses a different sensor model for the forecast variable. Structurally, the ID representing Waldo’s view is the same as ours, but the values of the parameters are different. We would like to be able to answer a query such as “What is the probability, according to our view of the world, that Waldo will get wet?” This requires reasoning with both Waldo’s view and our view. We can determine Waldo’s decision function by maximizing his expected utility relative to his own model. We can then plug his decision function into our model, and compute our subjective belief in the event that he will get wet. We can also ask, “What is the cost to Waldo of using his flawed model?”, by comparing the utility he gets using his decision function with the one he would get if he used the correct model to determine his decisions.

In the previous example, Waldo’s model differed from our own only in the conditional probabilities, but it can differ in any aspect. For example, we might imagine that Waldo does not believe the forecast to depend on the actual weather at all. We might imagine that instead, Waldo listens to the Dow Jones index, which he believes to be a barometer of the

weather, in order to decide whether to take an umbrella. We also might have uncertainty as to which model Waldo uses.

We introduce the following language to allow for all of these possibilities:

A *Network of Influence Diagrams (NID)* is a rooted directed acyclic graph, in which each node is a MAID. To avoid confusion with the internal nodes of each MAID, we will call the nodes of a NID *blocks*. The root of the graph is called the *top-level model* and represents the real world from the modeler’s point of view. Edges in the graph from block U to block V are labeled $\{i, \mathbf{D}\}$, where \mathbf{D} is a set of decision variables in U belonging to agent i . Intuitively, such an edge means that in model U , we view agent i as using mental model V to make decisions \mathbf{D} . We say that the decisions \mathbf{D} in U are *modeled* by V .

For this to make sense, every decision variable $D \in \mathbf{D}$ must appear in V . However, we do not require that it be a decision for agent i in V ; it may be a chance variable instead. In both cases, we require that the parents of D in V be a subset of the informational parents of D in U . In other words, agent i cannot have more information when making decision D in V than she did in U . If she did, we could not use V to model her decision in U , because the decision rule in V would require her to examine information she does not have access to in U .

If the variable D is a chance variable for i in V , we are viewing agent i as behaving like an automaton with respect to the decision D . For example, agent i may be following a pattern or social convention. This allows one important form of bounded rationality to be captured. Agents do not always optimize all of their decisions, but may perform some of them according to fixed rules or heuristics.

There may be more than one edge out of a block U labeled by an agent i . For example, there may be two different models V_1 and V_2 , with edges $\{i, \mathbf{D}_1\}$ from U to V_1 and $\{i, \mathbf{D}_2\}$ from U to V_2 . In this situation, we are using two different models to model agent i ’s decision making processes with regard to decisions \mathbf{D}_1 and \mathbf{D}_2 . We will see later how this capability is also useful for modeling boundedly rational agents. We also allow for uncertainty over which model agent i uses for a decision. This is represented by multiple edges labeled $\{i, \mathbf{D}\}$ leaving U .

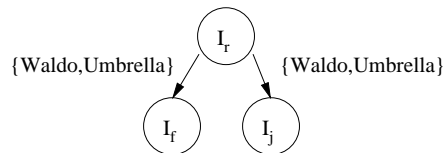
In order for the model to be coherent, we do require the following property: if $\{i, \mathbf{D}_1\}$ and $\{i, \mathbf{D}_2\}$ are two edges leaving U , then either $\mathbf{D}_1 = \mathbf{D}_2$ or $\mathbf{D}_1 \cap \mathbf{D}_2 = \emptyset$. This means that if we have uncertainty over whether some set of decisions is modeled by V , either all of them will be modeled by V or none of them will.

As an example, let us revisit our umbrella scenario and suppose that an analyst has uncertainty over which model Waldo uses to decide whether to take an umbrella, when the top-level model, denoted I_r is given in Figure 1. The analyst is unsure whether Waldo uses model I_f , where he observes the weather forecast, or model I_j , where he looks at the Dow Jones index. These models make up our NID in Figure 2(a).

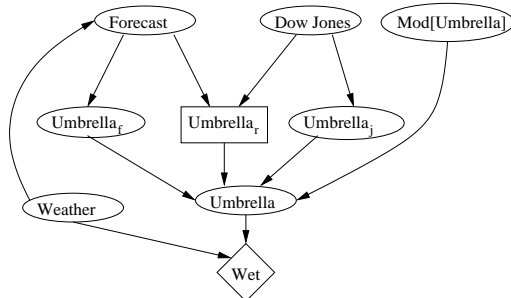
When there are multiple edges leaving U labeled with the same label $\{i, \mathbf{D}\}$, we must quantify our uncertainty over which model agent i uses for deciding decisions \mathbf{D} . This is captured by introducing a node $\text{Mod}[\mathbf{D}]$ into the influence diagram U . $\text{Mod}[\mathbf{D}]$ is a chance node, that may be influenced by other nodes of U , just like any other chance node. The values of $\text{Mod}[\mathbf{D}]$ range over the blocks V such that

there is an edge from U to V labeled $\{i, \mathbf{D}\}$. In addition, $\text{Mod}[\mathbf{D}]$ can take on the value U itself. Intuitively, when $\text{Mod}[\mathbf{D}]$ takes value V it means that decisions \mathbf{D} are actually modeled by V , and when $\text{Mod}[\mathbf{D}]$ has value U it means the decisions are actually taken according to U . (The correct model, from the point of view of U). $\text{Mod}[\mathbf{D}]$ has a conditional probability distribution, just like any other chance node.

Following our example, suppose the analyst believes Waldo to use model I_j with probability 0.7, model I_f with probability 0.2, and the real-world model I_r with probability 0.1. The top-level model I_r will now include a chance node labeled $\text{Mod}[\text{Umbrella}]$ whose only child is Umbrella . The conditional probability table(CPT) for $\text{Mod}[\text{Umbrella}]$ includes entries for I_j , I_f and I_r with corresponding probabilities 0.7, 0.2, and 0.1.



(a) umbrella NID



(b) transformed top-level ID

Figure 2: Umbrella scenario revisited

Solving a NID means computing a decision rule for every decision in every MAID in the network. This can be done easily 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 [14]. In the case that they only involve decisions of a single agent, they can be solved using a standard ID algorithm such as [18, 5]. For an internal block U , once all of its children have been solved, decision rules are available for all of the nodes in U modeled by one of U ’s descendants.

We then transform U into a new MAID U' . All chance and utility nodes of U become nodes of U' , with the same parents and conditional probability distributions. For each decision D in U , we introduce a decision node D_U into U' , with the same informational parents as D in U . We also introduce a chance node D_V for each V such that D may be modeled by V . The parents of D_V are the same as its informational parents in V , which as we stated earlier must exist in U . The conditional probability distribution for d_V

is the decision rule computed for d in V . Finally, D is made into a chance node in U' , with parents $\text{Mod}[\mathbf{D}]$, D_U and all the D_V . Its CPT is a multiplexer, with the value of $\text{Mod}[\mathbf{D}]$ determining which of the possible decisions gets assigned to D . The children of D are the children of the original decision D in the original U . Once this process has been done for every set of decisions, U' is a MAID containing the models for decisions taken outside of U and incorporating the uncertainty about which models are used. It can then be solved to obtain decision rules for the decisions that are made in U .

Returning to our example, after solving the leaf models and transforming the ID at the root, we will end up with the ID depicted in Figure 2(b). We then solve this ID, converting it to a Bayesian network in which we can answer all of the queries mentioned above.

3. APPLICATIONS AND EXAMPLES

3.1 Divided Decision Making

One of the most common approaches to simplifying complex problems is divide and conquer. In complex situations involving many decisions, agents commonly solve different decisions separately. Similarly, different decisions may be delegated to different agents. This is an important form of bounded rationality that can be modeled by our language. One of the key aspects is modeling the assumptions made by each agent about the decisions not under its control.

test the ground before deciding whether or not to drill for oil. The test is a probabilistic indicator of whether or not drilling will be profitable, but the test itself is costly. Each of the decisions is made by a separate division of the company. Unfortunately, the divisions, nicknamed “the drillers” and “the testers”, are not willing to cooperate with the analyst, leading to uncertainty over the decision-making processes of OilRon. The analyst believes that with probability 0.2, the two divisions work together rationally to determine the optimal decisions. However, with probability 0.8, the decisions are optimized separately, with each division using its beliefs about the other. The testers believe that the drillers are wild and reckless, and will drill for oil no matter what the test results are; the drillers believe that the testers will test anything and everything. The corresponding NID for OilRon is shown in Figure 3. In the real-world model I_w , the analyst’s uncertainty about the decision-making process is captured by a WORK-TOGETHER chance variable that has probability 0.2 of being true. This variable determines the model used for each of the decisions. When WORK-TOGETHER is true, both $\text{Mod}[\text{DRILL}]$ and $\text{Mod}[\text{TEST}]$ have value I_w , the real-world model, because then the decisions are taken as if by rational decision makers. When WORK-TOGETHER is false, $\text{Mod}[\text{DRILL}]$ has value I_d and $\text{Mod}[\text{TEST}]$ has value I_t , where I_d and I_t are models of the drillers’ and testers’ decision making processes respectively, described in Figures 3(c) and 3(d).

When this model is solved, we can ask queries such as “What happens when the two divisions do not work together?” It turns out that because the testers believe the drillers will always drill, they will never test, because their test can never change the drilling decision. Interestingly, the drillers believe that the testers always test, whereas in actuality they never test. This shows that we can model situations in which the individual agents hold beliefs that are inconsistent with each other.

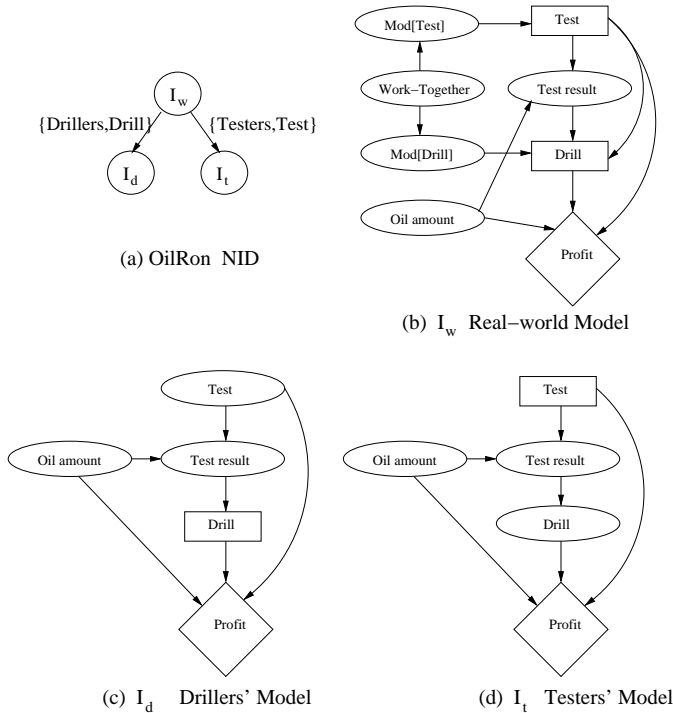


Figure 3: OilRon scenario

Consider the following scenario: Following a sharp loss in revenue of OilRon, a big oil corporation, an analyst is called in to assess decision-making policies within the company. In an exploratory oil-drilling situation, OilRon can decide to

3.2 Opponent Modeling

Real agents often use rules, heuristics, patterns or tendencies when making decisions. Furthermore, agents often reason about the workings of other agents. Consider, for example, a baseball manager ordering a pitch-out in a certain situation because the opposing manager has a tendency to order a stolen base in that situation. One of the main approaches to game playing with imperfect information is opponent modeling, in which one tries to learn the patterns exhibited by other players. NIDs provide a solid, coherent foundation for opponent modeling. They also provide a smooth integration between the opponent modeling approach and classical game theory, by allowing uncertainty over whether an agent is computing its decisions heuristically or rationally.

For example, consider the game of RoShamBo (commonly referred to as Rock-Paper-Scissors). In a single round of the game, two players simultaneously choose one of *rock*, *paper*, or *scissors*. If they choose the same item, the result is a tie; otherwise rock crushes scissors, paper covers rock, or scissors cut paper, as shown in the matrix below.

	ROCK	PAPER	SCISSORS
ROCK	(0, 0)	(-1, 1)	(1, -1)
PAPER	(1, -1)	(0, 0)	(-1, 1)
SCISSORS	(-1, 1)	(1, -1)	(0, 0)

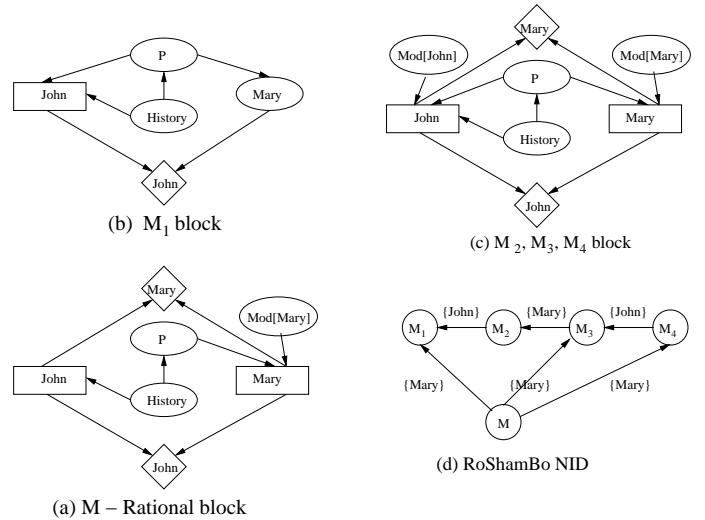
The game has a single Nash Equilibrium in which both players play a mixed strategy over $\{rock, paper, scissors\}$ with probability $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$. Therefore, if both players do not deviate from their equilibrium strategy, they are guaranteed an expected payoff of zero. In fact, it is easy to verify that a player who always plays his equilibrium strategy is guaranteed to get an expected zero payoff regardless of the strategy of his opponent. In other words, sticking to the equilibrium strategy guarantees not to lose a match, but it also guarantees not to win it!

Now consider a situation in which two players play repeatedly against each other. If a player is able to pick up the tendencies of a sub-optimal opponent, it might be able to defeat it, assuming the opponent continues to play sub-optimally. In a recent competition [2], programs competed against each other in matches consisting of 1000 games of RoShamBo. As one might expect, Nash equilibrium players came in the middle of the pack because they broke even against every opponent. It turned out that the task of modeling the opponent’s strategy can be surprisingly complex, despite the simple structure of the game itself. This is because sophisticated players will attempt to counter-model their opponents, and will hide their own strategy to avoid detection. The winning program, called Iocaine Powder [4], did a beautiful job of modeling its opponents on multiple levels. Iocaine Powder considered that its opponent might play randomly, according to some heuristic, or it might try to learn a pattern used by Iocaine Powder, or it might play a strategy designed to counter Iocaine Powder learning its pattern, or several other possibilities.

Inspired by “Iocaine Powder”, we constructed a NID for a player that is playing a match of RoShamBo and is trying to model his opponent. Suppose that John wishes to model Mary’s play using a NID. The top-level model of the NID, shown in Figure 4(a), is a “rational” block, which is simply a MAID depicting a RoShamBo round between John and Mary. Both players have access to the history of the game, which might be summarized by some statistics. Also, the history determines some signal P , which is available to Mary to use in her decision making process. In the rational block, Mary will ignore P , and play the Nash Equilibrium strategy. However, there are several alternative models of Mary’s decision. According to block M_1 , shown in figure 4(b), John believes Mary to be an automaton that follows some predictive algorithm that is dependent on the signal P . We can then solve M_1 to determine John’s best response to Mary. For example, if John thinks, based on the history, that P is most likely to tell Mary to play *rock*, then John would play *paper*. Let us denote this strategy as $BR(P)$.

According to block M_2 that is a block that models Mary’s beliefs, John’s play is modeled by M_1 . In other words, Mary believes that John plays $BR(P)$, as a result of John’s belief that Mary plays P . Therefore, she will rationally (with respect to her model) play a best response to $BR(P)$, thereby second-guessing John. Mary’s strategy in M_2 is thus $BR(BR(P))$. Following our example, in M_2 Mary would not play *rock* at all, but *scissors*, in order to beat $BR(P)$. Now, solving for John’s strategy in M_3 that is a block that models M_2 , we obtain $BR(BR(BR(P)))$. Again, following the example, this would prompt John to play *rock*, in order to beat $BR(BR(P))$. We can continue this to a third level at M_4 , in which Mary believes John plays $BR(BR(BR(P)))$, and so plays $BR(BR(BR(BR(P))))$.

Figure 4: RoShamBo scenario



The entire NID is shown in Figure 4(d). In the root block M , John models Mary as playing at one of the possible levels M_1 , M_3 or M_4 . The uncertainty is captured in the $Mod[Mary]$ variable, which also has some probability for Mary playing according to the rational block M . John can then compute his best possible play, according to his beliefs about Mary’s possible strategies.

3.3 Collusion and Alliances

In a situation where an agent is modeling multiple agents, it may be important to know whether those agents are working together in some fashion. In such situations, the models of how the other agents make their decisions may be correlated, due to possible collusion. For a simple illustration of this, consider a voting game involving 3 agents A, B and C, who are voting one of them to be chairman of a committee. Agent A is the incumbent, and will be chairman if the vote ends in a draw. Each agent would like itself to be chairman, and receives utility 2 in that case. Agent A also receives a utility of 1 if he votes for the winner but loses the election, because he wants to look good. Agents B and C, meanwhile, dislike agent A and receive utility -1 if A wins.

It is in the best interests of agents B and C to coordinate, and both vote for the same person. If B and C do indeed coordinate, it is in A’s best interest to vote for the person they vote for. However, if B and C miscoordinate, A should vote for himself to remain the chairman. In taking an opponent modeling approach, A would like to have a model of how B and C are likely to vote. A might be uncertain about whether or not B and C have discussed the vote and agreed to collude against A. This uncertainty can easily be captured in a NID.

The top-level block is shown in Figure 5. There is a node COLLUDE, which will have three possible values, *none* indicating no collusion, *B* and *C* indicating collusion to vote for B or C respectively. The nodes $Mod[B]$ and $Mod[C]$ depend on COLLUDE. If COLLUDE is *none*, $Mod[B]$ will be the top-level block. If COLLUDE is *B*, $Mod[B]$ will be a simple block describing an automaton that votes for B. If COLLUDE

is C , $\text{Mod}[B]$ will with high probability be another simple block describing a vote for C , but will with low probability will be the top-level block. This accounts for the possibility that when B and C have agreed to vote for C , B might renege. The conditional probability distribution for $\text{Mod}[C]$ is similar.

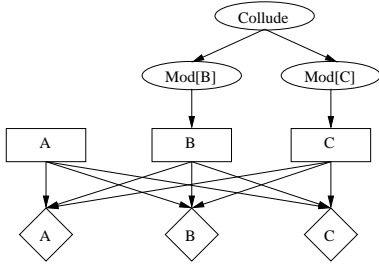


Figure 5: Collusion scenario

Moving beyond this simple example, one of the most important issues in multi-player games is alliances. When players form an alliance, they will act for the benefit of the alliance rather than purely for their own self-interest. Thus an agent’s beliefs about the alliance structure affects its models of how other agents make their decisions. When an agent has to make a decision in such a situation, it is important to be able to model its uncertainty about the alliance structure.

This issue is very important in the game of Diplomacy. In Diplomacy, seven players control different European countries with the goal of becoming the dominant player. The game proceeds in rounds. In each round, the players negotiate and then all players simultaneously submit a set of orders, and the position changes as a result. The process continues until a winner emerges. From a game-theoretic point of view, Diplomacy is a dynamic game consisting of a sequence of one-shot stage games. In every round, agents have difficult game-theoretic decisions to make.

One of the characteristics of the game Diplomacy is that there are usually alliances, but they shift, and players often have uncertainty about the alliances. One of the key components of intelligent play is the ability to exploit the uncertainty about the alliance structure. For example, England might believe that France is allied with Germany. However, England also believes that although France is allied with Germany, she might suspect that Germany is allied with England. Therefore England sends messages to France designed to make it look like Germany is allied with him, thereby increasing France’s suspicions of Germany and causing her to break her alliance with Germany. On seeing these messages, France now has uncertainty about England’s model. Perhaps Germany really is allied with him, or perhaps England is sending those messages to make her think Germany is allied with him. This type of reasoning is commonplace in Diplomacy. All these complex layers of strategic reasoning can be modeled in a NID.

3.4 Coordination

One of the difficulties with classical game theory is that while every game is guaranteed to have a Nash equilibrium, many games have more than one. When there is more than one equilibrium, the classical theory does little to predict which will be chosen. In reality, every agent, even a rational

one, will proceed according to its beliefs about which equilibrium is being played. Once again, modeling the beliefs of the agents becomes crucial.

The classic Battle of the Sexes (BoS) [16] game describes a situation in which a husband and wife would like to coordinate their activity on a given evening. The husband would prefer to go to the ballet, while the wife would like to go to the football match. They would both rather spend the evening together than apart. In matrix form, the game is

	H_B	H_F
W_B	(1, 2)	(0, 0)
W_F	(0, 0)	(2, 1)

where the entry (1, 2) for actions (W_B, H_B) means that the wife gets 1 and the husband 2 when they go to the ballet. This game has two Nash equilibria in pure strategies, where they both go to the ballet or both go to the football match. It also has a mixed strategy Nash equilibrium, but that is strongly dominated by the pure equilibria.

A puzzle about BoS and coordination games in general is this: since the game has multiple equilibria, how do players coordinate on which equilibrium to play? Schelling [19] introduced the concept of focal points as external, out-of-game factors that make the players fixate on a particular equilibrium. However, focal points are mysterious — there is no explanation within game theory itself of how they work.

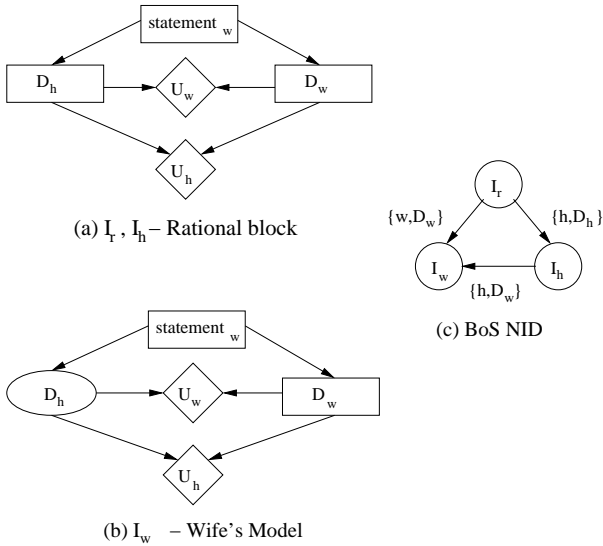
One mechanism that has been proposed for achieving coordination in the BoS is for one of the players to announce her intentions before the game. From the psychological point of view, it is understandable that this would create a focal point at the equilibrium where the player carries out her stated intention. However, from the game-theoretic point of view, the explanation remains mysterious, because the opposite focal point could just as easily have been created.

One can see this immediately by modeling the game in extensive form. First, one of the players, say the wife, announces where she will go. Both players observe this announcement. Then a BoS ensues. There is nothing binding in the wife’s announcement, nor does it have any direct impact on either player’s utility. All outcomes are still possible, and from a purely game-theoretic point of view, the announcement is irrelevant. Yet somehow we have the intuition that it is not. We believe that our intuition is founded on models of the players’ decision-making processes.

Using our language, we can describe such models that would explain how coordination comes about, for agents that are *rational with respect to their models*. The top-level block in the NID in Figure 6 is the “rational” model, which is a MAID depicting the extensive-form BoS game described above. Looking at this block alone, there is no reason to expect coordination after the wife’s statement. However, we have a model for the wife’s decision-making process, called I_w . According to this model, the wife believes that the husband will do whatever she says, so the husband’s decision D_h is turned into a chance variable, that is determined by the wife’s statement. I_w is a leaf block, so we can solve it to determine her best response to her automaton husband: because her husband does whatever she says, so should she in order to coordinate with him; furthermore, she should say “football match” in order to maximize her utility.

Thus far, we have only explained why the wife does what she says, but not the husband. If the husband is indeed the automaton the wife believes he is, then of course he will do

Figure 6: Battle-of-the-sexes scenario



what she says. But we have a more sophisticated model of the husband. The husband rationally believes he is playing a BoS, since I_h is similar to I_r . However, his model of the wife’s decision D_w is the same as ours, described by block I_w . In other words, the husband will not just do what the wife says, but he thinks the wife thinks he will! Unfortunately for him, this is enough to make him do what she says anyway. To see this, consider that we have already determined that according to I_w , the wife will do what she says. This means that in the husband’s rational model, he will substitute a behavior for the wife’s decision where she does what she says. Since he wants to coordinate with her, he will therefore do what she says.

This is only one possible NID model that leads to coordination. We are not claiming that the model that we have presented is the correct account of why coordination happens, only that it is a plausible explanation of how it happens. What we are providing is a representation language that allows a modeler to tell a clear story, using a simple model of the agents’ beliefs, that explains how an interesting phenomenon might have happened.

4. DISCUSSION AND RELATED WORK

In game theory, uncertainty is traditionally handled by the Bayesian game framework [11], in which each agent has a discrete type embodying its private information. A Bayesian game includes a set of players, a set of possible actions and a prior probability distribution over the types of all players, deemed the objective prior distribution. Associated with each type for an agent is a utility function for that agent given each possible outcome of the game. A player’s uncertainty is captured by a probability distribution over the types of other players, given her own type.

In a paper which presents formal semantics for NIDs [7], we show that every Bayesian game can be reduced to an equivalent NID. We also show that some subclasses of NIDs are equivalent to Bayesian games. This is accomplished by having beliefs over decision-making processes reduced to

types, and uncertainty over models expressed as a probability distribution over the types. In general, however, NIDs are a super-set of Bayesian games, since they can model bounded rationality—they describe not only best-response strategies, but also the model’s prediction of how agents actually play. We refer the reader to the paper for details.

NIDs provide a much better knowledge representation language than Bayesian games. The basic reason is that Bayesian games are abstract and obscure, while NIDs provide explicit, descriptive and easy to understand models. Moreover, there are a number of technical advantages of NIDs: First, Bayesian games are a flat representation. All of the salient features of the world are folded into the utility function associated with each type. The connection between chance variables and outcomes is obscured. Consider our first example, in which we had uncertainty over Waldo’s beliefs about the weather forecast model. By introducing the forecast explicitly into the model, we were able to capture this uncertainty in a natural way. In contrast, in Bayesian games, we would have to presolve each of the ID models to compute the expected utility to Waldo under each possible action given the forecast, and use that utility to distinguish between Waldo’s models.

Second, a NID may be exponentially smaller than the corresponding Bayesian game. Suppose that an agent has n decisions to make in a game. For each decision, we have two models for how the agent makes her decision. Since a type must capture everything that is salient about an agent, it must dictate which model the agent uses for all of its decisions. Thus the number of types is 2^n , but the number of NID blocks is polynomial. In this case, there is a lot of structure in the type system that is not captured by the type framework that is captured by NIDs.

Third, in Bayesian games, each type for an agent has an associated joint probability distribution over the other agents’ types. This distribution can be extremely unwieldy to represent without making strong independence assumptions. NIDs allow graphical models to be used to represent the distribution over types in a natural and compact manner. We do not require the types to be independent. In our collusion example, the types of agents B and C were correlated due to the possibility of collusion. As with other graphical models, this is not only a representational benefit but also may yield great computational savings.

Fourth, NIDs make it possible to model some forms of bounded rationality in an explicit, natural way, which is hard to do with Bayesian games. As we have shown, providing separate models for different decisions separately allows the modeling of divide-and-conquer bounded rationality, while allowing decision variables to be modeled by chance variables allows us to model agents acting according to heuristics, or agents thinking about other agents acting according to heuristics. Both of these can be done in Bayesian games, but only in an unnatural way. The first can be achieved by introducing types corresponding to the cross-product of the separate decision models. As we noted above, this also incurs blowup in the number of types. The second can be achieved by creating a utility function that forces a player to take the right action.

Finally, after solving a NID to obtain the agents’ strategies, we have a Bayesian network that is an interesting model that can be used for answering queries about the game. We can ask conditional probability queries about the situation.

Bayesian games do not provide this capability because they do not model the chance variables, only utilities.

The algorithm we presented in this paper for solving NIDs assumed that the graph structure was acyclic. This was to allow us to compute strategies in a bottom-up manner. We show in [7] that NIDs also allow for cyclic type structures. This leads to infinite chains of reasoning of the form “I think that maybe you think that maybe I think that maybe you think ...”. We are currently developing algorithms for solving cyclic NIDs.

In their work introducing MAIDs, Koller and Milch developed an algorithm for computing an equilibrium strategy for a MAID, while exploiting the local independence structure of the model. MAIDs, along with the related approach of Kearns et al [13] introduced the benefits of graphical models to games. However, both of these approaches follow the classical normative approach to game theory, assuming that all agents know the real world model and there are no discrepancies between agents’ models.

In work by Suryadi and Gmytrasiewicz [3], each agent embodies the decision process of all other agents together in a separate ID, where decisions of other agents are modeled as chance variables and given a prior CPT. A neural network is then used for updating CPTs of variables in an agent’s model from observations. In this model, the representation of agents’ model is limited to a single influence diagram for each agent. The approach can be viewed as a very special case of the one in this paper, together with an algorithm for agents to update the model parameters.

Gmytrasiewicz and Durfee [15] have developed a framework in which the building blocks are trees, where the nodes consist of payoff matrices for a particular agent. Uncertainty of agents is modeled through payoff matrices that are different, either in the numbers or in structure. Their language, like ours, is capable of expressing agents’ models of other agents’ models, and so on. However, the bandwidth for expressing uncertainty is even lower than for Bayesian games, since the agents’ do not even have types; all uncertainty must be captured in the structure of the payoff matrix and the utilities. Also, there is no capability to express uncertainty over which models agents use.

Our language serves to complement the two prevailing perspectives on bounded rationality. Ever since Simon’s challenge to the notion of perfect rationality as the foundation of economic systems [9], the theory of bounded rationality has grown in different directions. From an economic point of view, bounded rationality dictates a complete deviation from the utility maximizing paradigm, in which concepts such as “optimization” and “objective functions” are replaced with “satisficing” and “heuristics” [8]. These concepts have recently been formalized by Rubinstein [1]. From an AI point of view, an agent exhibits bounded rationality if its program is a solution to the constrained optimization problem brought about by limitations of architecture or computational resources [17]. Our approach is much closer in spirit to that of economists such as Selten than the traditional AI approach.

Finally, there is a natural application of our language to learning in games [6]. Learning agents in games maintain models of other agents. Usually, these models are quite simple, as for example in fictitious play, which assumes that strategies exist a-priori. NIDs provide a language for describing the hypothesis space of models that agents have

about other agents’ play. They are rich enough to allow agents to reason about other agents learning about them, while they learn about the other agents. In future work, we will show that NIDs provide a framework for learning the distribution over strategies and also over chance variables in separate NID blocks. Furthermore, we will investigate whether agents using NIDs can predict their opponents’ behavior, even when their strategy changes over time.

Acknowledgments

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

5. REFERENCES

- [1] A. Rubinstein. *Modeling Bounded Rationality*. MIT Press, 1998.
- [2] D. Billings. The first international RoShamBo programming competition. *International Computer Games Association Journal*, 23(1), March 2000.
- [3] D. Suryadi and P.J. Gmytrasiewicz. Learning models of other agents using influence diagrams. In *Proc. Seventh International Conference on User Modeling (UM-99)*, pages 223–232, 1999.
- [4] D. Egnor. Iocaine Powder. *International Computer Games Association Journal*, 23(1), March 2000.
- [5] F.V. Jensen F. Jensen and S. L. Dittmer. From influence diagrams to junction trees. In *UAI*, pages 367–373, 1994.
- [6] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, 1998.
- [7] Y. Gal and A. Pfeffer. Modeling agents’ beliefs using networks of influence diagrams. Technical Report TR-06-03, Harvard University, 2003.
- [8] G. Gigerenzer and R. Selten editors. *Bounded Rationality: The Adaptive Toolbox*. MIT Press, 2001.
- [9] H.A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.
- [10] R.A. Howard and J.E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.
- [11] J.C. Harsanyi. Games with incomplete information played by ‘Bayesian’ players. *Management Science*, 14:159–182, 320–334, 486–502, 1967–68.
- [12] J. Nash. Equilibrium points in n-person games. *Proc. National Academy of Sciences of the USA*, 36:48–49.
- [13] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *UAI*, pages 253–260, 2001.
- [14] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *IJCAI*, 2001.
- [15] P. Gmytrasiewicz and E.H. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems Journal*, 2000.
- [16] R.D. Luce and H. Raiffa. *Games and decisions*. New York: Wiley, 1957.
- [17] S. Russell and E. Wefald. *Do the right thing: Studies in Limited Rationality*. MIT Press, 1991.
- [18] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.
- [19] T.C. Schelling. *The Strategy of Conflict*. Harvard University Press, Cambridge, Mass., 1960.