A Stochastic Language for Modelling Opponent Agents

Gerardo Simari Amy Sliva Dana Nau V.S.Subrahmanian
University of Maryland
College Park, Maryland, USA
{gisimari,asliva,nau,vs}@cs.umd.edu

ABSTRACT

There are numerous cases where a reasoning agent needs to reason about the behavior of an opponent agent. In this paper, we propose a hybrid probabilistic logic language within which we can express what actions an opponent may take in a given situation. We present the syntaxis and semantics of the language, and the concept of a Maximally Probable Course of Action.

Categories and Subject Descriptors

I.2 [**Computing Methodolgies**]: Artificial Intelligence

Keywords

Logic for agent systems, Formal models of agency

1. INTRODUCTION

There are numerous multiagent applications where we need to build a *model* of an opponent agent. For example, consider a computer implementation of a multi-player game such as *Diplomacy* or *Civilization* — in such a game, a human (or agent) player may wish to have a model of how another player might play in order to decide on the best move or moves. Likewise, in strategic decision making applications such as negotiations, it is a good idea to have one or more models of how an opponent might behave.

Alternatively, we have been building an application for reasoning about the actions that certain tribes involved in the opium business in the Pakistan Afghanistan borderlands might take. We have been developing a set of rules that would allow us to predict what possible reactions could arise when US forces take certain actions. This has clear benefits as US forces would like to take those actions that have a desirable outcome with a high probability.

The two classes of applications - games and real world cultural reasoning scenarios - have one significant difference. Games are limited in scope and are governed by strict rules. Cultural models are less strict and operate under very different rules. Second, and more important, in the case of games, one can try to mine rules automatically from large collections of past games. In the case of cultural models, this is far harder. We are yet to see a single mined collection of rules dealing with any culture at all.

Moreover, most opponent models are *handcrafted* in a language that is selected on a case by case basis. In this paper, we develop a single uniform language called SOMA within which opponent models may be expressed *by a human* for a wide variety of applications ranging from automated game playing programs to strategic adversarial reasoning problems. Of course, in the case of games, automatically extracting models from past histories is relatively easy compared to performing similar extractions in cases where relevant histories are harder to find.

In Section 2, we develop a formal syntax for SOMA-programs. Section 3 develops a formal model-theoretic semantics for SOMA-programs, and Section 4 includes a declarative definition of the ‘most probable course of action” (MPCOA) for the opponent.

2. SYNTAX OF SOMA-RULES

Throughout this paper, we assume the existence of a *reasoning agent* which wishes to model and reason about an *opponent agent.* At any given point in time, the reasoning agent may have a state. This state may allow it to, for example, record the actions she has observed an opponent making and/or include the beliefs of the reasoning agent. Without loss of generality, we will assume that the state of the reasoning agent is logically describable using an alphabet consisting of a finite set of *predicate symbols* (each with an associated arity), a finite set of *constant symbols*, and an infinite set of variable symbols. We assume that no function symbols are present. As usual, a *term* is either a constant symbol or a variable symbol. If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are all terms (resp. constants), then $p(t_1, \ldots, t_n)$ is an atom (resp. *ground* atom). The *observable state* at any given point in time is a set of ground atoms.

In addition, the reasoning agent believes in the existence of some finite set of elementary actions that the opponent can take. This set is represented by a *finite* set $\mathcal{A}$ of symbols $\mathcal{A}$ called *action symbols* disjoint from the set of predicate symbols. Every $a \in \mathcal{A}$ has an associated arity; if a is an n-ary action symbol, and $t_1, \ldots, t_n$ are all terms (resp. constants), then $a(t_1, \ldots, t_n)$ is an *action atom* (resp. *ground* action atom).

DEFINITION 2.1 (ACTION FORMULA). *Every action atom is an action formula. If P and Q are action formulas, then so are $(P \wedge Q), (P \vee Q)$, and $\neg P$.*

An action formula denotes a complex action. For example, $\neg a$ is the action where the opponent does not perform a. $(a \wedge b)$ is the complex action where the opponent does both a and b.

DEFINITION 2.2 (SOMA-RULE). *If $B_1, \ldots, B_n$ are atoms, $0 \leq \ell \leq u \leq 1$ are probability values, and P is an action formula, then*

$$P : [\ell, u] \quad \leftarrow \quad B_1 \wedge \ldots \wedge B_n$$

The above rule is an elementary SOMA-rule *if P is an action atom.*

Intuitively, suppose $\mathcal{S}$ is the current state of the reasoning agent (including any beliefs about the opponent agent). The above rule states that if the state $\mathcal{S}$ satisfies all the atoms $B_1, \ldots, B_n$, then the opponent agent will take action(s) P with a probability between ℓ and u.

DEFINITION 2.3 (SOMA-PROGRAM). *A (elementary) SOMA-program is a finite set of (elementary) SOMA-rules.*

Here is an (admittedly simplistic) example of a simple SOMA-program – a more sophisticated and carefully developed program of this kind that extracts the rules using sample data such as *Gallup* poll data might enable an unprincipled politician to determine what stances will get him votes.

EXAMPLE 2.1. *Consider a simple example involving* politics. *By examining a set of surveys, we might be able to write down a set of rules applicable to an arbitrary member m of the population (opponent) as follows:*

$$\begin{aligned} vote\_myway : [0.2, 0.3] &\quad \leftarrow \quad democrat. \\ vote\_myway : [0.3, 0.6] &\quad \leftarrow \quad promise\_cut\_tax. \\ \neg vote\_myway : [0.3, 0.9] &\quad \leftarrow \quad promise\_cut\_tax. \end{aligned}$$

The first rule says that I (the reasoning agent) expect m to vote for me with 20 to 30% probability if I am a democrat. The probability goes up to 30 to 60% if I promise a tax cut (independently of whether I am a democrat or not). The probability that the example citizen either does not vote for me is 30 to 90% if I promise a tax cut.

SOMA-programs are variations of generalized probabilistic (gp) logic programs [Ng and Subrahmanian 1993] - the heads of SOMA-rules are more expressive than gp-logic program rule heads, but conversely the bodies of gp-logic programs are more expressive. The semantics of SOMA-programs builds on that of gp-programs and is briefly described below.

3. SEMANTICS OF SOMA-RULES

A reasoning agent can use a SOMA-program to represent its beliefs about an opponent agent and estimate what the opponent might do in a given situation.

A *course of action* (COA for short) that the opponent might take is any finite set of ground action atoms. Suppose $\mathcal{C}$ is the set of all possible courses of action. We assume the existence of a function ϕ which takes a course of action and a state as input, and returns either 0 or 1 denoting whether the course of action is feasible or not (a course of action may be infeasible, for example, if the actions in it are mutually incompatible or if the actions cannot be executed in the current state due to pre-conditions that do not hold or if the the new state that results does not satisfy some integrity constraints). In the following, we will write $gr(\mathcal{A})$ to denote the set of all ground action atoms. We first define what it means for a COA to satisfy an action formula P.

DEFINITION 3.1. *A course of action C **satisfies** a* ground *action formula P, denoted $C \mapsto P$ if:*

1. *P is an action atom and $P \in C$, or*
2. *P has the form $P_1 \wedge P_2$ and C satisfies P_1 and P_2, or*
3. *P has the form $P_1 \vee P_2$ and C satisfies P_1 or P_2, or*
4. *P has the form $\neg P_1$ and C does not satisfy P_1.*

A SOMA-interpretation defined below assigns a probability to each COA.

DEFINITION 3.2 (SOMA-INTERPRETATION). *A* SOMA-interpretation *I is a mapping from $\mathcal{C}$ to $[0, 1]$ such that $\Sigma_{X \in \mathcal{C}} I(X) = 1$.*

Intuitively, $I(X)$ represents the probability that the opponent will perform exactly those actions in X. Clearly, the opponent must choose some course of action which is why $\Sigma_{X \in \mathcal{C}} I(X) = 1$. Let us now return to our political example.

EXAMPLE 3.1. *We have two COAs: $C_1 = \{\}$; $C_2 = \{vote\_myway\}$. A SOMA-interpretation I_0 may assign 0.3 and 0.7 respectively to these two COAs.*

DEFINITION 3.3. *Suppose $\mathcal{S}$ is a state and I is a SOMA-interpretation. I is said to **satisfy** a ground rule*

$$P : [\ell, u] \quad \leftarrow \quad B_1 \wedge \ldots \wedge B_n$$

with respect to state $\mathcal{S}$ iff

1. *$\{B_1, \ldots, B_n\} \not\subseteq \mathcal{S}$ or*
2. *$\ell \leq \Sigma_{\phi(C,\mathcal{S})=1 \,\wedge\, C \mapsto P} I(C) \leq u$.*

We say that I satisfies a rule if and only if it satisfies all ground instances of the rule.

Let us return to our political example.

EXAMPLE 3.2. *Consider the first rule of example 2.1 and suppose the current state $\mathcal{S}_0 = \{democrat, promise\_cut\_tax\}$. The first rule is* not satisfied *by the interpretation I_0 of Example 3.1 because $I_0(vote\_myway) = 0.7 \notin [0.2, 0.3]$. Had the probability interval in the head of this rule been $[0.5, 0.8]$, then the rule would have been satisfied by I_0.*

We use the usual notion of consistency.

DEFINITION 3.4 (CONSISTENCY). *A SOMA-program Π is* consistent *with respect to state $\mathcal{S}$ if there is at least one SOMA-interpretation that satisfies all rules in Π.*

The following definition shows that we can associate a set of linear constraints with any SOMA-program.

DEFINITION 3.5 ($\mathsf{CONS}(\Pi, \mathcal{S})$). *Suppose* Π *is a* SOMA-*program and* $\mathcal{S}$ *is a state. The* constraints associated with Π, *denoted* $\mathsf{CONS}(\Pi, \mathcal{S})$ *associates a variable* p_i *with each course of action* C_i *(denoting the probability of* C_i*) and are defined as follows:*

1. *If*
$$P : [\ell, u] \quad \leftarrow \quad B_1 \wedge \ldots \wedge B_n$$
is a ground instance of a rule in Π *and* $\{B_1, \ldots, B_n\} \subseteq \mathcal{S}$, *then the constraint:*
$$\ell \leq \Sigma_{\phi(C_i,\mathcal{S})=1 \wedge C_i \mapsto P} p_i \leq u$$
is in $\mathsf{CONS}(\Pi, \mathcal{S})$.

2. $\Sigma_{\phi(C_i,\mathcal{S})=1} p_i = 1$ *is in* $\mathsf{CONS}(\Pi, \mathcal{S})$.

EXAMPLE 3.3. *Consider the "political" example shown in Example 2.1 and suppose the state in question is* $\mathcal{S}_0 = \{democrat, promise\_tax\_cut\}$. *Let* C_1 *and* C_2 *be the two COAs from Example 3.1. The constraints in this case are:*
$0.2 \leq p_2 \leq 0.3$;
$0.3 \leq p_2 \leq 0.6$;
$0.3 \leq p_1 \leq 0.9$;
$\Sigma_{\phi(C_i,\mathcal{S})=1} p_i = 1$.

The following result shows that a SOMA-program Π is consistent w.r.t. a given state iff the set $\mathsf{CONS}(\Pi, \mathcal{S})$ of constraints has a solution.

4. MAXIMALLY PROBABLE COAS

When a reasoning agent builds a SOMA-program about another opposing agent, his goal is to determine what the opponent will do in a given state. We define below, the lower probability that the opponent will take a given course of action.

DEFINITION 4.1 (LOWER PROBABILITY OF A COA). *Let* Π *be a* SOMA-*program and* $\mathcal{S}$ *be a state. The* lower probability, $\mathsf{low}(C_i)$ *of course of action* C_i *is defined as follows:*

1. $\mathsf{low}(C_i) = 0$ *if* $\phi(C_i, \mathcal{S}) = 0$, *i.e.* C_i *is not valid.*

2. *Otherwise,* $\mathsf{low}(C_i) =$ **minimize** p_i ***subject to*** $\mathsf{CONS}(\Pi, \mathcal{S})$.

The *upper probability*, $up(C_i)$ of course of action C_i can be defined in a similar manner. Note that for an given course of action C, we cannot *exactly* determine the probability that the opponent will do C. This is true even if all rules in Π have a point probability in the head — this is because our framework does not make any simplifying assumptions (e.g. independence) about the probability that the agent will perform action b, given that he will do action a.

The following example explains what happens in the situation of our political example and the state $\mathcal{S}_0$ introduced in example 3.1.

EXAMPLE 4.1. *As we saw in example 3.3, there are 4 constraints that must be satisfied at once in order to find an interpretation that satisfies all rules in* Π *with respect to* $\mathcal{S}$. *It can be seen that* $\mathsf{low}(C_1) = 0.3$, *and* $\mathsf{low}(C_2) = 0.3$. *Similar calculations lead to* $\mathsf{up}(C_1) = 0.9$, *and* $\mathsf{up}(C_2) = 0.3$.

5. RELATED WORK

We are not aware of much work on probabilistic agent reasoning - the work that exists focuses on agents that are reasoning about an uncertain state and trying to figure out how best to act in such a situation. However, there are a few relevant works that we build on top of.

[Gmytrasiewicz and Durfee 1992] have developed a logic of knowledge and belief to model multiagent coordination. Their framework permits an agent to reason not only about the world and its own actions, but also to simulate and model the behavior of other agents in the environment. In an earlier paper, they show how one agent can reason with a probabilistic view of the behavior of other agents so as to achieve coordination.

[Dix et al. 2000] develop a model of probabilistic agents on top of legacy code — in their framework, an agent reasons about uncertainty in its state. However, there is no uncertainty about what the agent will do. A subsequent work by these authors [Dix et al. 2005] extends this framework to the case of temporal probabilistic reasoning agents that can reason with uncertainty about when certain things will be true in the state.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a hybrid logic/statistical language called SOMA within which a reasoning agent can represent knowledge about the probability that an opponent agent may take one or more actions. We have developed a formal model theoretic semantics for such agents.

In the full version of this paper, we make several additional contributions: first, we prove that checking consistency of SOMA-programs is NP-hard. We have identified syntactic fragments of SOMA-agents for which consistency checking is polynomial. We show that checking if a given course of action has a probability exceeding a given threshold is NP-hard. We have developed an exact algorithm to find an MPCOA and several heuristic algorithms that can approximate the most probable course of action very quickly and with high degrees of accuracy.

7. REFERENCES

DIX, J., NANNI, M., AND SUBRAHMANIAN, V. 2000. Probabilistic agent reasoning. *ACM Transactions of Computational Logic 1,* 2, 201–245.

DIX, J., KRAUS, S., AND SUBRAHMANIAN, V. 2004. Heterogeneous temporal probabilistic agents. *ACM Transactions of Computational Logic 5(3).*

GMYTRASIEWICZ, P. AND DURFEE, E. 1992. A Logic of Knowledge and Belief for Recursive Modeling. In *Proceedings of the 10th National Conference on Artificial Intelligence.* AAAI Press/MIT Press, San Jose, CA, 628–634.

HANKS, S. AND MCDERMOTT, D. 1994. Modeling a dynamic and uncertain world I: Symbolic and, probabilistic reasoning about change. *Artificial Intelligence 65(2)*, 1–55.

NG, R. AND SUBRAHMANIAN,V.S. 1993. A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases. *Journal of Automated Reasoning 10 (2)*, pps 191–235.