I'm OK, You're OK, We're OK: Experiments in Distributed and Centralized Socially Attentive Monitoring

Gal Kaminka and Milind Tambe
Information Sciences Institute and Computer Science Department
University of Southern California
Los Angeles, CA 90292
{galk, tambe}@isi.edu

ABSTRACT

Execution monitoring is a critical challenge for agents in dynamic, complex, multi-agent domains. Existing approaches utilize goal-attentive models which monitor achievement of task goals. However, they lack knowledge of the intended relationships which should hold among the agents, and so fail to address key opportunities and difficulties in multi-agent settings. We explore SAM, a novel complementary framework for social monitoring that utilizes knowledge of social relationships among agents in monitoring them. We compare the performance of SAM when monitoring is done by a single agent in a centralized fashion, versus team monitoring in a distributed fashion. We experiment with several SAM instantiations, algorithms that are sound and incomplete, unsound and complete, and both sound and complete. While a more complex algorithm appears useful in the centralized case (but is unsound), the surprising result is that a much simpler algorithm in the distributed case is both sound and complete. We present a set of techniques for practical, efficient implementations with rigorously proven performance guarantees, and systematic empirical validation.

Keywords

Monitoring, Diagnosis, Social Agents, Coordination, Teamwork

1. INTRODUCTION

Execution monitoring [2], [8] is a critical challenge for applications of autonomous agents in training and instruction [12], command-and-control operations [14], coordination ([3], [13]), UAVs for traffic surveillance, etc. In these real-world applications, a monitoring agent monitors the behavior of an individual agent or a team (which may include itself), detecting and diagnosing failures as they occur. Interactions among multiple agents complicate the task of the monitor, since it now has to consider interactions not only between the agents and the environment, but also among the agents themselves. Also, failures in interaction between an agent and the environment (e.g., failed sensors) affect its interactions with other agents.

Previous investigations of execution monitoring (e.g., [1], [2], [8], [14]) have focused on a single agent that utilizes task-specific condition monitors, or a model of the monitored system, to generate *goal-attentive,* or teleological, expectations of correct execution, i.e., testing achievement of tasks and plan-steps. This approach is very powerful, but fails to treat key opportunities and problems in multi-agent settings.

First, the monitored target of these systems is usually a single agent/system. When applied to monitoring multiple agents, the failures they must detect may be distributed, and may not be detected when monitoring an individual agent *per se*, e.g., the failure may be that an individual agent did not help a teammate in need, or that a team of agents is not in agreement on the goals of the team. Second, a lack of knowledge of the relationship with other agents prevents them from using other agents' behavior as a knowledge source for correcting the execution. For example, a driver may not see a road-sign that tells it to turn. But if she is driving in a convoy, where everybody shares the goal, she can infer the existence of the road-sign when everybody else turns, or at least look more carefully for it. Third, the focus of these approaches on *goal-attentive* models of the monitored system does not allow them to consider the relations that must be maintained independently of the achievement of the goal. This may cause recovery to be done in a way that violates important relationships--Sacrificing teammates to achieve an individual goal is hardly acceptable behavior.

This paper explores Socially Attentive Monitoring (SAM), a *social* execution monitoring framework, complementary to existing approaches. The key idea is to use various models of social *relationships* among agents, rather than *goal-attentive* models of the task, to drive the monitoring process. Our hypothesis is that judging whether two agents are maintaining a relationship (social monitoring) is often much easier than determining whether they are acting correctly with respect to some goal (goal-attentive monitoring). The behavior of a convoy can be monitored by verifying in detail the goal of each agent individually. However it is easier to detect a failure by noting that the agents drive in different directions.

SAM has knowledge of different types of social relationships, which are used to drive the monitoring process: teamwork relationships, coordination relationships, similarity relationships, etc. An agent running SAM uses communications, plan-recognition, etc. to gain knowledge of monitored agents' state. Then the relationship models are used to verify that specified relationships among agents are not violated. For instance, by verifying the relative velocity and position of two agents, SAM

can judge whether they are maintaining formation (a type of coordination relationship). Once a failure is detected, SAM diagnoses the failure, attempting to explain the violation of the relationships. The relationship models focus the diagnosis process on relevant explanations, and greatly reduce the computational effort. The resulting explanation paves the way to recovery via negotiations, commands, etc.

Previous work [6] presented SAM in an initial form, adopting an approach of a single social monitor observing its teammates, without systematic experiments exploring the degrees of freedom in SAM's design, or any formalization of the method. This paper presents a significant advance. It presents an experiments-driven exploration of SAM's parameters—centralized/single monitoring agent vs. a distributed configuration, and the effects of explicit representation of ambiguity in plan-recognition. The repeating theme in all of these explorations is the desire to find simple, cheap, and powerful techniques that agents can employ in practice, with guaranteed results. In particular, we will show that a simple distributed scheme which requires no explicit representation of ambiguity, nor communications, performs better (sound and complete detection) than a more complex scheme that is used by a single monitoring agent (complete and unsound, requiring communications and explicit representation of ambiguity). SAM's dimensions will be mostly illustrated using collaboration relationships, but other relationship models are also implemented (coordination, role similarity, etc.).

2. MOTIVATION AND EXAMPLES

SAM was born out of frustration. During several years of intensive development, and despite rigorous testing, our automated pilot agents in a realistic battlefield training simulation [12] displayed an annoying tendency to fail at the most critical moments. The environment's many uncertainties, such as behaviors of other agents, unreliable communications and sensors, etc., presented the agents with never-ending opportunities for failure, which we could simply not anticipate. Our teamwork model [11] prevents many failures, but it is not sufficient. SAM can extend the model using plan-recognition.

Some examples may serve to illustrate. The first example (henceforth, **example 1**) involves a scenario where a team of three helicopter pilot agents were to fly to a specified waypoint (a given position), where one of the team-members, the *scout*, was to fly forward towards the enemy, while its teammates (*attackers*) land and wait for its signal to join it. During flight, all of the agents monitored for the waypoint. However, due to an unexpected sensor failure, one of the attackers failed to sense the waypoint. So while the other attacker correctly landed, the failing attacker continued to fly forward with the scout.

In a different run (**example 2**), after all three agents reached the waypoint and detected it, the scout has gone forward and identified the enemy. It then sent a message to the waiting attackers to join it and attack the enemy. One of the attackers did not receive the message, and so remained behind indefinitely while the scout and correct attacker continued the mission alone.

We have collected dozens of similar reports during the last three years. While the failures are obvious to the human designer once they occur, they are very hard to anticipate in design time. These failures are almost always catastrophic, as the agents are often unable to continue correct execution; yet the failure are not due to lack of domain expertise, as *agents could recover if they monitored relationships with other agents*. The goal-attentive condition monitors are insufficient, as individual agents are behaving correctly given their task-specific input.

3. SOCIAL DIAGNOSIS & MONITORING

The general structure of SAM (Socially Attentive Monitoring) is shown in Figure 1. It consists of (1) a knowledge-base containing models of relationships that should hold among the monitored agents; (2) the agent modeling component responsible for collecting and representing knowledge about the monitored agents; (3) the *detector* that monitors for violations of relationships among monitored agents; and (4) the *diagnoser* that verifies the failure, and provides an explanation for it.

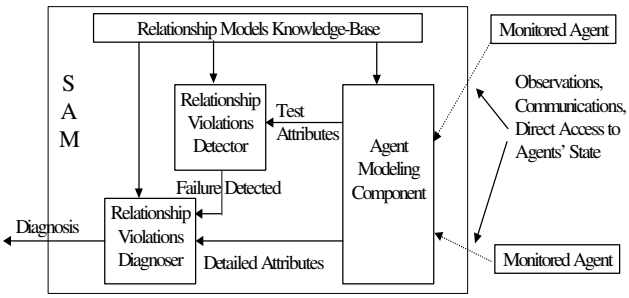


Figure 1. General Structure of a SAM System.

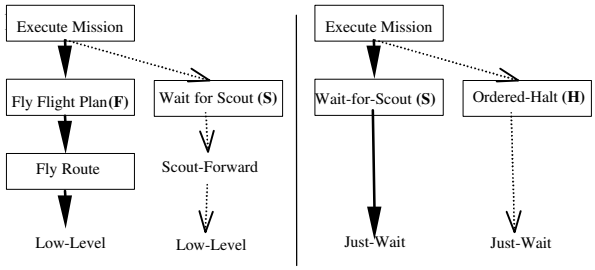
3.1 Agent Modeling and Representation

When an agent monitors itself, it may have direct access to its own internal state information. When monitoring others, however, this is often not possible. Even if monitored agents cooperate, they cannot, in complex domains, continuously communicate their internal state to the monitor, as it is intrusive and requires communications to be safe, cheap, fast, reliable, etc. (which is often not possible). Instead, a monitor may choose to use plan-recognition to infer the agents' unobservable state from their observable behavior. This approach has greater computational cost and reduced certainty, but is unintrusive, and robust in face of communication failures (the monitor may still benefit from focused communications with the other agents).

To enable such plan-recognition, we have developed a new algorithm called RESL (REal-time Situated Least-commitments). The key idea is to maintain all reactive-plan (henceforth, plan) hypotheses matching each agent's observed behavior. The entire plan library hierarchy is expanded for each modeled agent, and all paths matching the observed behavior of the agent being modeled are tagged.

Figure 2 gives a simplified presentation of the plan hierarchies for a variation of example 1, in which the two attackers (Figure 2-b) detected the waypoint and have landed (switching to the wait-for-scout plan, henceforth denoted by **S**), while the scout (Figure 2-a) did not and is flying forward (executing **F**). The filled arrows indicate the actual executing plans. Dotted arrows mark additional plans that from an external observer's perspective match the observable behavior. Team-plans (see below) are boxed. For instance, when the observed scout's speed and altitude match "low-level", one of the possible flight-methods under both the fly-flight-plan (**F**) and Wait-for-Scout (**S**) plan, they are both flagged as matching. This match is then propagated up. Similarly, when the attackers land, RESL recognizes that they are executing the

"Just-Wait" plan. However, this plan can be used in service of either the Wait-for-Scout plan (**S**), or the Ordered-Halt (**H**) plan -- a plan in which helicopters are ordered by their headquarters to land immediately. Therefore, both **H** and **S** are tagged as matching.



(a) Scout's Plan Hierarchy (b) Attackers' Plan Hierarchy

Figure 2. Scout (a) and Attackers' (b) actual and recognized *abbreviated* reactive plan hierarchies. Both attackers use (b).

3.2 Social Failure Detection

The detector signals possible relationship violations (serving as an initial trigger for the diagnoser, which verifies the failures). We illustrate with an example of teamwork relationships. Recently, domain-independent teamwork models that provide guidelines on team-members behavior and responsibilities have been introduced [7], [11], [5], to provide teamwork flexibility. STEAM (and similarly, other teamwork models) dictate that team plans (boxed in Figure 2) must be identical for all team-members. The detector thus performs the teamwork test by comparing the agents' plans at each level of the hierarchy. The topmost difference found (if any) signifies a difference in the goals of the agents, and therefore the breakdown of teamwork. For instance, in Figure 2, the scout differs from the attackers at the second level.

The detector may be correct in detecting relationship violations despite being wrong about the details. For instance, when the flying attacker notices its teammate has landed, it doesn't know which plan the other attacker has switched to. But it can be sure that the other attacker is no longer executing the flight plan jointly with itself. In terms of social failure detection, then, negatives and positives are defined *in terms of whether a relationship violation has occurred*, not in terms of the whether the agent modeling hypotheses correctly match the observed agents actual plan, nor in terms of whether a goal-attentive failure has occurred. Negatives are defined as the detector not detecting a violation, and positives are defined as where it does. They could be true or false, depending on whether a violation has actually occurred. Following this, completeness and soundness of social failure detection are defined as not having any false-negatives and false-positives, respectively.

The relationship tested determines the test used in SAM. Coordination relationships, for instance, may require plans to be *unequal* at all times: When engaging the enemy, pilots are sometimes supposed to pop-up from behind the hills in sequence, such that no two helicopters appear at the same time. Here, a discovery that two agents *are* executing the same plan at the same time is a sign of failure (exactly the opposite from collaboration). In formation relationships, all the helicopters' velocities are defined in terms of the leader's velocity. Failures here would be expressed in failing to maintain the *relative* velocities within nominal ranges.

3.3 Social Diagnosis

The diagnoser verifies the detected violation and constructs an explanation for it. Although the details may vary depending on the type of relationship, the diagnosis is often given in terms of a (minimal) set of belief differences between the agents that can account for the failure to maintain the relationship. These are used as a basis for recovery. The starting point for this process is the detected failure (e.g., the exact difference in executing team-plans), and the monitor's matching hypothesis(es) for what plans the monitored agents are currently executing. The diagnoser compares the beliefs of the agents to produce a set of differing beliefs that account for differences in the plans. However, each agent in a real-world domain may have many beliefs, and many of them will vary among the agents, though most of them will be irrelevant to the diagnosis. Moreover, the monitor is not likely to have access to all of the beliefs held by the agents.

Knowledge of the relationships between the agents is useful here, since they specify how the beliefs, goals, and actions of the different agents are related. For example, the teamwork model dictates which beliefs the agents hold *must be* mutually believed by all the agents in the team [11]. Any difference that is detected in those beliefs is a certain failure, as the team members do not agree (because of a failure) on issues on which agreement is mandatory to participation in the team. In particular, team plans which mandate joint execution by the team members are jointly selected (or terminated) by team members' establishing mutual belief in at least some of their preconditions (or termination conditions). Differences in plan selection are therefore a certain failure. Differences in conditions that are to be mutually believed also signify certain failures.

Upon detection of the difference in team-plans (henceforth, plans) among the agents, the diagnoser requests information from the agent-modeling component about the beliefs of the agents involved. In general, the diagnoser can infer only that each agent believes that the termination conditions for its associated plans have not been satisfied, and that the preconditions for those plans just selected have become true. Often, however, the inference can go even further. In domains in which changes in plans are always observable, SAM will detect the difference in plans as soon as it happens. Then, the diagnoser notes the last operator the now-differing agents have been known to agree on, and their currently executing plans. It then classifies each agent as: (a) still executing the last-agreed-upon plan (i.e., never made a switch); or, (b) having switched from the last agreed upon plan to a different one. In case (a), the agent in question has never made the switch, and so believes that the termination conditions of the last-agreed-upon plan have not been achieved. If an agent has made a switch (case b), then it believes that the previous plan's termination conditions were achieved, as were the preconditions for its next plan. Let O_p be the last agreed upon plan, with a disjunctive list T of termination conditions T_i, each a conjunction of (possibly negated) atomic terms. Let O_n be the plan currently executed, with a disjunctive list P of preconditions P_j (again, each a conjunction). O_p could have terminated because of one or more T_i have become true, and O_n could have been selected because one or more P_j have become true.

The diagnoser thus potentially faces numerous hypotheses. The agent may hold true any one of the following sets of belief: $T_1 \wedge P_1$, $T_1 \wedge P_2$, …. $(T_1 \wedge T_2) \wedge (P_1 \wedge P_2)$, etc. The number of possible

hypotheses is $2^{|T|} \times 2^{|P|}$, the cardinality of the cross-product of the two power-sets. Fortunately, the situation is not so bad in practice. First, inconsistent hypotheses can be eliminated. (Suppose for example that $T_1 = X \wedge Y \wedge \neg Z$ and $P_2 = Q \wedge Y \wedge Z$. Any hypothesis that contains $T_1 \wedge P_2$ would be inconsistent). Second, the diagnoser may choose to heuristically rank the hypotheses. SAM uses the minimal cardinality heuristic.

Now that it has access to the beliefs of each agent involved, the diagnoser looks to the relationship model (teamwork, in this case) to provide knowledge of how beliefs should have been related (here, mutually believed or identical). Beliefs that do not maintain the relation (here, not mutually believed) form the diagnosis set, i.e., they provide an explanation for why the agents selected different plans when they should have selected identical plans.

4. CENTRALIZED MONITORING

Our methodology is that of systematic experimental investigation of the dimensions of SAM. We seek to find practical techniques which maximize monitoring capabilities but minimize computational and communications costs. We explore SAM by presenting empirical results from systematic variations of examples 1 and 2, in which we explore all combinations of agent failures (i.e., a single agent failing, two agents, etc.), and all monitor roles in the team (attacker/scout). We focus on examples using collaboration relationship monitoring.

The simplest case of execution monitoring is where the monitoring is done by a single agent (who may be part of the monitored group). We experiment here with the effects of explicit representation of ambiguity in SAM's plan recognition. We begin by examining a SAM variation which limits computational complexity by causing the agent modeling component to select a single plan-recognition hypothesis for the detection process.

We first experiment with a novel disambiguation method that ranks plan-recognition hypotheses by the level of team-coherence they represent (the number of agents in agreement in each of them). The more coherent an hypothesis, the higher its rank, e.g., any hypothesis which has all team-members executing identical team-plans will be ranked highest (further disambiguation may use other criteria, such as minimal assumptions). The intuition is that failures to maintain relationships are occurring *despite* the agents' attempts. Thus often major parts of the team will remain coherent.

In **example 2**, no matter which one of the agents is the monitor, the failure to maintain relationship will be detected regardless of ambiguity in the plan-recognition. The first set of experimental results are shown in Table 1 (see end of paper), presenting the results from the scout's point of view of all systematic failure variations of this scenario. Here, J marks the team plan where the scout waits for the attackers to join it after identifying the enemy and sending a message to the attackers; S marks the wait-for-scout plan, selected when the waypoint is originally detected; and H marks the ordered-halt plan, selected when a command is received from HQ to halt execution. These are the topmost team plans at which a difference in team-plans among agents will occur. The failures are on the part of the attackers (to receive the message) or on the part of the scout (to send it). There are five variations possible (a failure to send is the same as both attackers not receiving). The first three columns show the actual plans selected by the three agents: A1, A2, and A3. The next three columns show A3's selected modeling hypothesis using team-coherence. These show what A3 believes the team is doing as a whole. The next two columns show in which of the cases a social failure has actually occurred, and whether it was in fact detected by A3. The next column summarizes the detection class: True positive, true negative, etc. Finally, the last column explains the physical failure that actually took place. For example, row 4 presents the case where both attackers did not receive the message (physical failure: A3's message was lost). They have landed and are waiting for the scout (S). The scout, however, believes the message was received, and so it lands, waiting for the attackers to join it (J). Here the scout (acting as monitor) notices the two attackers are not moving forward. So it cannot be the case that they are joining it (executing J). Under the team-coherence criteria, they are both executing either S (i.e., they never received the message), or H (i.e., the scout failed to received the message to halt execution). But in any case, the collaboration relationships have been violated. This social failure is detected and is therefore a true-positive.

Many failures are successfully captured by SAM with the team-coherence heuristic (for example, all teamwork failures in Table 1). However, as we continued to experiment, we discovered cases in which it doesn't work as well. Table 2 presents the results of the centralized social monitoring in a set of experiments based on example 1 (**Figure 2**). These present the attacker A1's non-ambiguous view of the team under all possible permutations of the failure to see the waypoint, using the team-coherence criteria. The plan Fly-flight-Plan is marked F.

The two highlighted rows point out A1's failures to recognize that a social failure is taking place. The team coherence criteria has caused A1 in both of these cases to believe the scout is in fact maintaining its relationship with the rest of the team, i.e., the scout (A3) is executing the same team-plan as A1 and A2. A false negative has occurred in both of these cases. The ordered-halt plan (see Figure 2) have been ruled out in all rows as hypotheses in which it appeared were less team-coherent than those appearing in the table.

In none of our experiments did the team-coherence criteria produce a false-positive result, i.e. a social failure is detected when none occurred. Indeed, we were able to verify these capabilities formally:

Theorem 1: *The team-coherence heuristic will not lead to a false positive detection.*

Proof sketch. An agent is observing a group of agents $a_0, \ldots, a_n$. (which may include itself). For each monitored agent there is an individual matching hypotheses set H_i. If all the agents are executing the same plan P, then for each $P \in H_i$. Thus (P, …. , P) will be at least one of the group hypotheses produced by the monitoring agent. This hypothesis implies maximal team coherence and so would be ranked at the top. If it is selected, the detector will give a negative result (i.e., no false positive). If it isn't selected, the selected one would have equal coherence ranking (have all agents executing some other plan Q). Resulting again in a negative detection. QED.

The team-coherence heuristic is thus *guaranteed* to cause no false positives in the detector, and is therefore sound – any failure it detects is indeed a true failure in maintaining relationships. However, the detection may be incomplete (i.e., contain false negatives) as we have seen.

Of course, we would have preferred an algorithm that is complete–that is guaranteed to detect all social failures. We therefore experimented with an alternative disambiguation criteria, *team-incoherence* (i.e., the opposite of the team-coherence criteria). Here, the modeling component prefers hypotheses allowing for the possibility of a disagreement between the agents. Table 3 gives the Attacker's view, similarly to Table 2, but using the team-incoherence criteria.

This heuristic guarantees that no false negatives would occur (the proof follows that of theorem 1). It would thus cause the detector be complete. However, completeness comes at a cost of false positives (highlighted rows above), and thus detection is unsound. If in practice most of the time the systems are performing nominally, many useless false alarms would be generated.

The two inverse criteria – team-coherence and incoherence – represent two extremes of a space of disambiguation methods for social monitoring. The team-coherence criteria guarantees soundness of social failure detection, at the price of completeness. The team-*in*coherence criteria achieves completeness at the cost of soundness. Moreover, without having additional information, we can show that no disambiguation scheme exists that is both sound and complete, and still consistent—given the same set of possible matching hypotheses, it will have to sometimes choose one hypothesis and sometimes another in order to be both sound and complete.

Definition 1: Let M(A1/P1) denote the set of all hypotheses matching a monitor's observations of agent A1 when it executes team-plan P1. We say that A1's role in P1 is *observably different* than in team-plan P2 if $M(A1/P1) \cap M(A1/P2) = \emptyset$; i.e., the monitor can determine the team-plan selected.

Theorem 2: *Any disambiguation scheme D that does not utilize external knowledge and is both sound and complete is not consistent in its selection of a disambiguated hypothesis.*

Proof sketch: Let D be the candidate disambiguation scheme that is complete and sound. Assume that it is deterministic. Let A1,A2 be agents whose role in P1, P2 is *not* observably different: M(A1/P1)=M(A1/P2)=M(A2/P1)=M(A2/P2)={P1, P2}. Let B be a monitoring agent who observes A1, A2 and uses D for disambiguation. The set of matching hypotheses when A1 and A2 are both executing P1 is H={(P1,P1), (P2,P1), (P1,P2), (P2,P2)}. And since D is both complete and correct, B (using D) will choose (P1,P1). Now, when A1 and A2 are executing P1and P2, respectively., the matching hypothesis set is again H. But now B (using D again) must select (P1,P2). Since the same set of options was available in each case, and no other information was supplied, D must be inconsistent in its selection. QED.

It would thus seem that our exploration of a centralized monitor using a single, unambiguous, hypothesis, leads to discouraging results. We therefore let go of our insistence on a single hypothesis, and examine the effects of explicit representation of ambiguity on the centralized monitoring process. We now allow examining all individual matching hypotheses as maintained by RESL. Table 4 shows again the attacker A1's view of the team, but this time with all hypotheses explicitly represented (Note that RESL actually represents only individual hypothesis, so the group combinations are actually implicit in our implementation, and presented here for clarity). Due to the exponential size, we present the results of only the first few variations.

The cost of this representation is clear just from a first look at the table: the detector and diagnoser potentially have an exponential (in the size of the team) number of hypotheses to reason about. The question is whether this enabling of multiple hypotheses representation alleviates our concerns as to lack of soundness or completeness.

The detection can be qualified simply by examining a candidate team-incoherent hypothesis and a candidate team-coherent hypothesis. These, in essence, bound the matching hypotheses between them with respect to failure detection. If they agree that a failure exists, then the detector can be sure of a true positive. If they disagree (i.e., the team-coherent hypothesis produces negative detection), the detector cannot be certain, and must revert back to verification. Though detection is still unsound, this process reduces the number of cases needing verification (50% in tables 2-3). It also requires representing only two candidate hypotheses–one team-coherent hypothesis, and one team-incoherent hypothesis.

Representing multiple hypotheses explicitly is also necessary for centralized diagnosis. The diagnosis procedure as described (section 3.3) implicitly assumes the underlying detection is sound, and the models of the diagnosed agents are correct. But it may be possible that the diagnosis will take place using an incorrect description (plan recognition hypothesis) of the agents. These cases can be categorized as follows:

1. The detection is correct, but the models of the agents are incorrect. For example, row 4 of Table 1 presents a case of correct detection using incorrect models.
2. The detection is incorrect (a false positive). The entire diagnosis is irrelevant.
3. The detection is correct, but some of the agents are falsely assumed to be executing a non-differing plan. For instance, in row 2 of Table 2.

Explicit representation of all matching hypotheses means that the correct hypothesis, even when not selected, is still available for the diagnoser. It is therefore able to assess the state of its knowledge, and attempt verification when appropriate.

We have explored SAM in this section in centralized configuration, with and without explicit representation of ambiguity, and under several different disambiguation schemes and combinations. The emerging technique for centralized social monitoring is complete, but unsound, and is quite complex and expensive, requiring representation of multiple hypotheses and communications for verification.

5. DISTRIBUTED MONITORING

The next configuration of SAM we explore is distributed monitoring, where all participating agents monitor themselves and the team using SAM. We begin exploration again with the simple scheme of team-coherence.

The results for the variations of **example 1** are in Table 2 for the attackers, Table 5 for the scout. When we examine them more closely, we find that the two false-negatives of (Table 2. , rows 4-5), occur when the attackers monitor the team, but not when the scout does. In fact, as Table 5 shows, the scout correctly detects all cases where a social failure would occur, and so it can

compensate for the attackers' false-negative detection (though it may select the wrong hypothesis, H, to explain their actions, the detection is still correct).

The surprising result is that the team, relying on the scout to identify failures, can in fact *completely* detect all failures using simple team-coherence, without the need for the complex algorithm deployed by the centralized monitor.

We attempt to formally define the circumstances under which this phenomenon holds. Under the team-coherence criteria, a failure is detected between agent A1 (running P1) and agent A2 (running P2) if $M(A1/P1) \cap M(A2/P2) = \emptyset$. First we prove a lemma on the conditions in which a single agent will detect a difference. We then use this lemma to prove the conditions under which a self-monitoring team would detect a failure.

Lemma 1: *A monitoring agent who is part of the team and is executing P1 would detect a difference with an agent A2 executing a different plan P2, if A2 has an observably different role in P1 and P2.*

Proof sketch: Since $M(A2/P1) \cap M(A2/P2) = \emptyset$, therefore $P1 \notin M(A2/P2)$. QED.

Before proving the main theorem, let us define the following property:

Definition 2: A agent team T executing a set of team-plans P is said to be *observably-partitioned* with respect to P if for any two plans P_i, P_j there exists ***an*** agent in the team whose role is observably different in P_i and P_j.

For instance, our helicopter pilots team is observably-partitioned with respect to the plans fly-flight-plan, wait-at-point, and ordered-halt. The attackers' behavior is different in the first two plans. The scout's behavior is different in the second and third, and the attackers' is different in the first and third.

Theorem 3: *If an agent team T is observably-partitioned with respect to a set of team-plans P, and all agents are using the team-coherence heuristic, the social failure detection is sound and complete.*

Proof sketch: First, since no individual member in the team would report a false positive, it follows that no false positive detection would be reported in general. Thus this distributed monitoring scheme is sound. We will prove at least one agent will detect a difference between itself and others whenever team-members are not all executing the same plan (i.e., a failure is occurring). Let team-plans P_i, $P_j \in P$ be two of the plans currently executed by team-members (there are at least two, otherwise there is no failure). For P_i and P_j there exists at least one agent a_1 whose role is observably different in P_i and P_j. There are three cases: (i) a_1 is executing P_i. In this case any agent executing P_j would detect failure (lemma 1); (ii) a_1 is executing P_j. In this case any agent executing P_i would detect failure (lemma 1); (iii) a_1 is executing some other plan Q. Its role in Q must observably different than in P_i or in P_j (or both), and again any agent executing P_j /P_i would detect the failure. (a_1's role in Q cannot be observably non-different in both P_i and P_j, since then $M(a_1/P_i) \cap M(a_1/P_j) = Q$, which contradicts a_1 being observably-different in P_i and P_j). Based on (i)-(iii), distributed monitoring is complete. QED.

Thus if we could guarantee at design-time that the team is observably-partitioned w.r.t the plans executed, we would be guaranteed sound and complete distributed social failure detection. This may not be a difficult property to design: Teams are very often composed such that not all agents have the same role in the same plan. Monitoring in a team where this property holds can be achieved by a very simple protocol: Use the team-coherence heuristic to choose a most coherent hypothesis. If it contains a failure, go ahead with diagnosis and/or let the other agents know. If it doesn't, go ahead with the execution of the individual task and continue monitoring.

If the team, however, is not observably-partitioned, there may be a case where any two agents are each executing a different plan, but both will have a false negative detection, and therefore overall the distributed monitoring will have a false negative. Fortunately, this problem may be alleviated. It occurs when A1 (executing P) has $P \in M(A2/Q)$, and A2 (executing Q) has $Q \in M(A1/P)$. A check for this condition can be made a part of the plan design process, marking *risky points* in the execution in which verification by communications can be prescribed proactively. Or, the check could be inserted into the protocol for run-time analysis—the agent would simulate the other's hypotheses matching their own actions, and detect risky points dynamically.

Moving from centralized detection to distributed detection has both simplified and improved the results of the detection, due to shared responsibility. Similarly, the distributed diagnosis problem can potentially made easier because other agents in the team are participating in the monitoring. In specific circumstances, for instance, an agent might be able to not only detect a failure in its own behavior, but also to diagnose it and recovery by quietly adopting its team-mates plan, without the other team members even knowing that a problem exists. However, here the situation is more complex. In general, adoption of others' team-plans may lead to dead-locks, looping behavior (A adopts B's plan, B adopts A's), etc. These issues are subject for future work.

6. RELATED WORK

In the arena of distributed monitoring, the idea of comparing beliefs has earlier appeared in [10] and [9]. Sugawara and Lesser [10] have used comparative analysis [4] in the context of a group of diagnosis agents that coordinate in diagnosing a network. Their work focuses on learning situation-specific coordination rules for each of the agents, to optimize their coordination strategy. This involves the construction of a global system view by the agents involved (via communications), and comparing this view to the agents' own local views to detect errors in local view to be corrected by learning. Schroeder and Wagner [9] have proposed a scheme for distributed diagnosis by cooperating agents who receive requests for tests and diagnoses, and send responses to other agents. The agents each construct a global diagnosis based on the local ones they produce and receive. They do not consider communication costs, uncertainties, nor conflicts in the construction of the global view, and in both investigations, cooperation is assumed. In contrast, SAM uses plan-recognition and minimizes communications, and uses explicit models of the relationships between the agents to drive both detection and diagnosis.

Huber and Durfee [3] use plan recognition for coordination. They do not assume an explicit relationship model, but instead assume opportunistic agents, which coordinate with others when it suits

their individual goals. Because of this, they have no guarantees of failure: when an agent no longer maintains a relationship, it may just have opportunistically found a better goal to pursue. Work on teamwork [5], [11] concentrates on establishing and preventative maintenance of collaboration relationships, often by attempting to establish mutual belief (undecideable in theory) in team goals and plans. SAM offers useful complementary capabilities, using plan-recognition to attempt detection of cases when the establishment or maintenance of mutual belief fails. It also generalizes to coordination, similarity, etc., and to non-participatory monitoring.

7. SUMMARY AND FUTURE WORK

This paper presents SAM, a fully implemented novel framework used by our agents for social execution monitoring in multi-agent domains. SAM uses models of relationships among agents to drive monitoring, not a model of the task to be executed. It is able to capture failures in maintaining relationships even when each individual is correct, and offers significant improvements to the goal-attentive schemes. We explored SAM in single and distributed configurations, with and without explicit representation of ambiguity, demonstrating its effectiveness by systematic empirical evaluation and rigorous mathematical analysis. Key novel aspects in this paper include: (a) an overall framework, SAM, for *social* monitoring, (b) practical algorithms for social monitoring with proven guarantees of detection, (c) integration of plan-recognition and communication for monitoring, (d) a simple distributed monitoring algorithm proven to be better than a complex centralized scheme, and (e) a condition for risky-points in which agents are to communicate regardless of monitoring state. Future work includes further exploration of distributed diagnosis, and additional relationship models (besides the coordination, collaboration, and role-similarity already implemented).

8. ACKNOWLEDGEMENTS

This work was supported in part by NSF Grant ISI-9711665, and in part by AFOSR contract #F49620-97-1-0501. We thank Jeff Rickel and the anonymous reviewers for their useful comments. As always, special thanks to K. Ushi.

9. REFERENCES

1. Atkins, E.M.; Durfee, E.H., and Shin, K.G. Detecting and Reacting to Unplanned-for World States. in Proceedings of *the National Conference on Artificial Intelligence (AAAI-97)*. 1997.
2. Doyle, R.J.; Atkinson, D.J., and Doshi, R.S. Generating Perception Requests and Expectations to Verify the Execution of Plans. in *the National Conference on Artificial Intelligence (AAAI-86)*. 1986.
3. Huber, M.J. and Durfee, E.H. An Initial Assessment of Plan-Recognition-Based Coordination for Multi-Agent Teams. in Proceedings of *the Second International Conference on Multi-Agent Systems (ICMAS-96)*. 1996.
4. Hudlicka, E. and Lesser, V., Modeling and Diagnosing Problem-Solving System Behavior. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987. **17**(3): p. 407-419.
5. Jennings, N., Controlling Cooperative Problem Solving in Industrial Multi-Agent System Using Joint Intentions. *Artifical Intelligence*, 1995. **75**: p. 195-240.
6. Kaminka, G.A. and Tambe, M. What's Wrong With Us? Improving Robustness through Social Diagnosis. in Proceedings of *the National Conference on Artificial Intelligence (AAAI-98)*. 1998.
7. Levesque, H.J.; Cohen, P.R., and Nunes, J. On Acting Together. in Proceedings of *the National Conference on Artificial Intelligence (AAAI-90)*. 1990. Menlo-Park, CA: AAAI Press.
8. Reece, G.A. and Tate, A. Synthesizing Protection Monitors from Causal Structure. in Proceedings of *Artificial Intelligence Planning Systems (AIPS-94)*. 1994. Chicago, IL.
9. Schroeder, M. and Wagner, G. Distributed Diagnosis by Vivid Agents. in Proceedings of *Autonomous Agents 1997 (Agents-97)*. 1997. Marina del Rey, CA: ACM Press.
10. Sugawara, T. and Lesser, V., Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments. *Machine Learning*, 1998: p. To Appear.
11. Tambe, M., Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, 1997. **7**: p. 83-124.
12. Tambe, M.; Johnson, W.L.; Jones, R.; Koss, F.; Laird, J.E.; Rosenbloom, P.S., and Schwamb, K., Intelligent Agents for Interactive Simulation Environments. *AI Magazine*, 1995. **16**(1).
13. Washington, R. Markov Tracking for Agent Coordination. in Proceedings of *the Second International Conference on Autonomous Agents (Agents-98)*. 1998.
14. Williams, B.C. and Nayak, P.P. A Model-Based Approach to Reactive Self-Configuring Systems. in Proceedings of *the National Conference on Artificial Intelligence (AAAI-96)*. 1996.

ACTUAL PLANS			INTERPRETATION (A3's)			SOCIAL FAILURE			PHYSICAL FAILURE
Attacker A1	Attacker A2	Scout A3	A1	A2	A3	Occurred	Detected	Detect. Class	
J	J	J	J	J	J	-	-	True Negative	None
S	J	J	S	J	J	+	+	True Positive	A1 Fails to Receive
J	S	J	J	S	J	+	+	True Positive	A2 Fails to Receive
S	S	J	H	H	J	+	+	True Positive	A3 Message Lost
S	S	S	S	S	S	-	-	True Negative	Enemy not identified

Table 1. Scout's view in all permutations of the broken radio-link scenario (example 2).

ACTUAL PLANS			INTERPRETATION (A1's)			SOCIAL FAILURE			PHYSICAL FAILURE
Attacker A1	Attacker A2	Scout A3	A1	A2	A3	Occurred	Detected	Detect. Class	
S	S	S	S	S	S	-	-	True Negative	None
F	S	S	F	S	F	+	+	True Positive	A1 Vision Fails
S	F	S	S	F	S	+	+	True Positive	A2 Vision Fails
F	F	S	F	F	F	+	-	False Negative	A1, A2 Vision Fail
S	S	F	S	S	S	+	-	False Negative	A3 Vision Fails
F	S	F	F	S	F	+	+	True Positive	A1,A3 Vision Fails
S	F	F	S	F	F	+	+	True Positive	A2, A3 Vision Fails
F	F	F	F	F	F	-	-	True Negative	All agents' vision fails

Table 2. Attacker A1's view of the team in all permutations of example 1, using team coherence.

ACTUAL PLANS			INTERPRETATION (A1's)			SOCIAL FAILURE			PHYSICAL FAILURE
Attacker A1	Attacker A2	Scout A3	A1	A2	A3	Occurred	Detected	Detect. Class	
S	S	S	S	H	F	-	+	False Positive	None
F	S	S	F	H	S	+	+	True Positive	A1 Vision Fails
S	F	S	S	F	F	+	+	True Positive	A2 Vision Fails
F	F	S	F	F	S	+	+	True Positive	A1, A2 Vision Fail
S	S	F	S	H	F	+	+	True Positive	A3 Vision Fails
F	S	F	F	H	S	+	+	True Positive	A1,A3 Vision Fails
S	F	F	S	F	F	+	+	True Positive	A2, A3 Vision Fails
F	F	F	F	F	S	-	+	False Positive	All agents' vision fails

Table 3. Attacker A1's view of the team using the team-incoherence criteria (example 1).

ACTUAL PLANS			INTERPRETATION (A1's)			SOCIAL FAILURE			PHYSICAL FAILURE
Attacker A1	Attacker A2	Scout A3	A1	A2	A3	Occurred	Detected	Detect. Class	
S	S	S	S	H	F	-	+	False Positive	None
			S	H	S		+	False Positive	
			S	S	F		+	False Positive	
			S	S	S		-	True Negative	
F	S	S	F	H	F	+	+	True Positive	A1 Vision Fails
			F	H	S		+	True Positive	
			F	S	F		+	True Positive	
			F	S	S		+	True Positive	
S	F	S	S	F	F	+	+	True Positive	A2 Vision Fails
			S	F	S		+	True Positive	
F	F	S	F	F	S	+	+	True Positive	A1, A2 Vision Fail
			F	F	F		-	False Negative	
S	S	F	S	H	F	+	+	True Positive	A3 Vision Fails
			S	H	S		+	True Positive	
			S	S	F		+	True Positive	
			S	S	S		-	False Negative	

Table 4. Representing explicitly all matching hypotheses for A1's view of team (example 1, *portion of results*).

ACTUAL PLANS			INTERPRETATION (A3's)			SOCIAL FAILURE			PHYSICAL FAILURE
Attacker A1	Attacker A2	Scout A3	A1	A2	A3	Occurred	Detected	Detect. Class	
S	S	S	S	S	S	-	-	True Negative	None
F	S	S	F	S	S	+	+	True Positive	A1 Vision Fails
S	F	S	S	F	S	+	+	True Positive	A2 Vision Fails
F	F	S	F	F	S	+	+	True Positive	A1, A2 Vision Fail
S	S	F	H	H	F	+	+	True Positive	A3 Vision Fails
F	S	F	F	H	F	+	+	True Positive	A1,A3 Vision Fails
S	F	F	H	F	F	+	+	True Positive	A2, A3 Vision Fails
F	F	F	F	F	F	-	-	True Negative	All agents' vision fails

Table 5. Scout's (A3's) view of all permutations of the example 1, using team-coherence.