In Working Notes of *AI Meets the Real World (AIMTRW-98): Lessons Learned.* Stamford, CT. September ,1998.

The Role of Agent-Modeling in Agent Robustness

Gal A. Kaminka **Milind Tambe**

Information Sciences Institute
University of Southern California
Los Angeles, California 90292
{galk, tambe}@isi.edu

Chahira M. Hopper

Air Force Research Laboratory
AFRL/SNAT
Wright Patterson Air Force Base
OH 45433
hoppercm@sensors.wpafb.af.mil

Abstract

A key challenge in using intelligent systems in complex, dynamic, multi-agent environments is the attainment of robustness in face of uncertainty. In such environments the combinatorial nature of state-space complexity inhibits any designer's ability to anticipate all possible states that the agent might find itself in. Therefore, agents will fail in such environments, as the designer cannot supply them with full information about the correct response to take at any state. To overcome these failures, agents must display *post-failure robustness*, enabling them to autonomously detect, diagnose and recover from failures as they happen. Our hypothesis is that through agent-modeling (the ability of an agent to model the intentions, knowledge, and actions of other agents in the environment) an agent may significantly increase its robustness in a multi-agent environment, by allowing it to use others in the environment to evaluate and improve its own performance. We examine this hypothesis in light of two real-world applications in which we improve robustness: domain-independent teamwork, and target-recognition and identification systems. We discuss the relation between the ability of an agent-modeling algorithm to represent uncertainty and the applications, and highlight key lessons learned for real-world applications.

Introduction

A key challenge in using intelligent systems in real-world environments is the attainment of robustness in face of uncertainty (Toyama and Hager 1997). The explosion of state-space complexity completely inhibits the ability of any designer, human or machine, to specify in advance the correct response in each possible state (Atkins et al. 1997). In addition, an agent in such environments cannot completely and correctly sense their environment at all times. Examples of such domains include virtual environments for training (Johnson and Rickel 1997), combat simulations (Tambe et al. 1995), and automatic target recognition and identification (SPIE 1995). Robust (fault-tolerant) agents in such environments must be endowed with the capability to autonomously compensate for the inherent uncertainty of such environments. They must be able to autonomously detect, diagnose and recover from failure in sensors and communications, using inference to enhance the knowledge gained from sensors which have limited reliability.

Indeed, the difficulty of agent robustness has emerged as a key lesson learned in our work in real-world applications. In particular, for the past five years, we have been developing teams of pilot agents for DARPA's synthetic theater of war (STOW) combat simulation environment. This is a real-world, complex, dynamic simulation, commercially developed for the military (Tambe et al. 1995). Pilot teams in this environment face a range of uncertainties, from unscripted enemy behaviors, to pilot agents encountering differing, inconsistent or incomplete views of the world, to unexpected pilot or helicopter failures, to unexpected communication failures. Our first-hand experience here has been that enabling the pilot teams to perform multiple missions over the duration of a major simulation exercise requires a very high degree of robustness from participating agents and agent-teams.

To attack this problem, AI techniques have attempted to provide agents with the capability to demonstrate *post-failure robustness* (Toyama and Hager 1997), which allows the agents to autonomously detect, diagnose, and recover from failures as they occur. Previous approaches (e.g., Doyle et al. 1986, Williams and Nayak 1996) have attempted to improve the robustness of agents performance by specifying constraints which essentially allow an agent to compare the execution of its task to some ideal, specified by the designer in the form of a *self-model of the agent*, or *monitoring conditions* on the agents perceptions and actions. For instance, a common technique in robotics is to recognize and track objects by their color, which is fast and relatively reliable. The agent knows that it has lost sight of the object when the color is no longer in view. Or, as an example of model-based diagnosis, an agent may discover one of its internal components is faulty when the actions produced do not match the modeled ideal behavior.

While powerful in themselves, these approaches are geared towards use in a single-agent environment, and have several limitations in multi-agent settings. First, these approaches fail to take advantage of the opportunities existing in multi-agent settings, in that they do not allow the agent to compensate for its own sensor readings by utilizing knowledge of other agents' beliefs and/or behavior. For example, a driver of a car may infer the existence of an obstacle in the road from observing another car swerve. Or, a missile may recognize targets which are only partially viewed based on their observed behavior, rather than absolute visual identification. These examples, however, rely on the ability to reason about other agents, rather than the exact visual scene.

Second, these approaches do not consider the effects that failures in other agents have on the agent's own expected behavior. For example, flexible teamwork requires agents to maintain their coordination in face of unexpected failures in teammates. But this maintenance can only be carried out if the agents can detect when teammates are miscoordinating. Good teamwork requires the agents to compensate for distributed failures, where a number of agents may all fail together, and in different ways. Here again, the agents must reason about other agents involvement in executing the task.

Our hypothesis is that the key element which is lacking in previous approaches, preventing them from robust performance, is the ability to reason at a higher level about other agents. We have therefore extended our agents with agent-modeling capabilities: The ability to model the beliefs, intentions, and plans of other agents. We have developed several reactive agent-modeling techniques (for different applications), called RESC (REal-time Situated Commitments) and RESL (REal-time Situated Least-commitments), which enable an agent to infer the plans and beliefs of other agents from their observable actions. This capability enables our agents to improve their robustness by allowing them to take knowledge of other agents into account, compensating for some of the uncertainties in the environment. For example, knowledge of other agents allows a team-member to evaluate its performance, detect failures, diagnose and recover by comparing its own beliefs and plans to those of its teammates. Such knowledge can also be used to complement an agent's sensors by allowing it to indirectly sense the environment, through inference based on the behavior of other agents. For instance, a group of vehicles may not lend itself to recognition based on shape or color, but may be recognized by the particular formations used, tactics, etc. We will introduce the use of RESC and RESL in service of robustness through two applications: domain-independent teamwork, and automatic target recognition and identification (ATR ID).

In providing domain-independent teamwork with post-failure robustness capabilities, we have been developing SAM (Socially Attentive Monitoring), a plan-execution monitoring and diagnosis technique (Kaminka and Tambe 1998). In SAM, agents use agent-modeling to infer the plans and beliefs of teammates, and compare these to their own plans. This allows agents using SAM to detect failures and diagnose them in coordination by observations alone, and to recover.

In automatic target recognition, we have demonstrated preliminary use of RESL in enabling identification of groups of moving targets based not on their individual visual characteristics, but rather on their group behavior as a team. The key idea here is that the *behavior* of a group as a whole, while stemming from the individual agents composing it, can be recognized and used to identify the individuals. Additionally, the use of an agent-modeling technique lends itself to inferring the intentions of the targets in question, rather than just their identification.

These two applications highlight important lessons which we have learned from our investigation of real-world environments. One obvious lesson was that agents in such environments will sometimes fail. A second lesson is that post-failure robustness is greatly facilitated by explicit reasoning about the agents inhabiting the environment. But a more generalized lesson that we took from this research is that achieving robust performance may not be so much a question of coming up with an optimal technique, rather than a question of coming up with an optimal set of techniques, that complement each other. SAM and the previous approaches (condition monitoring, model-based diagnosis), for instance, offer complementary coverage of failure-space, rather than an overlapping one. Similarly, we hypothesize ATR-ID systems will benefit from incorporating agent-modeling methodology as part of the recognition process.

Motivating Application Domains and Examples

The motivation for our work comes from our two application domains, both of which involve challenging realistic environments, involving complexity, uncertainty, and multiple interacting agents. One is the development of synthetic pilot agent teams for a DARPA's STOW domain -- a commercially-developed high-fidelity distributed battlefield simulation environment in which up to thousands of agents (helicopter and aircraft pilots, tank commanders, infantry platoons, etc.) can all engage in virtual battlefield exercises. This real-world simulation includes uncertainties such as unscripted behaviors of other agents, unreliable communications and sensors, possibly incomplete task/mission specifications, etc. (Tambe et al. 1995). The other application domain is an Automatic Target Recognition and Identification system (ATR-ID). The challenge is to observe groups of moving vehicles and to recognize and identify the types of targets (e.g., tanks, anti-aircraft vehicles, etc.). Currently we are exploring this application within the context of a battlefield simulation.

The robustness issue in the first application domain (battlefield simulation) may be illustrated by the following two examples of failure: The first failure involved a scenario where a team of three helicopter pilot agents was to fly to a specified landmark position. Having reached this position, one of the team members, whose role was that of a *scout*, was to fly forward towards the enemy, verifying its position. The scout's two teammates (role: *attackers*) were to land and wait for its return to the specified position. All of the pilot agents were explicitly provided conditions to monitor for the landmark. However, due to an unexpected sensor failure, one of the attackers failed to sense the landmark marking the waiting position. So while the other attacker correctly landed, the failing attacker continued to fly forward with the scout, following the original plan which called for flying in formation! The failing attacker had clearly seen that the other attacker had landed, but it did not use this information to infer the position of the landmark. Furthermore, the other attacker and the scout did not complain to the failing attacker about its failure. In a second example, a similar team of three helicopters was to take off from the home base and head towards their battle position. One of the agents unexpectedly did not receive the correct mission specification, so while two of the agents began to fly out as planned, the failing agent kept hovering in place at the starting position indefinitely. Again, none of the agents complained about the unusual performance of the team.

We have collected dozens of such failure reports in this application domain during the last three years. While it is generally easy for the human designer to correct these failures once they occur, it is hard to anticipate them in advance. These failures occur despite significant development and maintenance efforts. Given the complexity of the dynamic environment, predicting all possible states and all possible interactions is impossible. Furthermore, these failures are not negligible. Rather, they are very obvious (to the human observer) catastrophic failures, for both individual agents and the team. In the second example above, not only was the single agent stuck behind unable to participate in the mission, but the remaining agents were unable to carry out the mission by themselves.

Furthermore, these failures are not due to a lack of domain expertise, as the domain experts expect some common sense handling of such failures even in the most structured military procedure. Indeed, by exercising social common sense, an agent may at least detect that something may be wrong. Social clues, such as (in the examples above) noticing that teammates are leaving while the agent is hovering in place, or that a team-member has landed while the team was flying in formation, would have been sufficient to infer that something may be wrong.

These characteristics underlying the need for robustness are true of ATR-ID systems as well. Such systems face the requirement to incorporate advanced algorithms that identify targets from images under an extensive set of operating conditions. These conditions may span several categories, from target and sensor related parameters, to sensing geometry and interacting environmental conditions. The targets may belong to several classes. There may be several variants of the same target, as well as the sensor. There may be external damage to the target. The sensors themselves may operate with varying degrees of squint, depression, resolution, frequency, noise levels and anomalies, single or multi look. The sensing geometry itself may include varying 6 degrees of freedom poses of target, squint and depression. The background may have varying levels of brightness, texture and various materials. In addition, environmental conditions, such as obscuration and layover, camouflage, revetment, may render the search space combinatorically explosive. The availability of this data for some sensors that are capable of operating day and night and for all weather conditions further increases the dimensionality of the problem space.

An ATR-ID system may have to continue to operate in the context of this large dimensionality, and make inferences on several hypotheses about the identities of the targets, even when parts of the information is unavailable, or is intermittently available, through uncontrollable circumstances. It is very important that the system correctly classify the targets, for example to avoid targeting friendly or neutral entities, and so it must overcome any failure of the visual recognition algorithms it employs. Current visual recognition algorithms may come to an impasse under certain conditions, and understanding the behavior of the group of targets as a whole (i.e., agent-group modeling) may contribute additional important information for the recognition task.

Agent Modeling for Robust Teamwork

Agent modeling is an important problem that deals with methods allowing an agent to represent knowledge about other agents, and to effectively reason about them. Plan recognition (Kautz and Allen 1986) is one aspect of this problem. We have been developing several agent-modeling techniques, based on the reactive-plan-recognition paradigm, which are all made to operate in real-time given the real-world environments involved in our application domains.

A common thread in our modeling techniques is that they utilize plans in an "analysis by synthesis" process (Anderson et al. 1990), effectively running the plans "in reverse" in service of plan-recognition. Besides eliminating the need for specialized off-line pre-processing (as is common in other plan-recognition work, e.g., (Kautz and Allen 1986, Huber and Durfee 1996), this enables uniform representation of the agent's own plans, beliefs, etc. and those of the modeled agents. As will be explained in later sections, this uniformity is critical for failure detection applications.

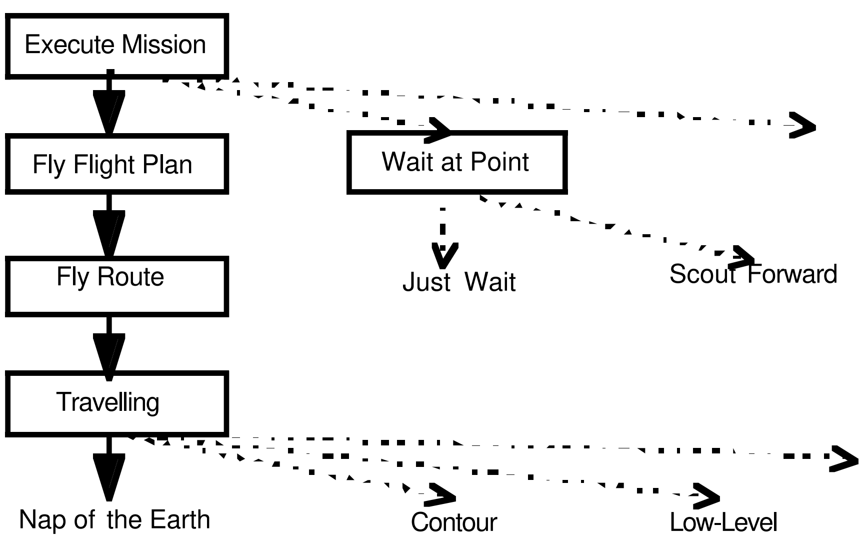

Figure 1. An example operator hierarchy.

Our agents' design is based on reactive plans (operators) (Firby 1987, Newell 1990), which form a decomposition hierarchy that controls each agent. Figure 1 presents a small portion of such a hierarchy. Each operator in the hierarchy has preconditions for selecting it, application conditions to apply it, and termination conditions. The design of the hierarchical plans uses the STEAM framework (Tambe 1997) for maintaining an explicit model of teamwork. Following this framework, operators may be team operators (that explicitly represent the joint activities of the team) or individual (specific to one agent). In Figure 1, boxed operators are team operators, while other operators are individual. The filled arrows signify the operator hierarchy currently in control, while dotted arrows point to alternative operators which may be used. In the figure, the agent is currently executing the execute-mission team operator which is its highest-level team operator, and has chosen (jointly with its team) to execute the fly-flight-plan operator, for flying with the team through the different mission-specified locations.

Following the concept of uniform representation, our agent modeling algorithms (RESC and RESL, explained below) both generate operator hierarchies to explain the observed actions of other agents. The operator hierarchies are inferred, of course, but are otherwise identical to the operator hierarchy that is actually in control of the agent doing the modeling[1]. Thus, the recognizing agent has uniform access to its own "in-control" hierarchy and the inferred hierarchies of other agents.

A key problem that haunts plan-recognition system is ambiguity, i.e., as an abductive process, there could potentially be several plans that match the observations (Kautz and Allen 1986). The two techniques we will show present two different solutions to this problem, which make them appropriate for different applications.

RESC: REal-time Situated Commitments

Our initial work in agent modeling resulted in the development of a reactive plan recognition algorithm called RESC. RESC was initially targeted for opponent-modeling applications in which the agent doing the modeling was to respond in real-time to its opponent's plans--real-time adversarial planning. RESC was therefore designed with the key feature of ensuring it had committed to a single worst-case interpretation of the opponent's behavior, allowing the agent to respond to the worst possible threat first. In other words, given several matching interpretations of the observations, RESC will commit to the one deemed to represent the biggest threat, rather than most likely explanation. RESC was described in detail elsewhere (Tambe and Rosenbloom 1995) and will therefore only be described here summarily.

To correctly provide team-modeling (as opposed to single agent-modeling) capabilities, RESC was extended in (Tambe 1996) to RESCteam, which followed the basic RESC framework as far as early commitments to recognized plans, but greatly enhanced the capabilities of recognizing team-plans -- plans that are being executed by teams jointly, rather than by a group of individuals. This was done by allowing RESCteam to reason about the role-constraints. In teamwork, agents are constrained by each other's role -- this is part of coordination. RESCteam attempted to use these constraints in understanding what the team is jointly attempting to achieve. For example, in air-combat simulation, RESCteam allows recognizing a maneuver in which agents move to the left and right, resp. as a pincer maneuver.

The recognition works by using the actions specified by operators as expectations for observable actions of the modeled agents. For example, the NOE operator sets the speed and altitude to that of a typical helicopter nap-of-the-earth flight: very low altitude above ground (6-8 meters), and relatively slow speed. RESC uses the operator's specified actions to create expectations -- so a helicopter being observed to fly at the appropriate altitude and speed potentially "matches" the NOE operator. In other words, a possible interpretation of that helicopters chosen plan is the NOE operator. Successful matches propagate up and down the inferred operator hierarchy to mark a single path (root-to-leaf) which is the chosen interpretation of the observations.

---

[1] There is a slight difference in terms of usage. Although the plans are completely uniform in form, a RESL-inferred hierarchy may contain multiple hypothesis, while the hierarchy in control of an agent is never ambiguous.

In case of ambiguity in the propagation (for example, two parents of a matching leaf), domain-specific information determines which interpretations are more significant and should therefore be preferred. In figure 2 below, the left hierarchy is the hierarchy actually in control of the agent. The right one is inferred by $RESC_{team}$ to be currently executed by another team (containing one or more agents).

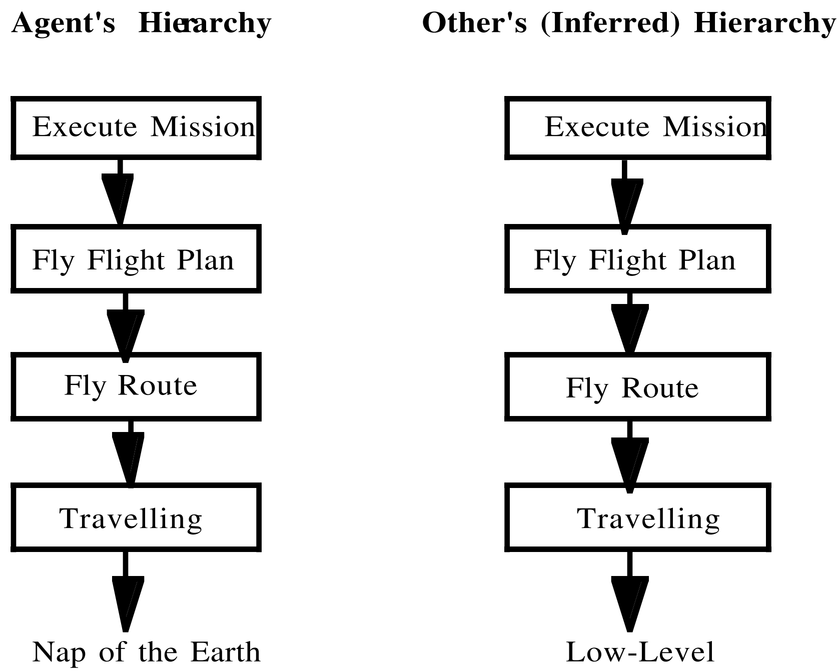

Figure 2. Two example hierarchies in the agent's memory.

In this case, the recognizing agent infers that the other agent is executing the execute-mission operator, in service of which it executes fly-flight-plan, etc. The observations in this case were matched against the leaf operator (low-level), and the successful match was propagated up.

RESC in Service of Teamwork

Teamwork (collaboration) is ubiquitous in multi-agent environments. It is characterized by a group of coordinating agents who are working together towards a joint goal. Recently, several promising models of teamwork have been suggested (Jennings 1995, Tambe 1997) which attempt to provide domain-independent teamwork capabilities. This work has been at least partially motivated by the constant need to patch ad-hoc domain-dependent coordination plans and replanning capabilities, and in that regard, can be viewed as an interesting attempt at a mix of pre-and post-failure robustness, i.e., as techniques for preventing failures in coordination and collaboration, and team-replanning capabilities upon detection of failure. To carry out this goal, teamwork models often require the establishment of "common grounds", a mutual belief shared by all members of the team.

While teamwork models have generally been successful at improving robustness of coordination plans by preventing them from happening, failures may still occur. In particular, teamwork models often rely of the establishment of mutual belief to make sure the team is coordinated. Yet mutual belief may be difficult to achieve in practice (Halpern and Moses 1990), where it is often achieved via communication (which can get blurred or even lost in realistic settings), or via common objects in the external environment that are assumed to be sensed by all agents (a risky assumption, at times, since sensors may fail). Indeed, the examples we have discussed in the previous section demonstrate that teamwork may breakdown under complex, real-world settings.

To address these robustness issues, we have developed a technique called SAM (Socially Attentive Monitoring) which relies on agent modeling to facilitate post-failure, observation-based robustness capabilities for agents in social settings in general, and teams in particular. SAM is described in detail elsewhere (Kaminka and Tambe 1998) and will only be introduced here as an application of agent-modeling for robustness.

Agent Modeling for Failure Detection. SAM is composed of three processes: (i) a failure detection process, which involves comparison with peers, in particular teammates, to detect the possibility of failure, (ii) a diagnosis process to confirm the detected failure and perform detailed diagnosis, and (iii) a recovery process. The key idea in the failure detection process is that agents can detect failures in their own or their peer's behavior by noting differences between their own state (where an agent's state may include its beliefs, goals, behaviors, etc.) and those of others. This process therefore relies on knowledge of the other agents' state, and on a representation that supports comparison between the agent's own state and those of others (as they are being modeled). This knowledge can, in practice, be acquired through either communications or agent-modeling techniques which rely on observations.

In many realistic domains, continuous communication incurs significant cost, both in overhead and risk. For example, in our battlefield simulation domain, the cost of communications is very high, as the agents operate in a hostile environment and expose themselves to risks by communicating with each other. In contrast, the cost of plan recognition is relatively low, being mostly a computational cost rather than a survival risk. In addition, in a team context, plan recognition is often quite reliable due to assumptions that can be made about the behavior of other agents, since they are team members. Our estimates of reliability and the cost make plan recognition an attractive choice for acquiring knowledge of others, and we have chosen RESCteam as it was readily available and fit the uniform representation requirement.

Using information supplied by RESCteam, SAM compares the agents' operator hierarchies by performing a top-down comparison of the operators in equal depths of the hierarchy. For instance, in figure 2, the difference that would be detected is between the two leaf nodes, since all operators above them are identical. Any such difference indicates a possible failure, if teammates were supposed to be executing similar operators in the first place.

Social Diagnosis. While SAM's detection process indicates possibilities of failures, its diagnosis process verifies the failure and generates an explanation for it utilizing social knowledge sources (other agents and the explicit teamwork model). These sources determine the expected similarity between the agents involved, and thus determine whether, and to what degree, the difference in operators is truly an indication of failure. In particular, differences can be detected at the team level (the monitoring agent and its teammates are not executing the same team operator), or individual level. The extent of the diagnosis and recovery depends on the type of difference detected.

In the case of team-operator differences, SAM's failure diagnosis is driven by the teamwork model. The key idea in SAM is based on the observation that teamwork models specify how a team should work in general. Thus, tracing back through such a model can help confirm and diagnose team failures. In our implementation, the agents utilize one such explicit model of teamwork, STEAM (Tambe 1997), for their collaborative execution of team operators. STEAM attempts to ensure that team operators are established jointly by the team via attainment of mutual belief in their preconditions, and terminated jointly by attaining mutual belief in the team-operator's termination conditions. In theory, team operators must therefore *always be identical* for all team members. However, as discussed before, establishing mutual belief in practice may be difficult. Furthermore, for security or efficiency, team members sometimes deliberately reduce communication, and inadvertently contribute to such team-operator differences. For instance, agents may assume that an external object in team members' visual range (such as a landmark) is visible to all, and may not communicate about it.

Given STEAM's guarantees that the team operator must always be identical, any team-operator differences detected by SAM therefore imply not only a possibility but a *certainty* of team coordination failure. Having established this certainty in the failure, SAM's team-level diagnosis next attempts to identify the exact differences between its own beliefs and its teammates' beliefs (again, provided by RESCteam) that have led them to execute different team operators--this aspect of diagnosis is key for recovery, and it proceeds as follows. First, given a difference between the monitoring agent's **T1** team operator and the other team-members' **T2** team operator, SAM infers that the entire team was initially executing **T1** (since no differences were detected earlier). However, now teammates have begun executing **T2**[2]. Therefore, SAM infers that the teammates believe that one or more of the disjunctive preconditions necessary for selection and execution of **T2** were satisfied. Furthermore, SAM infers that teammates believe that one or more of the disjunctive termination conditions of **T1** have been achieved. Typically, the intersection between these two sets of possible beliefs determines the actual set of beliefs that the teammates hold that are different at this point[3]. In addition, the teamwork model guarantees SAM that this is the only real difference that has led to the teammates' executing **T2**. Of course, this intersection idea can be applied to both team and individual operator differences, but it is of particular importance for team-level failures given the guarantees provided by the teamwork model.

In the example of the agent's failure to detect a key landmark, SAM infers that the other agents are carrying out the wait-at-point operator (one attacker lands, while the scout goes forward). Once this discrepancy is noted ("I am executing fly-flight-plan, they are executing wait-at-point"), SAM determines that since the other agents have terminated "fly-flight-plan" they have either met with an enemy, or reached the landmark. From their current choice of "wait-at-point" operator (whose preconditions include reaching the landmark), SAM infers that the teammates believe that they have indeed reached the landmark. Thus, given the guarantees of teamwork models, the difference in team operators is elaborated by the diagnosis process to infer specific differences in team members' beliefs.

Recovery. Recovery is greatly facilitated by better diagnosis. Currently, SAM's recovery assumes that the agent's perception is incomplete, but not inconsistent. For example, an agent's sensors may fail to detect the landmark (a "don't know" response), but would not erroneously say it is there when it isn't. Thus, SAM's diagnosis that teammates have come to believe in something which the monitoring agent does not know about (or vice versa) enables the monitoring agent to recover by adopting this belief (or in the reverse case, by letting others know about this belief). The above assumption is made only in the recovery stage, not in the detection or diagnosis stages, and removing it is a topic for future work.

In the example of the failure to detect the landmark, once the agent diagnoses the problem that the other agents have detected the landmark (making the "fly-flight-plan" achieved), it recovers completely by adopting the other agents' belief. This adoption of belief makes the preconditions for its own "wait-at-point" operator true, and it re-establishes mutual belief with the team, completely resolving the problem.

We have conducted a thorough evaluation of SAM, which is described in detail in (Kaminka and Tambe 1998). We examined SAM's performance on all possible permutations of the failure involving the landmark – allowing a single agent running SAM to play either scout or attacker, and looking at all 8 failure combinations the three helicopters – 16 experiments in all, of which 14 contained at least one failure. SAM was able to successfully resolve a large portion (11 out of the 14) of failures which cannot be captured at all using previous techniques. In fact, SAM was able to detect, diagnose, and recover from failures in other agents, which other technique cannot resolve even in theory. The agent modeling capability is essential in providing the basis for this type of diagnosis. Without the ability to reason about the knowledge that other agents possess, the agent cannot evaluate its performance by social means. It cannot detect the failure, and it cannot diagnose it.

---

[2] The case where the team did not switch but the monitoring agent did is also possible, but is not described here for brevity

[3] Empty intersection cases are a topic for future work.

Agent Modeling for Improved ATR-ID Robustness

The ATR-ID task is to recognize and classify target based on their observed characteristics--shape, color, appearance, and other elements of the visual scene. Generally, existing approaches to this problem use techniques from computer vision, image processing, model-based recognition and others to attempt to analyze the visual scene. However, key elements of the observed scene remain unutilized. If we know the targets are agents acting in the environment, we can use agent modeling techniques to help in the identification task. Different targets, depending on their intended tasks, will display different behaviors, and so the tasks may be inferred from observation. Thus in addition to matching a physical model to the observations to recognize targets, we propose matching a high-level intentional model -- if a target behaves as if it is a tank, it is likely to be one.

Our specific preliminary application involved recognizing groups of unidentified moving vehicles based on their speed, formation and other behavioral attributes. No visual information other than their location and movement is known. Here, even though there is still a need for real-time recognition, there is also critical need for explicit reasoning about the ambiguity in recognition. An agent-modeling system in service of an ATR-ID system must not return one interpretation if others exist as well, especially when trying to identify targets which could potentially be friendly or neutral. It should instead return all matching solutions, possibly ordered by likelihood, with a measure of confidence in each. This information can then be combined with other evidence, such as a visual recognition component. Also, unlike the adversarial-planning case, there is often time to consider the action to be taken based on the recognized targets, and more information about possible interpretations is useful. Therefore, an ATR-ID application requires an agent modeling algorithm in which the commitments to interpretations are done on a lazy basis -- only when absolutely necessary.

RESCteam's quick commitment to an interpretation, so vital for real-time adversarial-planning applications, proves too quick in this respect. Rather than reporting to the ATR-ID system that the recognition result is ambiguous, RESCteam comes back with its chosen interpretation, without providing the recognizing agent complete information about the uncertainty involved. This disallows the ambiguity to be cleared over a period of time by collecting more observations or other evidence.

RESL: REal-time Situated Least-commitments

To address this problem, we have been developing RESL (REal-time Situated Least-commitment). RESL allows multiple possible matching interpretations to be reasoned about at once, in parallel. The modeling agent therefore has access to all possible interpretations of the other agents' observable behaviors, and can reason explicitly about the relation between the possibilities and other source of evidence (for example, communications, visual-based recognition algorithms, etc.

In RESL's implementation, as in RESCteam, each hypotheses is a path through the operator hierarchy used to model the other agent. The difference is that while RESC and RESCteam decidedly maintain only a single path active at one time (thus committing to a single interpretation), RESL maintains all matching paths at the same time. In addition, RESL attempts to go from plan-recognition to belief ascription. When an operator matches, RESL infers that the preconditions necessary for the operator must be believed by the modeled agent. And when operator stops matching, RESL infers that the operator's termination conditions are believed by the modeled agent.

RESL works by first expanding the complete operator hierarchy for the agents being modeled, initializing all possible operators to a "non-matching" state. The modeled operator's preconditions are all flagged as non-matching, as are all termination conditions. All operator's action are set to be used as expectations on behavior. This process is described in Algorithm 1 (Init) below.

```
Init ( modeled agent )
{
    current hierarchy <- NIL
```

```
    for each operator in the hierarchy for modeling others:
    1. Add operator to current hierarchy
    2. Initialize operator's preconditions as a set of
       possible beliefs associated with the agent
    3. Initialize operator's termination conditions as a set
       of possible beliefs associated with the agent
    4. Initialize operator's actions as a set of expected
       observations associated with the agent

    return current hierarchy as the plan recognition network
}
```

Algorithm 1. Init operator-hierarchy for modeling by RESL.

Observations of other agents are continuously matched against the actions expected by the operators. Operators whose expectations match observations are tagged as "matching", and these flags are propagated along the hierarchy, so that complete paths through the hierarchy are flagged as matching or not. These paths specify the possible matching interpretations of the observations. In addition, precondition and termination conditions are flagged as true or not, signifying the inferred appropriate belief by the modeled agents. This process is described in Algorithm 2 (RESL) below.

```
RESL ( plan recognition network, modeled agent )
{
    get observations about agent

    // primitive matching
    for each operator that has a set of expected observations:
    1. attempt to match observations to expectations
    2. If succeed, flag operator as matching successfully
    3. If fail, flag operator as failing to match

    // propagate matching
    for each operator that is flagged as matching
    successfully,
    •  flag its parents as matching successfully
    for each operator whose children (all of them) are flagged
    as failing to match,
    •  flag it as failing to match

    // preconditions
    for each operator that is flagged as matching
    successfully,
    •  flag its associated set of preconditions as possibly
       true (the agent possibly believes this set of
       preconditions)

    // termination conditions
    for each operator that has just stopped matching
    successfully,
    •  flag its associated set of termination conditions as
       possibly true
}
```

Algorithm 2. RESL's main loop, matching observation and making inferences.

For example, suppose a group of vehicles is moving in a particular formation, following a tactic which can be attributed to either opponent tanks or friendly anti-aircraft vehicles. RESL may suggest these two hypotheses, rather than committing to one (as RESCteam would) and presenting it to the ATR-ID system. This would potentially allow the ATR-ID to combine this information with other source of evidence before committing to an action.

We have performed preliminary experiments in which a vehicle was identified based on it speed alone. The vehicle was traveling at certain speed, but its identity was unknown. RESL generated paths matching the operators in the hierarchy (Figure 3). These paths enable us to infer that the vehicle is either a tank or an anti-aircraft vehicles, since those are the preconditions for selections of the defend and attack operators. Normally, such inference regarding target ID would be passed on to the ATR-ID system, which can then combine this information (with the associated uncertainty) with other sources of evidence.

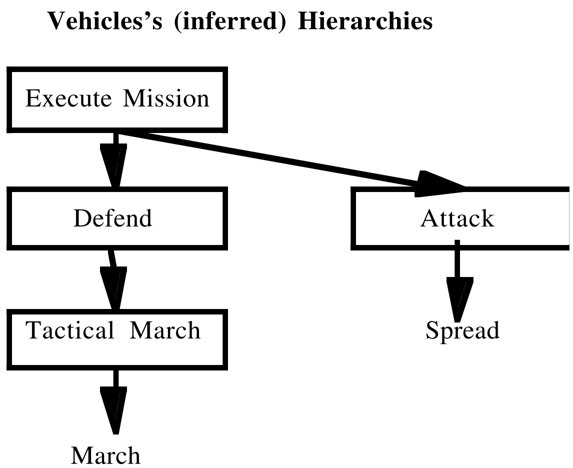

Figure 3. RESL-Inferred multiple hypotheses.

Now, as the vehicles slowed down or sped up, the recognized plans were disambiguated, and only a single path was left matching. This demonstrates the utility of this approach, complementing other ATR-ID methods, such as model-based recognition and others.

Discussion and Lesson Learned

Lesson 1. The need to face up to robustness as an issue in itself has been discussed before in the context of large-scale, real-world applications of AI (Toyama and Hager 1997). However, it does not rise in smaller-scale, simplistic domains. Our experience with both applications has proven again and again that agents in large-scale domains *will fail.* In addressing this issue, AI researchers must investigate applications in such domains.

Lesson 2. One obvious lesson was that post-failure robustness in multi-agent settings is greatly facilitated by explicit reasoning about the agents inhabiting the environment. Obviously, none of the applications described (SAM, and intentional recognition in ATR-ID) would be possible without an agent-modeling capability.

Lesson 3. Our experience with this issue was that the agent-modeling algorithm needs to be matched to the application that uses it. For example, RESCteam would be hazardous for use in the ATR-ID system, since it is overly zealous in ruling out matching hypotheses. In general, an approach such as RESL, supporting uncertainty and multiple hypothesis is beneficial in applications that have other sources of evidence.

Lesson 4. A more generalized lesson that we took from this research is that achieving robust performance may not be so much a question of coming up with an optimal technique, rather than a question of coming up with an optimal *set of techniques*, that complement each other. SAM and the previous approaches to agent robustness, such as condition monitoring and model-based diagnosis offer complementary coverage of failure-space, rather than overlapping one. In particular, SAM can diagnose failures that the other approaches can't, and vice-versa. Similarly, we hypothesize ATR-ID systems will benefit from incorporating agent-modeling methodology as part of the recognition process. The information it provides can be used to complement, not replace, the information received from the visual recognition algorithms regularly used in such tasks.

Related Work

There is generally little work on robustness reported on in the literature. Toyama and Hager (1997) discuss post-failure robustness in general, and provide a vision architecture that combines low-resolution/wide-coverage and high-resolution/low-coverage techniques for face tracking. Atkins et al. (1997) attack a similar problem of detecting states for which the agent does not have a plan ready. They offer a classification of these states, and provide planning algorithms that build tests for these states. However, their approach considers only the individual agent, and so has the same weaknesses as model-based and condition-monitoring approaches. Our work on using agent- and agent-team- modeling in service of robustness is thus novel.

SAM is related to work on multi-agent coordination and teamwork, although it generalizes to also detect failures in execution of individual operators, which are outside the scope of coordination. Particularly relevant are observation-based methods, which use plan recognition rather than communications for coordination. Huber and Durfee (1996) do not assume an explicit model of teamwork, but rather view collaboration as emergent from opportunistic agents, which coordinate with others when it suits their individual goals. These agents do not have the guarantees of maximal social similarity at the team level, and while they possibly will find the detected differences useful, they cannot be certain of failures, nor facilitate team recovery (since the other agent may simply have left the team opportunistically).

RESC and RESL are of course related to other work in plan recognition in general, and reactive plan recognition in particular. Specifically, Rao (Rao 94) presents a similar algorithm to that of RESL. However, the focus there was on task-independent agent modeling, while we are looking at the relation between a specific algorithm (RESL or RESC) and the applications which it serves. In addition, Rao's work does not extend to the belief ascription via inference of preconditions that we have utilized in both RESC and RESL.

Summary and Future Work

We have presented two applications of agent-modeling in significantly improving the robustness of agents in large-scale, complex, dynamic environments, involving multiple agents. One application, SAM, uses the RESCteam agent modeling algorithm in improving the robustness of agent teams, allowing team-members to detect and diagnose failures based on observations of their teammates. The other application, and ATR-ID system, utilizes the RESL algorithm, with its support for pursuing multiple hypotheses, to provide target recognition and identification based on behavior and high-level intentional models.

We hope to continue this line of research further by examining in greater detail the relationship between the agent-modeling algorithm and the application which it benefits. Given an application, we would like to be able to predict the characteristics of the agent modeling algorithm that is required to supply the information, and vice-versa. In addition, it is important to understand how to combine the more statistical nature of other evidence with the symbolic nature inherent to plan-recognition and agent-modeling algorithms.

We are also interested in continuing our development of agent modeling techniques, and their integration with other AI techniques. For example, both RESCteam and RESL currently use a static description of the plans. Making them adaptive, utilizing adaptation and learning technique, will not only have significant improvement in reduction of failures in recognition (as other agents may use a different set of plans), but also raises interesting questions regarding the effects of the learning on the application which uses the modeled information.

Acknowledgments

This research was supported in part by subcontract 090881-96-06 from Sverdrup Technologies. Additional support was provided by NSF grant IRI-9711665.

References

Anderson, J. R.; Boyle, C. F.; Corbett, A. T.; and Lewis M. W. 1990. Cognitive modeling and intelligent tutoring. In *Artificial Intelligence*, 42:7-49.

Atkins, E. M.; Durfee, E. H.; and Shin, K. G. 1997. Detecting and reacting to unplanned-for world states, in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. pp. 571-576.

Doyle R. J., Atkinson D. J., Doshi R. S., Generating perception requests and expectations to verify the execution of plans, in *Proceedings of AAAI-86*.

Festinger, L. 1954. A theory of social comparison processes. *Human Relations*, 7, pp. 117-140.

Firby, J. 1987. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-87).*

Grosz, B.; and Kraus, S. 1996. Collaborative Plans for Complex Group Actions. In *Artificial Intelligence*. Vol. 86, pp. 269-358.

Halpern, J. Y. and Moses, Y. 1990. Knowledge and Common Knowledge in a Distributed Environment., in *Distributed Computing* 37(3), pp. 549-587.

Huber, M. J.; and Durfee, E. H. 1996. An Initial Assessment of Plan-Recognition-Based Coordination for Multi-Agent Teams. In *Proceedings of the Second International Conference on Multi-Agent Systems.*.

Jennings, N. 1995. Controlling Cooperative Problem Solving in Industrial Multi-Agent System Using Joint Intentions. *Artificial Intelligence*. Vol. 75 pp. 195-240.

Johnson, W. L. and Rickel, J. 1997. Steve: An animated pedagogical agent for procedural training in virtual environments. In *SIGART Bulletin*, ACM Press, Vol 8., pp. 16-21.

Kaminka, G. A. and Tambe, M. 1998. What is Wrong With Us? Improving Robustness through Social Diagnosis, in *Proceedings of the 15th National Conference on AI (AAAI-98),* Madison, WI.

Kautz, A. and Allen, J.F. 1986. Generalized plan recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, pp. 32-37.

Kitano, H; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1995. RoboCup: The Robot World Cup Initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife*.

Levesque, H. J.; Cohen, P. R.; Nunes, J. 1990. On acting together, in *Proceedings of the National Conference on Artificial Intelligence (AAAI-1990)*, Menlo Park, California, AAAI Press.

Mossing, J.C. and Ross T. 1998. An Evaluation of SAR ATR Algorithm Performance Sensitivity to MSTAR Extended Operating Conditions, SPIE98.

Newell A., 1990. *Unified Theories of Cognition*. Harvard University Press.

Rao, A. S. 1994. Means-end plan recognition: Towards a theory of reactive recognition, in *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR-94)*

SPIE, 1995 Algorithms for Synthetic Aperture Radar Imagery, *SPIE Proceedings,* Vol. 2757.

Tambe, M.; Johnson W. L.; Jones, R.; Koss, F.; Laird, J. E.; Rosenbloom, P. S.; and Schwamb,

K. 1995. Intelligent Agents for interactive simulation environments. *AI Magazine*, 16(1) (Spring).

Tambe, M. and Rosenbloom P. S. 1995. RESC: An approach for real-time, dynamic, agent tracking. In *proceedings of IJCAI-95*.

Tambe, M. 1996. Tracking Dynamic Team Activity, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96).*

Tambe, M. 1997. Towards Flexible Teamwork, in *Journal of Artificial Intelligence Research*, Vol. 7. pp. 83-124.

Toyama, K.; and Hager, G. D. 1997. If at First You Don't Succeed..., in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97).* pp. 3-9.

Williams, B. C.; and Nayak, P. P. 1996. A Model-Based Approach to Reactive Self-Configuring Systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96).*