New Challenges in Multi-Agent Intention Recognition: Extended Abstract

Gal Kaminka
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
galk@cs.cmu.edu

Jan Wendler*
Department of Computer Science
Humboldt University Berlin
D-10099 Berlin, Germany
wendler@informatik.hu-berlin.de

Galit Ronen
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
gronen@andrew.cmu.edu

Introduction

Recently, there has been an emergence of interest in understanding how observations of the actions of agents can be used as the basis for inference of the unobservable state of these agents, in order to improve the ability of the observer to respond to these agents. In particular, there is increasing interest in the area of *Agent Modeling*, which investigates mechanisms allowing an agent to acquire, maintain, and infer knowledge of other agents. This area unites plan-, goal-, and intent-recognition under a single umbrella with user-modeling, behavior-recognition, belief ascription, agent tracking, etc.

Traditionally, agent modeling researchers have explored techniques in which two agents are involved e.g., (Kautz & Allen 1986; Charniak & Goldman 1993; Lesh, Rich, & Sidner 1999). In such techniques one agent observes the actions of another agent, and attempts to infer its unobservable state features, such as intent, goal, or plan. These techniques are successful in many cases, and new techniques are still being investigated, e.g., (Pynadath & Wellman 2000).

However, the transition from agent-modeling techniques, where an observing agent is monitoring the state of another agent, to *multi-agent modeling*, where the observing agent is monitoring the actions of more than one agent, present new challenges that have not been previously addressed by agent modeling researchers. These include both computational challenges, such as bandwidth and computational load, as well as conceptual challenges, such as reasoning about previously unseen behavior of teams of agents.

This extended abstract outlines some of these current key challenges in multi-agent modeling, and the steps we have begun to take in address these challenges, specifically in the context of agents that are collaborating with each other. We focus on the following key challenges:

- *The Monitoring Selectivity Challenge*: Given finite bandwidth and computation resources, how can an agent modeling system reason about *large* teams with many agents? How can is selectively use observations of perhaps only a few agents, to successfully carry out tasks which require modeling an entire team?
- *The Incomplete/Incorrect Knowledge Challenge*: How can an agent modeling system reason about unfamiliar teams, for which it has inaccurate, incorrect, or even no models?
- *The Model Challenge*: What type of models should an agent-modeling system reason about in modeling multiple agents? For instance, are plans and goals (traditionally, the building blocks for the "solution space" of an agent modeling process) useful for all multi-agent modeling tasks?

We address these challenges using an abstract framework for building multi-agent modeling applications. This framework, called *Socially-Attentive Monitoring* (Kaminka & Tambe 2000; Kaminka, Pynadath, & Tambe 2001), emphasizes using knowledge of the relationships (social structures) between modeled agents' states, and the procedures that the agents use to maintain them (norms and social laws). Using this framework as a guiding principle, an agent-modeling to alleviate the significant uncertainty and computational challenges facing multi-agent modeling systems. We present a set of promising results which address the incomplete/incorrect knowledge challenge by focusing on learning relationships separately from learning models of behavior, and that demonstrate that social models of interaction between agents can be very useful for addressing significant multi-agent modeling tasks.

The Monitoring Selectivity Problem

Traditionally, agent-modeling techniques have been investigated in a context of two agents—the observer and the observed (Kautz & Allen 1986; Charniak & Goldman 1993; Lesh, Rich, & Sidner 1999; Pynadath & Wellman 2000). However, in multi-agent modeling, the monitoring agent must observe and reason about potentially many agents. These new settings for the modeling task present significant challenges.

On one hand, if an agent attempts to observe all other agents, then in general it will face significant bandwidth

*This research was carried out while the author was visiting Carnegie Mellon University.

and computational difficulties (Jennings 1995). As the number of agents scales up, bandwidth requirements grow: For instance, the agent would need to observe all other agents continuously, which is in general not always possible, as agents become physically and logically separated from each other. In addition, computational requirements grow: If a monitoring agent is busy modeling all others, a significant computational task (Kautz & Allen 1986; Pynadath & Wellman 2000), it may run the risk of not having enough computational resources available to carry out its other tasks: After all, multi-agent modeling is a perceptual sub-task in service of an overall task for the monitoring agent, such as adversarial planning, coordination, visualization, failure detection, etc.

Thus the designer of a multi-agent system often builds the system such that there is some selectivity in observing and monitoring others. However, if the designer ventures too far, or if the environment physically constrains the design in such ways, the observing agent may find it self with insufficient observations, providing only partial information about the state of observed agents. Thus the observing agent will have uncertainty about the state of the monitored agents, which may hinder the observing agent from carrying out its tasks, such as coordinating with the other agents, planning interactions with them, etc.

The key challenge, which we refer to as as the *Monitoring Selectivity Problem,* is thus to identify, for a given task (a) how much modeling of other agents is necessary to carry out the task; and (b) techniques that will be able to carry out such modeling. overcoming uncertainty and computational burdens. Over recent years, we have been investigating modeling algorithms that provide concrete answers to these two questions in several complex, dynamic, multi-agent domains.

The techniques we investigate are guided by *socially-attentive monitoring*, a guiding principle which emphasizes using a model of the expected relationships between monitored agents' states to complement reasoning about any other previously-known application/agent models (Kaminka & Tambe 2000; Kaminka, Pynadath, & Tambe 2001). We focus in this extended abstract on utilizing knowledge of *collaborative* relationships between team-mates in service of different modeling tasks. Agents in a team are dependent upon each other, and their important state features, such as beliefs about the world, plans, and goals, will therefore be dependent as well. These dependencies can be identified by the designer, by collaboration engines (e.g., GRATE* (Jennings 1995), COLLAGEN (Rich & Sidner 1997), etc.), or by theory (Grosz & Kraus 1996).

Consider the following example, following (Kaminka, Pynadath, & Tambe 2001), that demonstrates multi-agent modeling with and without the use of knowledge of a particular collaborative relationship among agents: In this example, a set of heterogenous agents is integrated into a distributed application (team) to run on the Internet. Different agents are running on machines across the U.S., yet with focused communications between them, the agents are tied in together to function as a team. Yet due to their distribution, their actions and state are mostly unobservable to any single human operator sitting in one of these locations. For instance, it is difficult for such an operator to view a computer monitor hundreds of miles away.

To solve this problem, a multi-agent intention-recognition system was built to provide real-time visualization of the state of each agent. In theory, the system could have relied on report-based monitoring, in which each agent continuously updates the visualization system as to the agent's state. However, such a solution does not scale very well, since it requires modifying agents to send these reports (a problem with legacy and proprietary systems), and it adds significant costs to the computational and bandwidth requirements, since agents must devote some of their resources to generating and transmitting these reports.

We therefore proposed to visualize the state of the agents based on their *routine* communications, used as observations by a multi-agent modeling system. The initial solution attempted to model each agent independently of its peers, by utilizing an array of probabilistic plan-recognizers, in which each observed agent was modeled separately. But due to the scarcity of observations (agents communicate routinely only once per 20 state changes, on average—and some don't communicate at all), this approach proved unsatisfactory, resulting in 3-4% average accuracy in actual system runs (Kaminka, Pynadath, & Tambe 2001).

To address this difficult challenge, we focused on social knowledge (please see (Kaminka, Pynadath, & Tambe 2001) for details). First, although in the initial solution each agent was modeled independently, in reality agents were expected to be dependent on each other: They were supposed to switch *together* from one abstract mode of operation to the next (as part of any such abstract mode each agent would carry out different action). Thus agents were supposed to be in agreement as to the mode the team is in. Agreement on the joint plan to be executed by team-members is indeed common in teamwork applications (Jennings 1995; Tambe 1997). Second, the routine communications between agents during task execution were somewhat predictable, and in fact such predictions were amenable to learning. By eliminating modeling hypotheses that did not conform to the agreement relationship (i.e., hypotheses where agents were not in agreement), or did not conform to the predicted communications (i.e., hypotheses where agents switched mode without sending a predicted message), the accuracy of visualization was increased to 84% on average (up to 97% in some experiments). Furthermore, by restricting the modeling algorithm to only these hypotheses, we were able to realize computational space and time savings of about 90%.

The Knowledge Challenge

One of the key assumptions typically made in plan-recognition, user-modeling, and other agent modeling investigations is that the set of possible explanations for the observations is known a-priori in the form of a plan-library, i.e., the recognition problem is cast as a set membership query (Kautz & Allen 1986). However, as agent modeling techniques are being applied in multi-agent systems, this assumption is often problematic. For instance, in building a coach agent for RoboCup soccer games (Noda *et al.* 1998)—one of our current projects—one cannot assume that the behavior of opponent soccer teams will be explained correctly by an existing plan-library.

We are tackling this challenge by trying to reason separately about a task's plan-library, i.e., the set of recipes associated with carrying out a particular task (possibly by multiple agents), and the social plan-library, i.e., the set of relationships maintained and procedures used by the same agents as part of their task execution. Our hope is that this separation will yield not only computational benefits, but will also allow a designer to correctly characterize the nature of the incompleteness or incorrectness of the plan-library in question, for a given multi-agent modeling task.

We are investigating this problem in the context of two projects. The first projects deals with building multi-application assistant agents (in which the relationships are unknown), the second with building a soccer coach for the RoboCup soccer simulation environment (in which the task knowledge is unknown). Here, we focus on the multi-application assistant project.

The vision of the intelligent assistant agent has been explored in many different investigations, and of course is now finding commercial applications (for instance, as the Microsoft Office assistant). However, the traditional focus of such applications were on assisting a single user in the context of a particular application being used. For instance, the Microsoft Office assistant is able to provide detailed help about writing letters in Word, or equations in Excel. The usefulness of engaging in user-modeling for such collaborations between an application and a user has also been demonstrated in (Lesh, Rich, & Sidner 1999).

However, a user does not always use a single application in service of carrying out a single task. Consider a typical *Java* or *Python* development scenario. Much of the power of developing in these two different computer languages lies in their large, useful, standard libraries. As a result, a typical developer often finds herself switching back and forth between the programming editor (e.g., *Emacs*) used for writing code, and an online documentation browsing tool (e.g., Netscape) used to explore the depths of the online documentation for the standard library.

A traditional assistant approach would conceptually have an *Emacs*-assistant actively trying to help the user edit the program, and a *Netscape*-assistant helping the user browse the web-page. These two conceptual assistants would each see the user being active for a few minutes, then becoming inactive (in the context of the specific application) for a few minutes more, only to become active again later on, and so on. What is conceptually missing here is not detailed knowledge about what each application is capable of doing, or how a user uses each single application.

Instead, the missing knowledge is concerned with how a developer uses and coordinates these two applications together to accomplish a single task—writing code effectively. A better intelligent assistant would need to recognize such relationships and utilize them to provide better assistance (for instance, by predicting needed web-pages based on code being written). However, it is difficult to predict all the possible relationships that can occur between applications: After all, there are indeed many different possible tasks that a user may be carrying out, and enumerating all possible application combinations is not a practical approach.

We are therefore attempting to learn to recognize very general patterns of such dynamic interactions based on past observations of the user interacting with multiple applications. The key idea is to assume that all interactions between applications are equally-likely, and then recognize statistically significant patterns that emerge where the user interacting with one application is correlated with future interactions of the user with another application. To do this, we are currently adopting the *Multi-Stream Dependency Detection* (MSDD) algorithm (Oates & Cohen 1996) for reasoning about such observations of complex applications (the algorithm was originally built to reason about simple atomic observations). Furthermore, we are developing the infrastructure necessary to monitor multiple applications without modifying them, and this is proving to be a key technical challenge.

The Model Challenge

A final challenge we address here is the challenge of finding useful models to reason about as part of a multi-agent modeling system. Traditionally, plans (sequences of actions), beliefs, and goals (ending states) are used as the models that an agent-modeling application reasons about, e.g., (Kautz & Allen 1986; Charniak & Goldman 1993; Pynadath & Wellman 2000). However, the application of agent-modeling techniques in multi-agent settings breeds new challenges: For instance, the need to reason about relationships as first-class modeling objects.

Consider the following example, which rose in the soccer coach-building project briefly mentioned above. One of a coach-agent tasks is to automatically identify weaknesses in the capabilities of teammates, and to offer advice to counter such weaknesses. As an example, we wanted our coach to be able to improve player's passing abilities (the players are also software agents interacting with the coach in the RoboCup soccer simulation). Naturally, it is relatively easy for the coach to recognize passes that failed, versus those that succeed. However, a more difficult challenge is for the coach to rank *all successful* passes according to their quality: To do this, the coach must differentiate not between a failed attempt and a successful pass, but instead between two successful passes of different qualities.

We approached this problem by looking at passes as an instance of coordinated activity between two agents. In fact, passing conforms to a very common coordination process (Malone & Crowston 1994) composed of a producer (the kicker), who produces and delivers a resource (the ball) to a consumer (the intended receiver), *at the right time and in the right way.* Thus the coach agent does not need to recognize a plan, or a goal, for the two agents, but instead recognize to what degree they are engaged in a particular coordination relationship.

We built alternative models of producer-consumer coordination processes which the coach was able to recognize. Such models differed, for instance, in where they placed the optimal interception point for the receiver of the ball, the intended velocity vector of the receiver at the time the ball was to be intercepted, the velocity vector of nearby opponents, etc. However, a key idea in all of these models was that a perfect coordination conforming to the model would require minimal effort on the part of the receiver in intercepting the ball, where such effort is measured in deviation from the intended velocity vector of the receiver. In other words, a perfectly coordinated pass is one where the receiver is intercepting the ball exactly where and when it had *intended* to go all along, without needing to slow down, change course, or speed up. Based on these models, the coach ranks observed passes, in essence specifying which passes were most closely matched by which models. In other words, the coach is recognizing different coordination relationships.

To verify our approach, the coach was run on the recorded games of the RoboCup evaluation sessions (Kaminka 1998), data from controlled experiments in which dozens of RoboCup teams were run against a fixed opponent, in an effort to create a scientifically-meaningful data allowing for ranking teams. The results have been encouraging: In both the 1998 and 1999 data sets (dozens of mega-bytes of data, thousands of simulation cycles), we have found a correlation (on the order of 0.5) between the ranking of a team in terms of its pass success-rate (how many passes are successful), and conformity with two particular models of coordination that we have developed: In one of these models the receiver was assumed to intend to continue its velocity vector at the time of the kick, and in the other, the receiver was assumed to be slowing down to a halt starting at the the time of the kick. More details are provided in (Wendler, Kaminka, & Veloso 2001).

Summary

As multi-agent systems grow in popularity in research and in actual real-world applications, the role for multi-agent modeling techniques becomes increasingly critical. However, the transition from traditional intention-recognition (agent-modeling) settings, where a single agent is modeling another, to settings where an agent is modeling many other agents, is raising many difficult challenges to current techniques. In particular, three of these challenges are the monitoring selectivity problem, the plan-library incompleteness/incorrectness challenge, and the model challenge. We are actively investigating ways to address these challenges, within the framework of socially-attentive monitoring, which focuses on the social structures and procedures within the monitored multi-agent system. We have provided a summary of current preliminary results demonstrating the usefulness of this approach.

References

Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64(1):53–79.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group actions. *Artificial Intelligence* 86:269–358.

Jennings, N. R. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75(2):195–240.

Kaminka, G. A., and Tambe, M. 2000. Robust multi-agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research* 12:105–147.

Kaminka, G. A.; Pynadath, D. V.; and Tambe, M. 2001. Monitoring deployed agent teams. In *Proceedings of the International Conference on Autonomous Agents*.

Kaminka, G. A. 1998. The multi-agent systems evaluation repository. http://www.cs.cmu.edu/∼galk/Eval/.

Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proceedings of the National Conference on Artificial Intelligence*, 32–37. AAAI press.

Lesh, N.; Rich, C.; and Sidner, C. L. 1999. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modelling (UM-99)*.

Malone, T. W., and Crowston, K. 1994. The interdisciplinary study of coordination. *ACM Computing Surveys* 26(1):87–119.

Noda, I.; Matsubara, H.; Hiraki, K.; and Frank, I. 1998. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence* 12(2–3):233–250.

Oates, T., and Cohen, P. R. 1996. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 346–354.

Pynadath, D. V., and Wellman, M. P. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 507–514.

Rich, C., and Sidner, C. L. 1997. COLLAGEN: When agents collaborate with people. In Johnson, W. L., ed., *Proceedings of the International Conference on Autonomous Agents*, 284–291. Marina del Rey, CA: ACM Press.

Tambe, M. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research* 7:83–124.

Wendler, J.; Kaminka, G. A.; and Veloso, M. 2001. Automatically improving team cooperation by applying coordination models. In *The AAAI Fall symposium on Intent Inference for Collaborative Tasks.* AAAI Press.