A Context-Based Approach to Detecting Miscreant Behavior and Collusion in Open Multiagent Systems

Larry Whitsel and Roy Turner
Department of Computer Science
University of Maine
Orono, ME 04469–5752 USA
{lwhitsel,rmt}@cs.umaine.edu

Abstract

Most multiagent systems (MAS) either assume cooperation on the part of the agents or assume that the agents are completely self-interested, for example, in the case of bidding and other market-based approaches. However, an interesting class of MAS is one that is fundamentally cooperative, yet open, and in which one or more of the agents may be self-interested. Once self-interested agents are allowed, there is the potential for an agent to be a *miscreant*: to behave due to its own goals, either intentionally or otherwise, so as to be working against the goals and best interests of the MAS itself. Detecting such agents is tricky, whether directly or via reputation brokers. Approaches for doing this may work in one situation, yet fail in others. Even more difficult is the problem of detecting collusion between agents, when two or more work together against the best interests of other agents or the MAS.

In this paper, we report on a project that is beginning to address this problem using a context-based approach. Features of the MAS' situation are used by a subset of the agents to identify it as an instance of one or more known contexts. Knowledge the agent(s) have about those contexts can then be used to directly detect miscreant behavior or collusion or to select the appropriate technique for the context with which to do so. The work is based on context-mediated behavior (CoMB), and it develops a new form of collusion detection called society-level analysis of motives (SLAM).

Most work on multiagent systems (MAS) has either assumed that the agents are all cooperative (e.g., in cooperative distributed problem solving approaches [Durfee(1999)]) or all self-interested (e.g., in contracting and bidding approaches [Sandholm(1999)]). An interesting case, however, is a MAS where the fundamental intent is for it to be cooperative, but which may include self-interested agents. This class of MAS corresponds to many real-world systems, for example open client-server networks such as the Web, Wi-Fi networks, and so forth. In this kind of system, it is important to recognize when agents are at odds, either intentionally or unintentionally, with the goals of the system—when they are, as we term them, *miscreants*.

Detecting miscreant agents is difficult. It essentially is a problem of trust: Do we trust a particular agent (1) to be able to abide by the rules of the society, as well as the spirit of those rules, and (2) to be willing to do so? The first part of trust is called the agent's *capability*, while the second is its *trustworthiness*.

A very important kind of miscreant behavior is *collusion*. We define collusion as multiple agents either acting in unison or according to a secret agreement that in either case violates the intention of the society. It is essentially unsanctioned coalitions between agents.

There have been many approaches to deciding whether to trust an agent or not. These fall in several categories: socio-cognitive modeling [Gambetta(1990)]; stochastic game theory approaches [Goeree and Hold(1999)]; learning approaches, such as reinforcement learning [Iszuierdo and Izquierdo(2008)] and genetic algorithm-based [Birk(2001)] approaches; and reputation systems [Josang et al.(2007)Josang, Ismail, and Boyd]. While each of these works well in some situations, they do not all work equally well in all situations.

Collusion detection is difficult due to the variety of ways in which it can manifest. For example, explicit collusion often occurs with the fact of the collusion itself being hidden, for example, by a priori agreements, by using covert communication channels, or by masking any communication over shared channels. The agents involved may act differently under different circumstances, and the time course of their collusive behavior can vary, sometimes to intentionally mask the collusion. Implicit collusion can occur without any communication between the agents involved at all, and it is ephemeral: agents that collude in one situation may compete in another.

Even though in general there is great variation in miscreant behavior and collusion, in many domains there will be patterns that, if identified, can help detect the behavior. A simple example of this is the case in which two agents are known to have explicitly colluded in the past. When an agent observes these two agents entering the MAS, then this defines a context in which it should suspect collusion to be likely. There are other, society-level features as well that define contexts where miscreant behavior or collusion is likely. For example, in a MAS where there is some competition between agents for rewards (e.g., some types of contracting systems), there is adequate motive for self-interested agents to cheat, and implicit collusion should be watched for in which two agents "gang up" on a leader. We use the term *society-level analysis of motive* (SLAM) for the process of detecting collusion and other forms of miscreant behavior using these society-level features.

To capitalize on these patterns of features, we are developing an approach to handling miscreant agent behavior and collusion based on our prior work in context-mediated behavior (CoMB) for intelligent agent control [Turner(1998),Turner(1994)]. This approach explicitly represents contexts that are important from the standpoint of detecting miscreant behavior. These are represented as knowledge structures called *contextual schemas* (c-schemas), which in addition to describing contexts, also contain prescriptive knowledge about how to behave while in the context. One or more trusted agents in the MAS monitor the situation, constantly attempting to diagnose it as an instance of one or more contexts it knows. When this is possible, then the c-schemas representing the contexts can directly provide hypotheses about possible miscreant behavior and collusion as well suggestions for techniques to detect or confirm such behavior.

Domain

We are primarily interested in open systems in which there is some overall set of rules or expected standards of conduct, but in which agents are themselves self-interested. Examples of such systems would include a LAN or a wireless network, artificial markets, open computing grids, or even the Web itself.

In order to provide a concrete testbed for our work, we chose a toy domain that is simple, yet still incorporates features applicable to important classes of real-world multiagent systems. Our domain is a variant of the Liar's Poker game.

Liar's Poker is a turn-taking, zero-sum, partially-observable game of dice. Five dice are rolled to generate a hand that is ranked and bid as a poker hand. In a player's turn, it makes a concealed roll and announces a *bid*, or statement of what hand it has. The subsequent player, called the challenger, either challenges that bid or accepts it. If the bid is accepted, then it is the challenger's turn to bid, with the next player in turn becoming the challenger. If the bid is challenged, then if it was a true bid (that is, it accurately represents the hand), then the challenger loses the turn and a point is transferred from it to the bidder; if it was a lie, then the bidder loses, and a point is transferred to the challenger. What makes the game more interesting is that on each turn, the bid has to beat the previous bid. Thus, there is an incentive to challenge bids, since otherwise the risk of not being able to bid a higher hand passes to the challenger. Play continues until there is a challenge. After a challenge, a new game begins with the challenger.

In the variant we have chosen, the bidder is constrained to bid the maximum score that can be composed from the hand. (Thus bidding "two fours" when the bidder in fact had "three fours" is counted as an untrue bid). This variant was chosen to make the game more analogous to auctioning, in which the goal is to discover the true utility price of a transaction.

In this domain, we treat lying as the miscreant behavior we attempt to detect. In order to model some kinds of miscreant agent detection (based on reputation systems), we also allow commenting on play by agents, where comments can be intentionally misleading. The domain, although simple, is rich enough to model both miscreant behavior as well as collusion between agents.

A simulator was constructed for this domain [Whitsel(2010)] and various miscreant agent behavior detection mechanisms were implemented and compared. What we learned from this was that no single such mechanism is sufficient in all situations, which led to the current work.

Detecting Miscreant Behavior

Previous work on detecting miscreant behavior typically treats the problem as one of making social trust decisions. Within this broad category, work has typically focused on three areas: approaches based on socio-cognitive trust utility, reputation-based approaches, and machine learning approaches [Ramchurn and Jennings(2005)].

With respect to socio-cognitive trust, Falcone and Castelfranchi [Falcone and Castelfranchi(2001)] propose a model of belief-based degrees of trust. There are two kinds of trust [Castelfranchi and Tan(2001)]: trust that the agent is *capable* of performing as advertised, and *trustworthiness*, that is, whether or not the agent will perform what it agrees to. Their approach uses game theory to determine if we should trust the agent, in this case, enough to delegate to it. This requires the payoff from the decision to trust to outweigh the belief-weighted payoff of a decision not to trust.

One can define different strategies within this framework (e.g., [Axelrod(1984)]) based on game theory, such as "always defect",[1] "always cooperate", "grim trigger" (cooperate until the other player defects, than always defect thereafter), "tit-for-tat" (defect if the other agent defects, but cooperate when it cooperates).

Which strategy is appropriate depends on features of the game being played. Axelrod [Axelrod(1984)] showed that the tit-for-tat strategy provides the best results over a large number of games in the Iterated Prisoner's Dilemma. For multiple, mixed-motive games in which Nash equilibria do not predict what should be done, Macy and Flache [Macy and Flache(2002)] showed that within a narrow range of aspiration levels, adaptive agents can rapidly converge on a cooperative self-reinforcing equilibrium (stochastic collusion).

If we treat our toy domain in this manner, then it would suggest that the game will rapidly converge on a cooperative, self-reinforcing equilibrium in which each agent will be predisposed to accept the prior bid (cooperate) with the expectation that the next agent will similarly cooperate with it. Such a game would continue until one agent is finally forced to make an impossible bid.

[1]In our toy world, cooperating is accepting a bid, while defecting is challenging.

A clever agent, however, would recognize this and make a bid that would result in the next player being forced to accept the loss; although this would not directly accrue points to the miscreant (so-called since it would have to lie to do this), it would reduce its exposure to future risk by effectively restarting the game, thus reducing the hand it will have to bid on the next round.

An alternative to game theoretic approaches is a reputation-based approach to making trust decisions. In a reputation-based approach, each action is observed and a record of the participants, their actions, and the results are stored. Some reputation systems have a central repository, while others have multiple repositories based on local information. Some systems require direct observation for the reputation, while others accept witness testimony [Mui et al.(2002)Mui, Mohtashemi, and Halberstadt, Josang et al.(2007)Josang, Ismail, and Boyd]. An example of a reputation system is Scrivener [Nandi et al.(2005)Nandi, Ngan, Singh, Druschel, and Wallach], which regulates traffic in a peer-to-peer (P2P) system.

In our toy system, a reputation system could help a challenger decide what to do based on what other agents in the system suggest. For example, if an agent says it should challenge, and the reputation system identifies the agent as capable to make such a suggestion and reputable, then the challenger might decide to take the advice.

Reputation systems are subject to a number of common attack patterns [HOffman et al.(2009)HOffman, Zage, and Nita-Rotaru], including self-promoting (false augmentation of reputation), whitewashing (letting reputation degrade, then exiting the system to escape the consequence), slandering, denial of service attacks, and collusion (orchestrated attacks). Miscreants can seek to avoid the ramifications of their past actions by one or more of these attacks.

The third class of approaches to miscreant behavior detection is based on machine learning. For example, Bush–Mosteller reinforcement learning has been used to test machine learning in 2x2 trust games [Iszuierdo and Izquierdo(2008)]. Evolutionary selection has also been used, for example to solve a continuous-case, n-player Prisoner's Dilemma using genetic algorithms [Birk(2001)] and, using clonal selection, for social trust decisions [de Castro and Von Zuben(2002)].

Each of these approaches has its own strengths and weaknesses that determine which situations it is appropriate for. Elsewhere, we report on simulation experiments that compare the techniques in our toy domain [Whitsel(2010)] under different circumstances. While no single technique is appropriate for all situations, the group of techniques can be thought of as a toolbox from which a MAS can, once the context is known, select the appropriate tool.

In order to give trusted agent contextual knowledge to help it detect miscreant behavior, we must first pay some attention to which features of the situation have implication for which techniques, that is, which features define the space of contexts having important implications for miscreant behavior detection. We have begun to do this, and additional work is ongoing.

One feature that can indicate the likelihood of miscreant behavior is the degree to which the MAS rewards its participants. When reward for actions is relatively low, there may be little incentive for even self-interested agents to act in such a way as to circumvent MAS rules. However, with reward levels being higher, then the motivation to cheat is increased.

Coupled with this, possibly inseparably, is how likely the miscreant behavior is to be detected and what the consequences would be. If cheating is easy and or if the penalties are low, then agents are likely to make rational decisions to cheat to achieve rewards.

Another feature that impacts the likelihood of misbehavior is how easily the MAS' safeguards against cheating[2] can be detected by agents and how effective they are.

[2]Broadly defined; we also include in this term manipulating the MAS according to its rules, though against its intent.

Features of the agents participating in the MAS also affect the likelihood of miscreant behavior. Obviously, if all the agents are designed by the MAS' designers, then trust can increase, but the MAS may be able to reason about the likelihood of cheating even if this is not the case. For example, if the MAS knows the reasoning abilities of an agent, it may in some cases be able to predict its responses to the ongoing situation, and so predict whether or not it will behave. Even when the reasoning processes may not be so transparent, the MAS may be able to reason about an agent's likely behavior based on what it knows in general about the class of agent or about the particular agent, e.g., from past interactions.

Features of the task domain itself may allow the MAS to predict which miscreant behavior detection techniques are likely to work in the context. For example, if the task is basically a simple game, then game theoretic approaches may be best used, whereas in other situations (and where it is possible), reputation-based approaches might be better.

Detecting Collusion

Collusion can be implicit or explicit. Implicit collusion can arise without any formal agreement between the agents. Indeed, it can arise without any communication (in-channel or out-of-channel) between the agents at all. For example, two agents may independently realize that a third is close to a victory in a game or to winning a bid and begin to tacitly cooperate to bring the agent back into parity, or event or defeat it. Another example would be if agents realize another agent is weaker than they and so collude to harm that agent to improve their own standings.

Detecting explicit collusion may be as simple as intercepting communication between the colluding agents, but this is likely to be difficult. Communication can be encoded, possibly even to the extent that the fact that it *is* encoded (which can itself tip off others that something is amiss) is undetectable. In addition, the communication may be covert, or out-of-channel, communication. For example, an agent might communicate with another by a particular agreed-upon pattern of movements (for physical agents) that look innocent to others, or by taking actions to modify some system-level parameter, such as paging rate, I/O rate, or the availability of some resource [Moskowitz and Kang(1994)] (for software agents).

Since decoding or even identifying collusive communication is likely to be difficult, if not futile, we take the approach of looking for other society-level features in order to detect both implicit and explicit collusion. We refer to this as society-level analysis of motive (SLAM).

The question is, what are those society-level features that can be used to allow trusted agents in a MAS to detect collusion? One possibility has to do with agent actions. Each action can be characterized as being beneficial, neutral, or harmful to each other agent in the system, as well as to the system as a whole. If an agent's actions give a disproportionate benefit to another agent, then collusion between the two should be suspected.

It is more complicated than it seems to measure the benefits of an action to another agent, however, due to the time at which the benefit accrues. Actions may give an immediate benefit to another agent, for example, when a player in Liar's poker bids the "surrender" hand of all sixes to let the next player win, or when the player chooses not to challenge the previous player, and so assumes that player's risk. On the other hand, an action may have little immediate benefit, but more later, for example, when a player intentionally under-bids its hand so that a conspirator later in the playing order does not face a higher bid by the time play gets to it. There are more possibilities, as well, for when benefits accrue.

We will use a process of *disproportionate benefit analysis* to detect possible collusion between agents. Actions will be labeled as beneficial, neutral, or harmful by summing their effects over time on each other agent and comparing the effects to the average level of benefit of all actions in the society.

A side-effect of this analysis will be to also identify an agent's aggressiveness even if collusion is not present. For example, we may find that some agent is always more harmful or more helpful than is the norm to all other agents, or some particular class of agents. This can be used to help detect non-collusive miscreants at the society level, and at the individual agent level, it can be used to adjust an agent's responses to the aggressive or altruistic agent. In our toy domain, for example, an agent may use this information to determine that it is likely to be challenged or not by an aggressive player, which provides a good heuristic for choosing to bid its actual hand or to lie.

Another feature that can indicate collusion is uncharacteristic behavior by one or more agents, particularly when paired with other changes in the situation. For example, if a normally neutral agent suddenly becomes more aggressive or docile when a new player enters the game, then a reasonable hypothesis is that the two occurrences are linked. While there may be other reasons for the changed behavior (e.g., a past history of adverse or beneficial interactions with the new agent), the possibility of collusion between the two agents should at least be considered.

Other features that could be indicative or collusion have to do with detecting paired features of agents, as one would expect. For example, implicit collusion could be found by measuring the correlation between actions and comparing that correlation with the societal mean. In implicit collusion, we would expect, in general, for the agents to take actions that are similar: "ganging up on" an opponent, for example. On the other hand, for explicit collusion, although this can happen as well, there is a richer set of behavior available due to the ability of the colluding agents to agree on their roles and the ability to arrange to share risks and rewards. In explicit collusion, we would often expect the agents involved to take complementary actions.

Similarly, we would expect some collusive relationships to manifest as paired postures (e.g., aggressive/submissive) or as postures of one agent linked to the presence of another agent. For example, consider a pair of collusive agents in our domain of Liar's Dice. We might expect to see one agent intentionally inflating the score of the second by surrendering whenever possible and by challenging non-cooperating agents aggressively. A pattern another agent would see would be that of an agent acting weak when interacting with a particular agent and aggressive when interacting with the others.

Of course, sophisticated agents would attempt to obfuscate explicit collusion. Consequently, any pairing of actions or postures would have to be detected statistically, over time.

Other attributes of the MAS agents can also serve as features predictive of possible collusion. For example, agents owned by the same organization or even arriving together into society might be worth watching for signs of collusion.

Although we have earlier noted the difficulty of detecting explicit collusion, due to the agents likely attempting to hide communication, a MAS should still be alert to features that possibly predict that such communication is occurring. An example would be encrypted, nonsensical, or unexpected message traffic on the MAS' normal communication channels. Known covert channels could also be monitored, for example, to look for suspicious patterns of CPU utilization, file locking, or network traffic. Out-of-character behavior for an agent that occurs soon after a message to the agent might also be considered suspicious.

Using Contextual Knowledge

Miscreant behavior, including collusion, is more likely in some situations than others, and it presents differently in different situations as well. Consequently, the approach we take to miscreant detection is based on context-mediated behavior (CoMB), which uses a priori contextual knowledge to determine appropriate behavior in a particular situation. Here, that behavior is the process of detecting miscreants.

For now, we assume that there are one or more trusted agents in the multiagent system that are tasked with (or at least capable of) detecting miscreant behavior. Since these agents primarily use a society-level analysis of motive, we call them SLAM agents. When a SLAM agent detects such behavior, it is its responsibility to alert the rest of the MAS or in some other way take action. What action to take, while itself context-dependent, is MAS-dependent. Consequently, it is not the focus of this paper.

In our approach, a SLAM agent makes use of knowledge about the context to detect miscreant behavior. In order to do this, it needs to determine what the current context is, a process we call *context assessment*.

We make a distinction between a *situation* and a *context*. For a MAS, its situation is the sum total of all features, observable or otherwise, of itself, others, and its environment. The *observed situation* is the part of this that it has sensed or otherwise knows about, and the *observable situation* is that portion that can be observed. Situations can be grouped, with the members of each group all having the same or nearly the same implications for the MAS in terms of predictions about outcomes of actions or events, likelihood of miscreant behavior and the type of that behavior, its agents' appropriate behavior, and other inferences that can be made about it, its agents, or its environment.

These groups of situations are what we mean by *context*: A context is a class of situation with important implications for a MAS or its agents. In our domain, example contexts might be: playing Liar's Poker when there is an aggressive agent, playing when two known compatriots are playing, playing when there is a reputation system in place, and so forth. Note that a given situation may be an instance of more than one context; for example, the MAS might be in the context in which two known compatriots are playing and there is also a reputation system in place.

Our approach explicitly represents known contexts as knowledge structures called *contextual schemas* (c-schemas). Contextual schemas have a descriptive and a prescriptive part. We discuss the prescriptive part below. The descriptive knowledge describes the class of situation that the c-schema represents. It also provides predictions about possibly unseen features of the situation which can help the MAS or its agents appropriately disambiguate new information. It can also provide context-dependent meaning of concepts; in the autonomous underwater vehicle domain, for example, we have used this to define the context-dependent meaning of depth for fuzzy reasoners [Turner(1997)] and neural networks [Arritt and Turner(2003)].

Above, we discussed the features that are important in detecting miscreant behavior and collusion. These features define the space of contexts that are important, and they form the basis for the representation of the c-schemas.

Context assessment is the process of determining of which contexts a given situation is an instance. It is a diagnostic process, with features of the situation playing the role of signs and symptoms and contexts playing the role of "diseases". The descriptive knowledge contained in c-schemas is used to diagnose the situation as being an instance of one or more contexts the c-schemas represent.

Although any reasonable diagnostic technique would work, we favor the kind of abduction-based diagnosis found in the work of Feltovich [Feltovich et al.(1984)Feltovich, Johnson, Moller, and Swanson] and Miller, Pople, and Myers [Miller et al.(1982)Miller, Pople, and Myers]. Features of the situation *evoke* particular c-schemas, which are grouped into *logical competitor sets* (LCS) whose members (roughly speaking) each explain the same set of features. A differential diagnosis process then occurs to "solve" the top-ranked LCS by selecting one of its members as a diagnosis of the situation. New LCS are formed to explain any unexplained findings, and the process continues until the agent has a set of c-schemas, of each of which the current situation is an instance. The set of c-schemas is then merged to form a coherent picture of the current context.

Once the context has been assessed, the prescriptive part of the c-schemas will be used to help the agent or MAS decide how to behave. In general, we are interested using context to help agents in the MAS decide how to participate in the society as well as how to detect miscreant behavior. Our prior work in CoMB addresses the former; here, we focus on the latter.

A particular context assessment can directly serve as a hypothesis that miscreant behavior is occurring. For example, we can imagine an agent knowing about contexts in which miscreant behavior of various kinds exist. If its context assessment process determines that the current situation is an instance of one of these contexts, then the agent, with no additional effort, can directly know that the bad behavior is a real possibility. Depending on the degree of belief associated with the hypothesis, which will depend both on how strongly the agent believes its assessment and how strongly the c-schema in question predicts the offending behavior, it may or may not need to gather additional evidence to confirm the hypothesis.

Note that this is very similar to, for example, medical or other diagnostic reasoning. Determining the current context—for example, "current patient has pulmonary edema"—is essentially arriving at the diagnosis.

There are other times, however, when either no such hypothesis is forthcoming from the context assessment or when the strength of belief in a hypothesis about miscreant behavior is such that further confirmation is required. In this case, contextual knowledge can still help the agent by providing suggestions of appropriate ways to detect or confirm miscreant behavior in the current context.

For example, suppose in our toy domain that two new players arrive whom the trusted agents have never seen before. There is no a priori reason to suspect them of being miscreants, yet in this context, it would be appropriate to adopt a more wary security posture. This would be suggested by a c-schema representing the context, and when this c-schema is determined to be part of the context assessment, then that posture would automatically be adopted by the agent.

Context can also focus attention on salient features of the situation to take into account when looking for miscreant behavior. For example, noticing unusual communication is one way to detect explicit collusion. If our situation has the possibility of a covert channel, then recognizing that would help the agent detect explicit collusion. This would be the case for underwater vehicles operating together in an area, where the vehicles' sonar provides a signalling mechanism, even if acoustic communication is not in use. An agent might recall a c-schema to characterize this context that suggests listening to sonar pulses for covert communication between agents. Although this could have been arrived at via reasoning from first principles, by having it be mediated via context instead, we automatically get the suggestion as a by-product of context assessment, and we do not have to reason about this particular covert channel at all in other situations.

Reasoning types and the knowledge for them can also be affected by context, and so c-schemas can provide information to the agent about how best to behave and about knowledge useful for behavior. In our current work on SLAM, we intend to build a dynamic decision network (DDN) for each other player in the game, adjusting its belief about their attitudes each time an action is observed. Instead of building these structures from scratch when a new agent enters the system, we can store parameters of the networks in c-schemas representing contexts involving the agents. Then, when the agent is seen again, the c-schemas will be retrieved and will allow the networks to be immediately re-instantiated. This can also be done for classes of agents, so that if we see an agent we have never seen before, but that we know something about (e.g., who constructed it, its reasoning style, etc.), we can start with an appropriate DDN that is built for agents of its type.

It is important that an agent always maintains an accurate view of what the context is, even as the situation changes. This can be done in several ways. First, as we have done in the past (e.g., [Turner(1998)]), the agent can re-assess its context on a regular basis. When the assessment produces a new set of c-schemas, then the new merged c-schemas become the new context representation, and behavior changes automatically.

Another possibility is to identify for the domain some set of context change-inducing event (CCIE) that the agent should look for. Only when these are detected would context assessment be done, thus saving computational effort. Some examples of CCIEs we have so far identified include: detection of collusion; change in composition of the MAS (due to, e.g., an agent entering or exiting); detection of unusual message traffic, either within legal communication channels or via a covert channel; or, for some kinds of domains, temporal characteristics (e.g., whether we are early in a problem-solving session or game as opposed to later).

Finally, c-schemas can themselves suggest when the context should change, by providing features or events either that indicate the c-schema is no longer a good fit for the situation or that directly suggest other c-schemas that may capture the situation's context.

We will examine these mechanisms to see which work the best in our toy domain as well as in general for different kinds of MAS and domains.

So far, we have assumed that there are one or more trusted agents in the MAS whose duty it is to look for miscreant behavior, much like police do in the real world. Given this assumption, there are two issues. First, how can the work be distributed across all trusted agents? Does it make sense, for example, for each of the agents to work separately, or should the agents cooperate to reason about the situation, including sharing information about what the context is and whether or not someone is misbehaving? Second, if the watchers are themselves self-interested agents, then there is the possibility for their goals to lead them to behave as miscreants; who watches the watchers? Both of these questions will be the subject of work in this project.

We will also consider the question of whether we really need trusted agents in the MAS at all. In some ways, this seems like an unreasonable restriction on many real-world MAS, e.g., client-server systems such as the Web in which there really is no central authority to grant this trusted status. We will consider the situation in which there are no privileged agents, but rather in which some agents are simply capable, via context-based SLAM, of detecting miscreant behavior. In this case, issues to be examined include how many agents are needed to be effective, what to do should they disagree, the level of cooperation between such agents, and how the agents can take action, e.g., by attempting to raise the alarm about miscreant behavior or to persuade others to take action.

We have not yet addressed the question of where c-schemas come from in the first place. CoMB itself has the position that c-schemas are actually generalized cases (in the sense of case-based reasoning [Kolodner(1993)]) of problem solving, and that they are learned from experience via similarity-based or other kinds of learning. Learning is not the focus of this work, however, and we anticipate, initially at least, hand-crafting c-schemas for our agents to use. Future work will look at how c-schemas can be learned or modified from experience.

Conclusion and Future Work

Real-world multiagent systems are often neither completely cooperative nor completely competitive, but rather are societies in which there are both individual as well as society-level goals. While agents can generally be expected to behave appropriately, from the society's standpoint, the goal-directed behavior of the individuals comprising the society can often lead them to be miscreants, either intentionally or unintentionally. A MAS needs the ability to detect such miscreant behavior so that its goals are not compromised.

In this paper, we have described a context-based approach to the problem of miscreant behavior detection that is being developed, in which known contexts are explicitly represented and reasoned about, and in which the context representations (c-schemas) contain context-appropriate knowledge the agents can use to guide their decision making. We believe that this approach will allow agents to more easily detect such behavior by using knowledge that is appropriate to the current context.

Although work on context-mediated behavior, on which this work is based, is somewhat mature, the work reported in this paper is still in the early stages and is the subject of a PhD dissertation that is in progress. The short term will see a more complete design and implementation of the context-based SLAM approach described.

In the longer term, we anticipate this approach being the basis for more general context-aware multiagent systems. In such systems, context would itself be a first-class object for discussion and reasoning about by the agents, and context assessment would be a shared activity among at least a subset of the agents. Such systems would be able to adjust their behavior and possibly their form to fit their evolving context.

Also in the longer term, we will focus attention on the question of how to learn and modify c-schemas based on experience. In the case of a MAS, this, too, may be a task shared among the MAS' context-aware agents.

References

Robert P. Arritt and Roy M. Turner. Context-specific weights for a neural network. In *Proceedings of the Fourth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'03)*, pages 29–39, Stanford, CA, 2003. Springer, New York.

R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.

A. Birk. Learning to trust. In R. Falcone, M. Singh, and Y. Tan, editors, *Deception, Fraud and Trust in Agent Societies*, pages 133–144. Springer, 2001.

C. Castelfranchi and Y. Tan. Why trust and deception are essential for virtual societies. In C. Castelfranchi and Y. Tan, editors, *Trust and Deception in Virtual Societies.* Kluwer Academic Publishers, 2001.

L.N. de Castro and F.J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, 6(3):239–251, 2002.

Ed H. Durfee. Distributed problem solving and planning. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA, 1999.

R. Falcone and C. Castelfranchi. Social trust: A cognitive approach. In C. Castelfranchi and Y. Tan, editors, *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer Academic Publishers, 2001.

P. J. Feltovich, P. E. Johnson, J. A. Moller, and D. B. Swanson. LCS: The role and development of medical knowledge and diagnostic expertise. In W. J. Clancey and E. H. Shortliffe, editors, *Readings in Medical Artificial Intelligence*, pages 275–319. Addison–Wesley Publishing Company, Reading, Massachusetts, 1984.

D. Gambetta. Can we trust trust? In D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, pages 213–238. Basil Blackwell, Oxford, 1990.

J.K. Goeree and C.A. Hold. Stochastic game theory: For playing games, not just for doing theory. In *PNAS*, volume 96, pages 10564–10567, 1999.

K. HOffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. In *ACM Computing Surveys*, volume 42, pages 1–31, 2009.

L.R. Iszuierdo and S.S. Izquierdo. Dynamics of the Bush–Mosteller learning algorithm in 2x2 games. In C. Weber, M. Elshaw, and N.M. Mayer, editors, *Reinforcement Learning: Theory and Applications*, page 424. I-Tech Education and Publishing, Vienna, 2008.

A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

Janet L. Kolodner. *Case-Based Reasoning.* Morgan Kaufman, San Mateo, CA, 1993.

M.W. Macy and A. Flache. Learning dynamics in social dilemmas. In *PNAS*, volume 99, pages 7229–7236, 2002.

R. A. Miller, H. E. Pople, and J. D. Myers. INTERNIST–1, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476, 1982.

I.S. Moskowitz and M.H. Kang. Covert channels-here to stay? In *Computer Assurance, 1994. COMPASS '94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security. Proceedings of the Ninth Annual Conference on*, pages 235 –243, jun-1 jul 1994. doi: 10.1109/CMPASS.1994.318449.

Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, AAMAS '02, pages 280–287, New York, NY, USA, 2002. ACM. ISBN 1-58113-480-0. doi: http://doi.acm.org/10.1145/544741.544807. URL `http://doi.acm.org/10.1145/544741.544807`.

Animesh Nandi, Tsuen-Wan Ngan, Atul Singh, Peter Druschel, and Dan Wallach. Scrivener: Providing incentives in cooperative content distribution systems. In Gustavo Alonso, editor, *Middleware 2005*, volume 3790 of *Lecture Notes in Computer Science*, pages 270–291. Springer Berlin / Heidelberg, 2005.

S.D. Ramchurn and N.R. Jennings. Trust in agent-based software. In R. Mansell and B.S. Collins, editors, *Trust and Crime in Information Societies.* Edward Elgar Publishing LImited, Cheltenham, UK, 2005.

T.W. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA, 1999.

Roy M. Turner. *Adaptive Reasoning for Real-World Problems: A Schema-Based Approach.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

Roy M. Turner. Determining the context-dependent meaning of fuzzy subsets. In *Proceedings of the 1997 International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97)*, Rio de Janeiro, 1997.

Roy M. Turner. Context-mediated behavior for intelligent agents. *International Journal of Human–Computer Studies*, 48(3):307–330, March 1998.

Larry T. Whitsel. A simulator for rule-based agent trust decisions. Master's thesis, University of Maine, Department of Computer Science, Orono, Maine, 2010.