A Language for Descriptive Decision and Game Theory

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

Abstract

In descriptive decision and game theory, one specifies a model of a situation faced by agents and uses the model to predict or explain their behavior. We present Influence Diagram Networks, a language for descriptive decision and game theory that is based on graphical models. Our language relaxes the assumption traditionally used in economics that beliefs of agents are consistent, i.e. conditioned on a common prior distribution. In the single-agent case one can model situations in which the agent has an incorrect model of the way the world works, or in which a modeler has uncertainty about the agent's model. In the multi-agent case, one can 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.

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.

However, the focus in AI so far has been on the normative aspect of decision and game theory, in which the optimal behavior of a rational agent is prescribed. AI also has a fundamental contribution to make to the *descriptive* aspect of decision theory and game theory. At the same time, there are benefits to be gained for AI from studying these sciences from the descriptive point of view. Descriptive decision theory and game theory seek to model the decision-making processes of agents; thus they are thoroughly engaged in knowledge representation. AI's tradition of thinking about representation allows us to provide languages for elegantly and precisely describing models of agents and their decision-making processes. A good language can allow a modeler to provide a clear and precise model of behavior that would otherwise have to be described vaguely or inaccurately.

In this paper we present a language for descriptive decision and game theory that is based on a novel graphical model we call an *Influence Diagram Network (IDN)*. This language is rich enough to express uncertainty of agents not only over states of knowledge, as in traditional game theory, but also over models of other agents. Underlying the language is a network of influence diagrams (IDs) [14] and their multi-agent extensions [11].

In classical usage, an influence diagram plays a dual role. It is both a model of the real-world situation faced by an agent and the model the agent uses to determine its optimal decision. Identification of these two models makes sense from a normative point of view, since an agent can do no better than optimize its decisions relative to its own model of the world. However, if one wants to reason about the effects of these decisions, it is important to consider the possibility that the agent's model is incorrect. One can then consider questions such as the value to the agent of learning the correct model. Our language makes this distinction explicit, by making the agent's decision-making model a central part of the language. As a

result, one can model his uncertainty as to the agent's decision-making model. One may also analyze situations where an agent is boundedly-rational, in the sense that it does not optimize all its decisions simultaneously.

The idea of explicitly representing an agent's decision-making models is powerful in the multi-agent case. Here, our language is based on multi-agent influence diagrams, or MAIDs, an extension of IDs developed by Koller and Milch [11]. MAIDs represent a situation in which multiple agents make decisions, and the utilities of agents are influenced by their own decisions and those of other agents. Our language allows each agent's decision model, and agents' models of each others' models, and so on, to be represented explicitly as MAIDs. By explicitly modeling agents' models, our language can express fundamentally different models than those allowed by MAIDs and other classical game-theoretic formalisms such as Bayesian games [9]. In those formalisms, agents are assumed to have a common prior distribution over the information received by other agents and their utilities. In addition, an assumption of common knowledge of rationality is invoked to justify a Nash equilibrium solution. Both assumptions are subject to criticism and hard to justify a priori. Our framework allows both of these assumptions to be dropped, but in a controlled manner, where violations of the assumptions are explicitly modeled. Indeed, once we allow agents to have models of each other, there is no reason to assume that they should be consistent with each other. Nevertheless, we still assume that agents are *rational with respect to their models*. This assumption allows us to compute the predicted behavior of agents who hold models described in our language.

We demonstrate the power of our language for descriptive game theory using an example, showing how coordination can be achieved in a battle-of-the-sexes game [15] under a variety of assumptions about the players' models. The coordination problem is traditionally swept under the rug in game theory, by invoking the magical concept of focal points [18]. By definition, these are "out-of-model" — no attempt is made to explain how they work within the models themselves. Our language can provide such an explanation, showing how the agents' beliefs about each other lead to a coordination outcome.

2 Single-Agent Case

We first concentrate on describing a single agent's model of the world. For this purpose, the building blocks of our language are *influence diagrams (IDs)* [14]. 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 BN. Edges leading into decision variables represent information that is available to the agent at the time the decision is made. A standard assumption, known as *perfect recall* or *no forgetting*, is that any information available to the agent for one decision is also available for future decisions.

Traditionally, we take a given ID as representing the agent's view of its decision problem, i.e., the agent uses the ID to decide what to do. However, we might be concerned with describing situations in which the agent's view of the world is different from our view as analysts, or one in which we have uncertainty about the agent's view. Consider the following scenario: Waldo is about to leave his house, and needs to decide whether or not to take an umbrella.

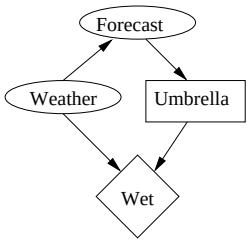


Figure 1: real-world ID of umbrella scenario

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. Suppose now that Waldo's view of the world is slightly different from ours, in that Waldo is more trusting of weather forecasters, and uses a different conditional probability table (CPT) for the forecast node. 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:

An *Influence Diagram Network (IDN)* is a rooted DAG, in which each node is an ID. The root of the DAG is called the *top-level model* and represents the world from an analyst's point of view. Edges in the DAG are labeled $\{\mathbf{D}, P\}$, where $\mathbf{D}$ is a set of decision variables belonging to an agent and P is a probability between 0 and 1. Intuitively, the meaning of an edge $\{\mathbf{D}, P\}$ from U to V is that with probability P, the decisions $\mathbf{D}$ in U are determined by the model V. If a decision d in U is not determined by any sub-model, we invoke the standard rationality assumption that U itself is used to determine the decision for D.

We place the following restrictions on IDNs:

1. If $\{\mathbf{D}, P\}$ is the edge from nodes U to V, and d is a decision in $\mathbf{D}$, we require that d appear in both U and V, and say that the decision d in U is *modeled* by V. The point is that an edge from U to V labeled by d indicates that model U is not used to make decision d, but V is used instead. We let M_d^U be the set of nodes by which decision d in U is modeled.

2. For any $V \in M_d^U$, the set of informational parents of d in V must be a subset of the informational parents of d in U. The reason for this restriction is that it is hard to make sense of an agent forming a decision rule on the basis of information it does not actually have. Moreover, it is hard to see how such a decision rule would play in the actual world, where the agent has to take action without getting the information it expected. Because of this rule, we have to take care when modeling the situation where Waldo listens to the Dow Jones index. We must provide this information to Waldo in the top-level model, even though it is not relevant to the weather.

3. For any two decisions d_1 and d_2 in U, we require that $M_{d_1}^U$ and $M_{d_2}^U$ are either equal or disjoint. If they are equal we say that d_1 and d_2 in U are modeled together. We can write $M_{\mathbf{D}}^U$ for any set of decisions $\mathbf{D}$ that are modeled together.

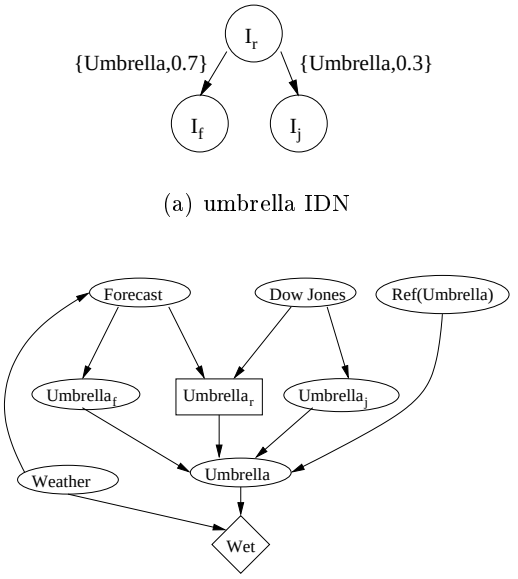
4. Let $M_{\mathbf{D}}^U = V_1, \ldots, V_n$, and let the probability on the edge from U to V be $P_V^{U,\mathbf{D}}$. Then $\sum_{V \in M_{\mathbf{D}}^U} P_V^{U,\mathbf{D}} \leq 1$. If the sum is strictly less than 1, the remaining probability is allocated to the case that $\mathbf{D}$ is modeled in U after all. We let $P_U^{U,\mathbf{D}}$ denote 1 minus the sum.

Solving an IDN means computing a decision rule for every decision in every ID in the network. This can be done easily in a bottom-up fashion, from the leaves of the DAG to the root. The leaves are simply IDs, and can be solved using a standard ID algorithm such as [17, 5]. For an internal node 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 influence diagram. For any set of decisions $\mathbf{D}$ that are modeled together, we introduce a new random variable *Ref*($\mathbf{D}$) to capture our uncertainty about where those decisions are made. The possible values of *Ref*($\mathbf{D}$) are U and every $V \in M_{\mathbf{D}}^U$. *Ref*($\mathbf{D}$) has no parents, and the probability of any value W is $P_W^{U,\mathbf{D}}$. For each decision $d \in \mathbf{D}$, we introduce a decision node d_U, with the same informational parents as d in U, and a chance node d_V for each $V \in M_{\mathbf{D}}^U$. The parents of d_V are the same as its informational parents in V, which must exist by restriction 2, while its CPT is the decision rule computed for d in V. Finally we introduce a chance node d, with parents *Ref*($\mathbf{D}$), d_U and all the d_V. Its CPT is a multiplexer, with the value of *Ref*($\mathbf{D}$) determining which of the d_W 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 has been transformed into an ID containing the models for decisions taken outside of U and incorporating the uncertainty about which models are used. It can then be solved like any other ID, to obtain decision rules for the decisions that are made in U.

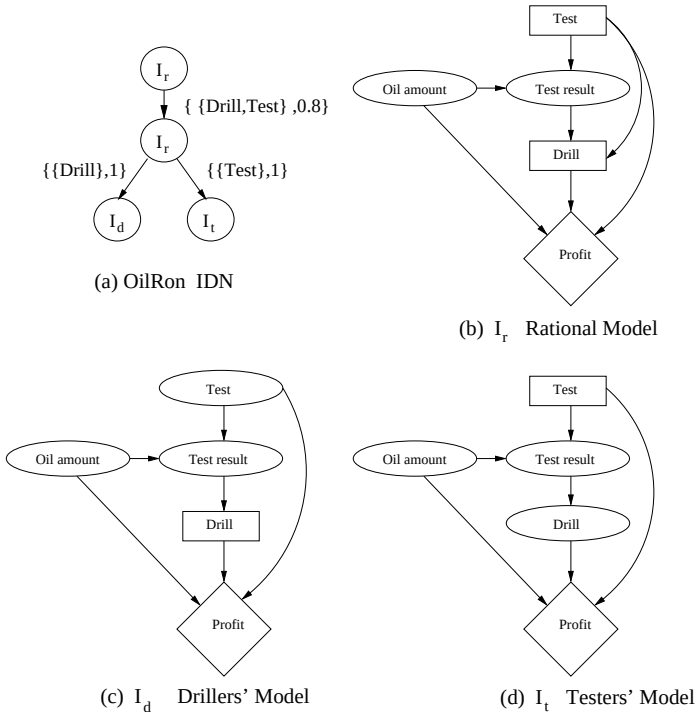
As an example, let us revisit our umbrella scenario and suppose that an analyst has uncertainty over which model Waldo uses to decide whether to take an umbrella, when the top-level model, denoted I_r is given in figure 1. With probability 0.7, we believe Waldo to use model I_f, where he observes the weather forecast, and with probability 0.3, we believe Waldo to use model I_j and look at the Dow Jones index. These models



(a) umbrella IDN

(b) transformed top-level ID

Figure 2: Umbrella scenario revisited



(a) OilRon IDN

(b) I_r Rational Model

(c) I_d Drillers' Model

(d) I_t Testers' Model

Figure 3: OilRon scenario

make up our IDN in figure 2(a). 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 BN in which we can perform to answer all of the queries mentioned above.

Once we have computed a probability distribution over all of the models at the top-level of the IDN, we are able to answer a wide variety of queries. For example, we can find the probability of Waldo getting wet by "plugging in" the CPT for umbrella from Waldo's model into the top-level model and inferring from it the probability $P(wet = T)$. Similarly, we can compute the cost to Waldo of using the wrong model by computing $E[U_{RW}^{RW}] - E[U_{RW}^{waldo}]$ where $E[U_{RW}^{A}]$ is defined as the expected utility under the real-world random variables given the decisions as in model A.

Our language is rich enough to model boundedly rational decision makers, i.e. agents that use heuristics to make some decisions, or to split up complex decision problems into separate decisions. 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 scenario, OilRon can decide to 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, OilRon is rational, in the sense that perfect recall is maintained and the testers make the test result available to the drillers. However, with probability 0.8, the divisions behave "dysfunctionally", meaning that they do not trust each other and decisions are optimized separately. 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 are drones who will test anything and everything. The corresponding IDN for OilRon is shown in figure 3. Both of the ID nodes I_d and I_t are almost identical to the real-world model; in I_d, the decision node *test* is a CPT, that is deterministically set to true, while in I_t, the decision node *drill* is a CPT, also set deterministically to true.

There is an additional intermediate I_r node in the model. This node is needed because the dysfunctionality of the two divisions is correlated. With probability 0.2, they are both rational, and with probability 0.8, they are both dysfunctional. Without the "dysfunctional" node, the decisions for test and drill would have been split up right away and their dysfunctionality would not have been correlated.

It is interesting to note what happens when the dys-

functional model is solved. 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. On the other hand, the drillers' incorrect model will not be as damaging because they will always choose the optimal decision given whatever test results are available. Therefore, this analysis reveals that OilRon should spend its management improvement efforts on the testing department.

One can regard the two divisions of OilRon as independent agents. Indeed, in interactions between agents that occur in strategic situations, agents often do not disclose their private information and strategies are devised separately, as was the case with the divisions of OilRon. Our language is most useful when we consider multiple agents who have models of each other's decision-making processes. We therefore extend our language to describe belief hierarchies of multiple agents.

3 Multi-Agent Case

We now extend our language to describe multi-agent environments. To this end, we use MAIDs, an extension of IDs to model multi-agent interaction [11]. A MAID is similar to an ID, except that each decision and value node is associated with a particular agent. MAIDs are representations of strategic situations, where agents choose the values of their decision nodes to maximize their own utilities, cognizant of the fact that other agents are making their decisions to maximize their utilities. Koller and Milch introduced methods for obtaining game-theoretic Nash equilibrium solutions for MAIDs.

Adapting our formalism to the multi-agent case and using MAIDs requires minimal changes: We let A be a set of agents, and we redefine an IDN to be a rooted DAG where each node consists of a MAID. Edges on the graph are labeled by the set $\{a, \mathbf{D}, P\}$ where $a \in A$, $\mathbf{D}$ is a set of decisions for agent a and P is a probability between 0 and 1. The same restrictions on the language hold as before.

The language is actually quite permissive about the way one agent models the decisions of other agents. Suppose V is reached from U by an edge labeled $\{a, \mathbf{D}, P\}$. Intuitively, this means that the model in U for how agent a makes decisions $\mathbf{D}$ is given by V (with probability P). Now, since U contains decisions for other agents, so might V. The language places no stipulations on how they are modeled. For example, agent a might "know" how a certain decision is made, and model it as a chance node. Or, agent a might believe that another agent has more information available to make its decision than it really has. Or, agent a might imagine that some chance nodes in U are actually under the control of another agent, and turn them into decision nodes in V.

In addition, it is straightforward to change our bottom-up solution algorithm to use MAID inference algorithms at each step rather than standard ID algorithms. At each level, we obtain a Nash equilibrium solution for the behavior of each of the players. We can then implement that solution as a mixed strategy at the next level up.

The only tricky issue involves situations with multiple equilibria. A given MAID may have many equilibria. This provides for many different possible strategies at the next level up, none of which can be ruled out a-priori. In theory, the number of equilibria in an IDN may be worse than exponential in the depth of the IDN. Therefore, enumerating all the equilibria will in general be unfeasible, and an equilibrium-selection criterion will need to be applied at each level.

From a modeling point of view, a leaf of an IDN describes a situation in which agents have common knowledge of rationality with respect to the game described at that leaf. One can thus limit rationality assumptions to particular aspects of a game. In the case that a leaf only involves a single decision-making agent, it describes a situation in which that agent is playing a best response to its model of the opponents, where the opponent is modeled by an automaton. Now, suppose that an internal node involves decisions by two agents, say agent a and agent b, and agent a's decisions is modeled by another node, which may be a simpler game. In this case, agent b is playing a best response to a more sophisticated model of the opponent, in which the opponent is playing a particular game rationally. Thus our language provides very sophisticated mechanisms for describing the bounded rationality of agents, and agents responding to other agents' bounded rationality.

We now demonstrate the power of our language by showing how it can be used to explain phenomena that are problematic in classical game theory. The classic Battle of the Sexes (BoS) [15] game describes a situation in which a husband and wife would like to coordinate their activity on a particular 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.

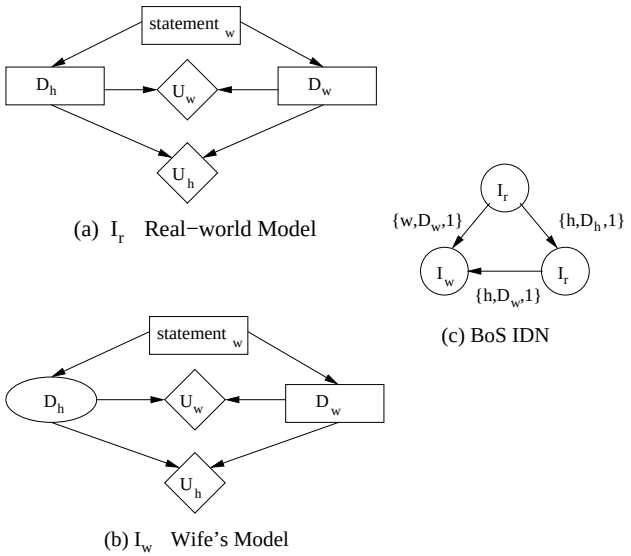
A puzzle about BoS and coordination games in general is this: since the game has multiple equilibria, how do players coordinate on which equilibrium to play? Schelling [18] introduced the concept of focal points as external, out-of-game factors that make the players fixate on a particular equilibrium. 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*. One possible model is shown in Figure 3. The top-level node in the IDN is the "rational" node, which is a MAID depicting the extensive-form BoS game described above. Looking at this node 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 node, 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.

Figure 4: Battle-of-the-sexes scenario



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 what she says. But we have a more sophisticated model of the husband. In fact, the husband rationally believes he is playing a BoS. However, his model of the wife's decision D_w is the same as ours, described by node 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 model that leads to coordination. In fact, any IDN in which the internal nodes are all I_r, and the leaves are all I_w, will work, as we can see by inductively computing the players' strategies from the leaves upward. Thus any thought that the players have that the husband will follow the wife is sufficient to lead to coordination, no matter how distant.

We are not claiming that one of these models is the "true" explanation of how coordination happens. What we are claiming is that our language allows us to state and frame an explanation of this sort in a very precise way.

4 Discussion and Related Work

In game theory, uncertainty is traditionally handled by the Bayesian game framework, due to Harsanyi[9], 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*. A player's uncertainty is captured by a probability distribution over the types of other players, given her own type. This conditional probability distribution can be computed from the objective prior distribution using Bayes Theorem. Also, the utility of each player is defined as a function from the space of types and type-contingent strategies to the real line. If we assume that the objective prior distribution is known to all players, and that players' beliefs are consistent in that they are all computed by conditioning on the common prior distribution, then standard techniques can be used to solve the game, leading to a *Bayesian equilibrium*. Since beliefs are consistent, we need only take into account each player's beliefs about his opponents' distribution of types and type-contingent strategies when computing the equilibrium.

Our framework diverges from the Bayesian games formalism in a number of fundamental ways. First, our framework relaxes the assumption that players' beliefs are consistent. There is no a-priori reason that they should be, and we allow situations to be modeled in which players have mutually incompatible beliefs. However, when players beliefs are consistent, a model in our language can be reduced to a Bayesian game, and the solution is a Bayesian equilibrium. Second, our model allows the models that agents have of other agents models to be encoded explicitly, in a way that cannot be captured in types. Third, types encode, in a single variable, all of a player's private knowledge about the state of the world and its utility function. This is a "low bandwidth" representation of uncertainty. All uncertainty is captured in a single variable, and uncertainty about other agents' reasoning processes cannot be expressed at all. In contrast, our language provides an expressive representation of uncertainty, including uncertainty over agents' models.

Fagin and Halpern offer a formalization of reasoning about probabilistic beliefs [10], in which an agent can assign a probability P to his knowledge of a formula ϕ. Fagin and Halpern provide a semantics for their language that is also applicable to our IDN language, once the solution process has computed decision functions for all the decision variables. More recently, the PEL model [12] was proposed as a representation of agents' beliefs and decision-making processes. This model both reasons about beliefs and computes optimal strategies for a single agent. The PEL model follows the assumptions layed down by Harsanyi, in that all agents use the same model, while differences in beliefs between agents are expressed by conditioning the model on an agent's private observations. 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. However, this algorithm assumes that all agents know the real-world model, and it does not allow for discrepancy 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 [13] 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 [8], 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" [7]. 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 [16].

Our language is capable of modeling bounded rational-

ity in several ways. Agents may be bounded in their knowledge of how the world works or of how other players think. Situations in which an agent uses a heuristic to make a decision can be modeled in our language by turning the decision variable into a chance variable in the agent's model. Also, agents may simplify their decision making processes by splitting multiple decisions into separate decision problems. Finally, agents may model other agents as boundedly rational, thereby becoming boundedly rational themselves.

The work in this paper was inspired by the recent research on opponent modeling in games, and in particular by the RoShamBo (rock-paper-scissors) competition [2]. In the competition, programs competed against each other in a series of 1000 games of Roshambo. Nash equilibrium players came in the middle of the pack. The winner of the competition was a program called Iocaine Powder [4], that 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. This type of reasoning can be captured very nicely in our language, by giving Iocaine Powder a probability distribution over the opponent's models, some of which themselves include models of Iocaine Powder.

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. IDNs provide a rich language for describing the hypothesis space of models that agents might 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. The next step in this research is to develop a learning algorithm so that agents can update their IDN models of other agents based on observed play.

Acknowledgments

Thanks to David Parkes, David Sullivan, Barbara Grosz and Wheeler Ruml for useful feedback. This work was supported by an NSF Career Award.

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] G.Gigerenzer and R.Selten editors. *Bounded Rationality: The Adaptive Toolbox.* MIT Press, 2001.

[8] H.A.Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[9] J.C.Harsanyi. Games with incomplete information played by 'bayesian' players. *Management Science*, 14:159–182,320–334,486–502, 1967-68.

[10] J.Fagin and J.Y.Halpern. Reasoning about knowledge and probability. *Journal of the Association for Computing Machinery*, 41-2:340–367, 1994.

[11] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *IJCAI*, pages 1027–1034, 2001.

[12] B. Milch and D. Koller. Probabilistic models for agents' beliefs and decisions. In *UAI*, pages 389–396, 2000.

[13] P.Gmytrasiewicz and E.H.Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems Journal*, 2000.

[14] R.A.Howard and J.E.Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.

[15] R.D.Luce and H.Raiffa. *Games and decisions.* New York: Wiley, 1957.

[16] S. Russell and E. Wefald. *Do the right thing: Studies in Limited Rationality.* MIT Press, 1991.

[17] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[18] T.C.Schelling. *The Strategy of Conflict.* Harvard Univesity Press, Cambridge, Mass., 1960.