Tracking Dynamic Team Activity: An Extended Report

Milind Tambe
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
tambe@isi.edu
http://www.isi.edu/soar/tambe

March 21, 1996

Abstract

AI researchers are striving to build complex multi-agent worlds with intended applications ranging from the *RoboCup* robotic soccer tournaments, to interactive virtual theatre, to large-scale real-world battlefield simulations. Agent tracking — monitoring other agent's actions and inferring their higher-level goals and intentions — is a central requirement in such worlds. While previous work has mostly focused on tracking individual agents, this paper goes beyond by focusing on agent teams. Team tracking poses the challenge of tracking a team's joint goals and plans. Dynamic, real-time environments add to the challenge, as ambiguities have to be resolved in real-time.

The central hypothesis underlying the present work is that an explicit team-oriented perspective enables effective team tracking. This hypothesis is instantiated using the *model tracing* technology employed in tracking individual agents. Thus, to track team activities, *team models* are put to service. Team models are a concrete application of the *joint intentions* framework and enable an agent to track team activities, regardless of the agent's being a collaborative participant or a non-participant in the team. To facilitate real-time ambiguity resolution with team models: (i) aspects of tracking are cast as constraint satisfaction problems to exploit constraint propagation techniques; and (ii) a cost minimality criterion is applied to constrain tracking search. Empirical results from two separate tasks in real-world, dynamic environments — one collaborative and one competitive — are provided.

A shortened version of this report is to appear in the National Conference on Artificial Intelligence, AAAI96

Content areas: Multi-agent systems, real-time systems, agent tracking, plan recognition

1 Introduction

In multi-agent domains, intelligent agents interact with each other, either collaboratively or non-collaboratively, to achieve their goals. Many of these multi-agent domains require the interaction to be dynamic and real-time. For instance, in education, intelligent tutoring systems interact with students to provide real-time feedback[12, 37]. In entertainment, projects such as interactive fiction[3], virtual immersive environments[21], and virtual theatre[11] all involve real-time and dynamic multi-agent interaction (collaborative and competitive). Similarly, in training, a recent thrust on dynamic, real-time simulations — e.g., realistic traffic[7], air-traffic control[25] and combat[27, 34] simulations — involves such collaborative and non-collaborative interaction among tens or hundreds of agents (and humans). Such interaction is also seen in robotic domains, e.g., collaboration by observation[19], *RoboCup* robotic soccer tournaments(beginning IJCAI-97)[17].

In all these environments, *agent tracking* — monitoring other agents' observable actions and inferring their high-level goals, plans and behaviors — is a central capability required for intelligent interaction[1, 26, 35, 37]. While this capability is obviously essential in non-collaborative settings, it is also important in collaborative settings, where communication may be restricted due to cost, risk, lack of a common protocol etc.[13]. The key to this capability is tracking an agent's flexible mix of goal-driven and reactive behaviors, as seen in dynamic, interactive, multi-agent domains. This contrasts with previous work in the related area of plan recognition[15, 2], which mostly focuses on recognizing plans in static, single-agent domains.

This paper takes a step beyond tracking individual agents — the current state of the art in agent tracking and plan-recognition — by focusing on team tracking. We (humans) see team activity all around, e.g., teamwork in games (soccer, hockey or bridge), an orchestra, a ballet, a discussion, a play, etc. Naturally, this teamwork is being reflected in virtual and robotic agent worlds, e.g., RoboCup. The key in tracking such teamwork is to recognize that it is not merely a union of individual simultaneous activity, even if coordinated[10, 6, 16, 14]. For instance, ordinary automobile traffic is not considered teamwork, despite the simultaneous activity, coordinated by traffic signs[6]. Teamwork involves team members' joint goals and joint intentions, i.e., joint commitments to joint activities[6]. Consequently, tracking teamwork as independent activities of individual members is difficult. Consider the example of two children collaboratively building a tower of blocks[10] — they cannot be tracked as building two individual towers of blocks with gaps in just the right places. Similarly, in soccer, the collaborative pass play of two attackers cannot be tracked by focusing on their independent activities — a (robotic) defender should track their dynamic teamwork. Success in robotic collaboration by observation[19] would also require tracking such joint activities.

Thus, team tracking raises the novel challenge of tracking a team's joint goals and intentions. Dynamic, real-time domains, the focus of the current work, add to this challenge in two important ways. The first is tracking a team's reactive behaviors, where team members or subteams may dynamically undertake different aspects of the team's joint task; but in so doing may be unable to fully synchronize their activities. The second is tracking a team's behaviors in real-time. The main difficulty here is the ambiguity in team members' actions

— disambiguation requires time. Yet, for timeliness in interaction, a tracker (tracking agent) must frequently resolve such ambiguity in real-time. While real-time disambiguation is more of a challenge in team tracking — actions of multiple agents have to be disambiguated — the jointness of teamwork is itself key in addressing this challenge. In particular, given this jointness, recognizing one team-member's actions helps to disambiguate other members' actions.

Previous approaches to tracking individual agents fail to meet these team-tracking challenges. To concretely illustrate the problems faced by these approaches, we describe a typical simulated air-combat scenario (Figure 1), from a real-world combat simulation environment[5]. We focus on this scenario since it demonstrates the above team tracking challenges in the context of *real-world* teamwork. Here, a pilot agent **D** confronts a team of four enemy fighters **J**, **K**, **L** and **M**. In Figure 1-a, **D** detects the four opponents turning towards its aircraft, and infers that they are approaching it with hostile intent. In Figure 1-b, the four opponents split up into two subteams, and begin a *pincer* maneuver[31]. That is, one subteam (**J** and **K**) starts turning left, while the other subteam (**L** and **M**) starts turning right. Their goal is to trap **D** in the center, and attack it from two sides. However, **D** correctly tracks the pincer, and counteracts it by turning away from the center. Although this puts the second subteam (**L** and **M**) outside of **D**'s radar sight, correctly tracking the pincer enables **D** to anticipate the subteam's possible actions.

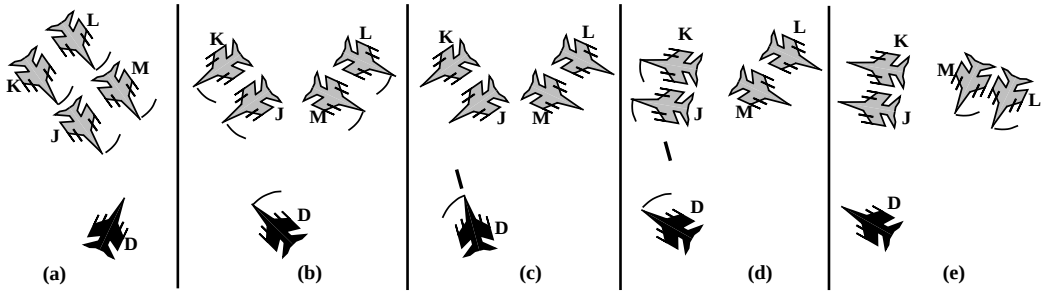

Figure 1: Simulated 1vs4 air-combat: an arc on an aircraft's nose shows its turn direction.

In Figure 1-c, upon reaching its missile firing range, **D** turns and fires a missile at **J**. In Figure 1-d, **D** executes an *Fpole* turn, to provide radar guidance to its missile, without flying right behind the missile. Although **D**'s missile is invisible to their radars, **J** and **K** track **D**'s maneuvers and infer a missile firing. Therefore, in Figure 1-d, they attempt to evade the missile (actually its radar-guidance) via a 90^0 *beam* turn. While beaming defeats **D**'s missile, it also, unfortunately, disrupts the team's pincer. Basically, unaware that **J** and **K** have reneged on their part of the pincer, the second subteam (**L** and **M**) continues with its part. Anticipating this second subteam's possible turn behind its (**D**'s) back (Figure 1-e), **D** plans an appropriate response.

One key point of this scenario is a concrete illustration of the need to track a team's joint goals and intentions. In Figure 1-b, for instance, the four opponents are not executing independent left and right turns! They are together executing a pincer. Indeed, it is only an accurate interpretation of the joint pincer that enables **D** to effectively counteract it. Recognizing this jointness also enables **D** to track a subteam's (**L** and **M**) behaviors when

it is not visible.[1]

Previous approaches[1, 26, 37, 12, 35], that focus on tracking individual agents, fail to track such joint team activities. In particular, these approaches are based on *model tracing*, which involves executing an agent's runnable model, and matching the model's predictions with actual observations. However, an individual's model does not express a team's joint goal and activities. One dramatic illustration of this problem is in **D**'s inability to track its opponents' joint pincer in Figure 1. **D** may possibly execute an individual model to track an individual, such as **J**, as executing a pincer. However, **J**'s singlehanded pincer is meaningless, since a pincer mandates the participation of two or more agents. This expressive inadequacy persists even if all agents are tracked as simultaneously executing individual pincers, e.g., is this one pincer with all four agents involved (Figure 1-b) or two separate pincers with two agents each?

Some recent work has attempted to go beyond individuals and track multiple agents. Tambe[33] tracks a group of agents engaged in identical activity. Yet, a group (e.g., cars driving in ordinary traffic) differs from a team (e.g., driving in a convoy) precisely due to the lack of any jointness of purpose. Thus, Tambe's approach, for instance, fails to interpret that the two subteams in Figure 1-b are engaged in a joint pincer. An alternative approach[28], although focused on individuals, attempts to indirectly track a team — it tracks one individual within the team, whose model includes a teammate's coordinated actions. However, this is still inadequate to express a team's activity, e.g., in Figure 1, **D** must track the whole team as attacking it, not just one opponent. Furthermore, tracking could fail if a teammate engages in uncoordinated actions.

In addition to their inexpressivity, the above approaches also have difficulties in real-time ambiguity resolution, as they fail to exploit a team's jointness. In Figure 1-b, for instance, a pincer is only one of many possible team tactics. The team could be beginning a *post-hole* tactic, where one subteam turns in a circle, to confuse **D** by disappearing and reappearing on its radar, while the second subteam attempts to attack **D**. Or, each subteam may possibly be attacking **D** independently. Jointness is the key to resolving such ambiguity in real-time (waiting is of course too hazardous for **D**). Thus, if one subteam is recognized as executing one half of the pincer, the other subteam must be executing the other half, and cannot be engaged in some unrelated activity. Unfortunately, unable to exploit such jointness, the above agent-oriented approaches engage in unconstrained search. A further problem hindering their real-time tracking is the inefficiency of tracking individuals, particularly given a large team (although group tracking[33] alleviates this aspect of inefficiency).

Finally, the above approaches also fail to address the dynamism in a team's jointness, particularly, dynamic formation and dissolution of subteams, and their sometimes unsynchronized activities. In the above scenario, for instance, team members begin with almost identical activities(Figure 1-a), then dynamically split into subteams to begin a joint pincer (Figure 1-b), and finally, one subteam dynamically deviates from its role (Figure 1-d).

The key hypothesis in this paper is that adopting a team perspective enables effective and efficient tracking of a team's activities, thus alleviating the difficulties dogging the agent-oriented approaches. In model tracing terms, this implies executing a team's runnable model, which predicts the actions of the team and its subteams (rather than separate models of

[1]Of course, it is also important to track individual agents, e.g., for **J** and **K** to detect **D**'s missile firing.

individual team members). A team model treats a team as a unit, and thus explicitly encodes joint goals and intentions required to address the challenge of tracking a team's joint mental state. Indeed, team tracking based on team models is among the first practical applications of the *joint intentions* theory developed in formalizing teamwork[6]. The paper shows that team models are uniformly applicable in tracking even if an agent is a participant in a team, rather than a non-participant. Furthermore, it shows that the team models are efficient: (i) they explicitly exploit a team's jointness to constrain tracking effort; and (ii) by abstracting away from individuals, they avoid the execution of a large number of individual agent models. This abstraction in team models also provides robustness, e.g., changes in number of team members may not disturb tracking. To track with such team models in real-time dynamic environments, we build on RESC[35], an approach for tracking individual agents in such environments. The new approach, RESC$_{team}$, is aimed at real-time, dynamic team tracking. RESC$_{team}$ incorporates several additions geared towards efficiency; key ideas introduced are not specific to RESC$_{team}$.

The remainder of this paper is organized as follows: Section 2 provides an overview of RESC. Section 3 describes tracking with team models. Section 4 focuses on enhancing tracking efficiency. Section 5 discusses mental simulations of unseen subteams. After a presentation of experimental results (Section 6), Section 7 concludes. The description below assumes as a concrete basis, pilot agents based on the Soar architecture[34]. We assume some familiarity with Soar's problem solving, specifically, applying operators to states to reach a desired state[24, 29].

2 RESC: Tracking Individual Agents

The RESC (REal-time Situated Commitments) approach to agent tracking[35] builds on *model tracing*[1, 37]. Here, a tracker executes a model of the trackee (the agent being tracked), matching the model's predictions with observations of the trackee's actions. One key innovation in RESC is the use of commitments. In particular, due to ambiguity in trackee's actions, there are often multiple matching execution paths through the model. Given real-time constraints and resource-bounds, it is difficult to execute all paths, or wait so trackee may disambiguate its actions. Therefore, RESC commits to one, heuristically selected, execution path through the model, which provides a constraining context for its continued interpretations. Should this commitment lead to a tracking error, a real-time repair mechanism is invoked. RESC is thus a repair-based approach to tracking (like repair-based approaches to constraint satisfaction[22] and natural language understanding[20]).

A second key technique in RESC leads to its situatedness, i.e., responsiveness to the present. To track the trackee's dynamic behaviors, it is necessary to execute the trackee's model so it is responsive to the changing world situation. A key assumption here is that the tracker (e.g., **D** in Figure 1) is itself capable of the flexible and reactive behaviors required in this environment. That is, the tracker's architecture can execute such behaviors. Therefore, this architecture is reused to execute the trackee's model to allow dynamic model execution. There is thus uniformity in the tracker's generation of its own behaviors, and its tracking of the trackee's behaviors.

To present a concrete example of RESC, we turn to Figure 1, and describe **D**'s tracking

of **J** in Figure 1-d, assuming **J** is the *only* opponent present (and assuming **D** is implemented in Soar). To illustrate the uniformity in acting and tracking, we first describe **D**'s generation of its own behaviors. Figure 2-a illustrates **D**'s operator hierarchy during its *Fpole* (Figure 1-d). The top operator, *execute-mission* indicates that **D** is executing its mission (e.g., defend against intruders). Since the mission is not complete, a subgoal is generated. Different operators are available in this subgoal, such as *fly-flight-plan* and *intercept*. **D** selects the *intercept* operator to combat its opponents. In service of *intercept*, **D** applies *employ-missile* in the next subgoal. Since a missile has been fired, **D** selects the *fpole* operator in the next subgoal to guide the missile with radar. In the final subgoal, *maintain-heading* causes **D** to maintain its heading. All these operators used for generating **D**'s own actions will be denoted with the subscript **D**, e.g., $fpole_{\mathbf{D}}$. $\text{Operator}_{\mathbf{D}}$ will denote an arbitrary operator in **D**'s operator hierarchy. $\text{State}_{\mathbf{D}}$ will denote **D**'s state. Together, $\text{state}_{\mathbf{D}}$ and the $\text{operator}_{\mathbf{D}}$ hierarchy constitute **D**'s model of its present dynamic self, referred to as $\text{model}_{\mathbf{D}}$.

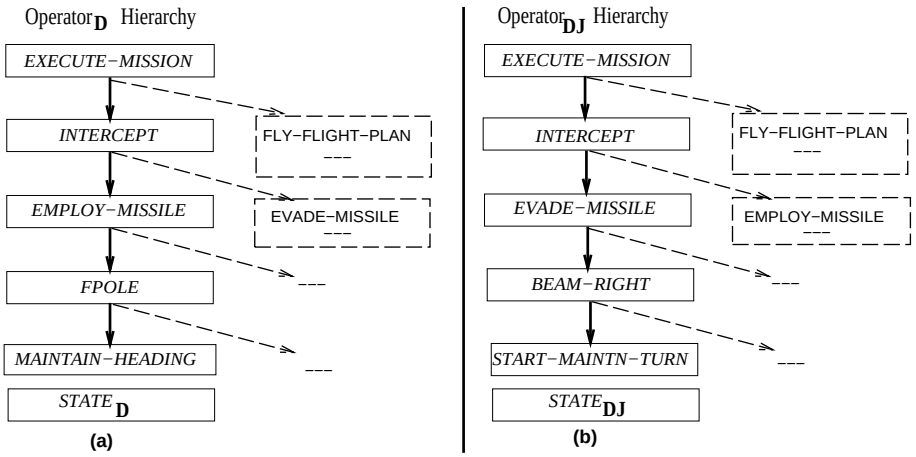

Figure 2: (a) $\text{Model}_{\mathbf{D}}$; (b) $\text{Model}_{\mathbf{DJ}}$. Dashed lines are unselected alternative operators. $\text{Model}_{\mathbf{D}}$ and $\text{Model}_{\mathbf{DJ}}$ need not be identical.

$\text{Model}_{\mathbf{D}}$ supports **D**'s dynamic behaviors, given Soar's architectural support for flexible operator selection and reactive termination[29]. **D** reuses this architecture in tracking. Thus, **D** uses a hierarchy such as the one in Figure 2-b to track **J**'s actions. Here, the hierarchy represents **D**'s model of **J**'s current operators in the situation in Figure 1-d. These operators are denoted with the subscript **DJ**. This $\text{operator}_{\mathbf{DJ}}$ hierarchy, and the $\text{state}_{\mathbf{DJ}}$ that goes with it, constitute **D**'s model of **J** or $\text{model}_{\mathbf{DJ}}$, used to track **J**'s behavior. For instance, in the final subgoal, **D** applies the $start\text{-}\&\text{-}maintain\text{-}turn_{\mathbf{DJ}}$ operator, which predicts **J**'s action and matches the prediction with **J**'s actual action. Thus, if **J** starts turning right towards beam, then there is a match with $\text{model}_{\mathbf{DJ}}$ — **D** believes that **J** is turning right to beam and evade its missile, as indicated by the higher-level operators in the $\text{operator}_{\mathbf{DJ}}$ hierarchy.

Such architectural reuse provides situatedness in RESC, e.g., $\text{operator}_{\mathbf{DJ}}$ may now be reactively terminated, and flexibly selected, to respond to the dynamic world situation. (Such architectural reuse is also possible with other architectures[11, 9].) As for RESC's commitments, notice that from **D**'s perspective, there is some ambiguity in **J**'s right turn in Figure 1-d — it could be part of a 90^o beam turn or a 150^o turn to run away. Yet, **D** commits

to just one operator$_{\mathbf{DJ}}$ hierarchy. This commitment may be inaccurate, resulting in a *match failure*, i.e., a difference between the model$_{\mathbf{DJ}}$'s prediction and the actual observed action. For example, if **J** were to actually turn 150°, there would be a match failure. RESC's primary repair mechanism to recover from such failures is "current-state backtracking", which involves backtracking over the operator$_{\mathbf{DJ}}$ hierarchy, within the context of the current continuously updated state. Thus, RESC attempts to generate a matching new operator$_{\mathbf{DJ}}$ hierarchy without re-examining past states.

3 Tracking with Team Models

To step beyond tracking individuals, and track a team's goals and intentions, team models are put to service. A tracker's model of a team consists of a team state and team operators. A team state is used to track a team's joint state, and it is the union of a *shared* part and a *divergent* part. The shared part is one assumed common to all team members (e.g., overarching team mission, team's participants). The divergent part refers to aspects where members' states differ (e.g., 3-D positions). One approach to define this divergent part is to compute a region or boundary encompassing all individual members. However, given the cost of computing such regions in real-time, the approach preferred in this work is to equate the divergent part to the state of a single *paradigmatic agent* within the team, e.g., the team's orientation is the paradigmatic agent's orientation (as in [33]). Such a paradigmatic agent is selected by a separate module (which currently selects one agent in a prominent location). Thus, a generic team Θ is represented as $\{m_p,\{m_1...m_p...\}\}$, where m_i are some arbitrary number of team members, and m_p is the paradigmatic agent. Θ may have N subteams, $\sigma_1...\sigma_N$, each also possessing its own members, state and paradigmatic agents. Unless subteams are known in advance, they are detected dynamically based on individual agent movements. Similarly, merging of subteams into a larger team is also detected. (Section 6 discusses details of detecting (sub)teams.) A dynamically detected subteam inherits the joint part, but not the divergent part. Thus, $\mathcal{T}$, the team of opponents in Figure 1-b consists of {**J**,{**J**,**K**,**L**,**M**}}, with two subteams $\mathcal{S}_1$={**J**,{**J**,**K**}} and $\mathcal{S}_2$={**L**,{**L**,**M**}}. For the sake of consistency, a single agent is considered a singleton team, which is its own paradigmatic agent: $\{m_1,\{m_1\}\}$.

A team operator in a team model represents the team's joint commitment to a joint activity. A key aspect of a team operator is the notion of a *role*, which defines an activity that a subteam undertakes in service of the team operator. For instance, in the game of bridge, opponents' *bidding* team operator has two roles, e.g., NORTH and SOUTH. The *pincer* team operator in Figure 1-b has two roles LEFT and RIGHT. These roles are exhaustive and specify subteams' activities in service of the team operator. Roles also embody a *role coherency* constraint, i.e., there must be one subteam per role for the performance of the team operator. However, roles for a single operator need not all be distinct. Furthermore, a single operator may be defined in multiple ways via multiple role combinations. For the team Θ, a team operator with $\mathcal{R}$ roles is denoted as operator$_\Theta$< $\gamma_1, ..., \gamma_R$ >. The children of this operator in the operator hierarchy must then define the activities for subteams in these roles. Thus, for instance, the opponents' pincer in Figure 1-b is denoted *pincer*$_\mathcal{T}$<LEFT,RIGHT> (and **D**'s model of it is denoted *pincer*$_{\mathbf{D}\mathcal{T}}$<LEFT,RIGHT>). If a team operator has only a

single role, that will not be explicitly denoted.

The RESC$_{team}$ approach to track team activity is now specified as follows:

1. Execute the team model on own (tracker's) architecture. That is, commit to a team operator hierarchy and apply it to a team state to generate predictions of a team's action. In doing so, if alternative applicable operators available (ambiguity):
   (a) Prefer ones where number of subteams equals number of roles.
   (b) If multiple operators still applicable, heuristically select one.
2. Check any tracking failures, specifically, match or role failures; if none, goto step 1.
3. If failure, determine if failure in tracking the entire team or just one subteam. If team failure, repair the team operator hierarchy. If one subteam's failure, remove subteam assignment to role in team operator, repair only subteam hierarchy. Goto step 1.

Step 1 reuses tracker's architecture for flexible team model execution, to track dynamic team activity. Step 1(a) selects among multiple operators based on the number of subteams, while 1(b) relies on domain-independent and dependent heuristics for such selection, e.g., one heuristic is assuming the worst about an opponent in an adversarial setting. The commitment in step 1 creates a single team operator hierarchy. With this commitment is RESC$_{team}$ always has a current best working hypothesis about the team activity, which provides it an anytime quality[4], important in real-time environments.

In step 2, tracking failure is redefined in RESC$_{team}$. Match failure — where a team's actions (e.g., orientation) does not match RESC$_{team}$'s current predictions — is certainly a tracking failure. However, in addition, inaccurate commitments in RESC$_{team}$ can also cause *role failure*, a new tracking failure, which may occur in one of three ways due to violation of the role coherency constraint. First, *role overload* failure occurs if the number of subteams exceeds the number of roles in a team operator. Second, *role undersubscribe* failure occurs if the number of subteams falls short of the required number of roles — particularly, if subteams merge together. Third, *role assignment* failure occurs if the number of subteams equals the number of roles, but they do not match the roles (see Section 4.1). Both match and role failures cause the same repair mechanism to be invoked — current-state backtracking — although in case of role failures, operators with higher (or lower) number of roles may be attempted next. (Abstract higher level operators are not susceptible to role overload failures, since they may not restrict the formation of subteams.) One novel issue in team tracking, outlined in step 3, is whether the match failure of one subteam is one of just that subteam or the whole team (discussed in Section 4.2). Thus, RESC$_{team}$ assumes that subteams begin a joint operator together, although over time, a subteam may deviate.

The result of RESC$_{team}$ in tracking the situation from Figure 1-a is shown in Figure 3-a. At the top-most level, *execute-mission*$_{\mathbf{D}\mathcal{T}}$ denotes the operator that $\mathbf{D}$ uses to track $\mathcal{T}$'s joint mission execution. Since $\mathcal{T}$'s mission is not yet complete, $\mathbf{D}$ applies the *intercept*$_{\mathbf{D}\mathcal{T}}$ operator in the subgoal to track $\mathcal{T}$'s joint intercept. In the next subgoal, *employ-weapons*$_{\mathbf{D}\mathcal{T}}$ is applied. Following that, *get-firing-position*$_{\mathbf{D}\mathcal{T}}$ tracks $\mathbf{D}$'s belief that $\mathcal{T}$ is attempting to get to a missile firing position, and so on. Each operator in this operator$_{\mathbf{D}\mathcal{T}}$ hierarchy indicates $\mathbf{D}$'s model of $\mathcal{T}$'s joint commitment to that activity.

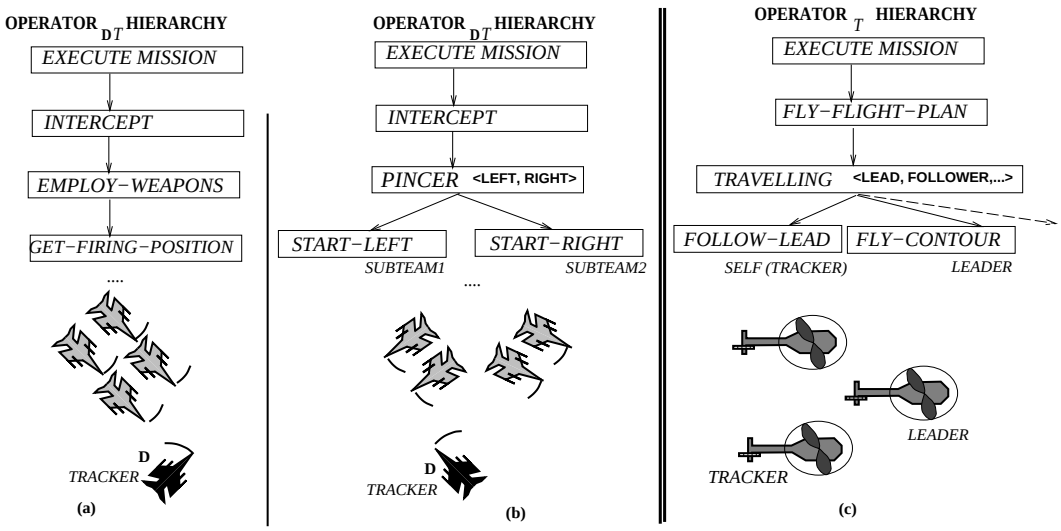

Figure 3: Team tracking: tracker a non-participant or a participant in the tracked team.

When the team in Figure 1-a splits into two subteams, role overload failure causes *employ-weapons*$_{\mathbf{D}\mathcal{T}}$ to fail. After current state backtracking, operator$_{\mathbf{D}\mathcal{T}}$ hierarchy in Figure 3-b results, which correctly tracks the on-going pincer. Here, *pincer*$_{\mathbf{D}\mathcal{T}}$<LEFT, RIGHT> has two roles. The children of this operator specify activities — starting left and right arms of the pincer — for the two subteams formed.

Team models and RESC$_{team}$ provide improved expressiveness, efficient ambiguity resolution and robustness required for team tracking. Team operators are expressive, since they explicitly encode a team's joint activity, with roles expressing different activities for subteams. For instance, the team operator hiearchy in Figure 3-b clearly expresses the team's joint pincer, the subteams involved and their roles in it. Team operators also facilitate real-time ambiguity resolution by enforcing jointness. Furthermore, role coherency in team operators adds further constraints, since subteams may only fill unassigned roles. For instance, in Figure 3-b, if one subteam is assigned to the LEFT role of a pincer, the other must also be part of the pincer, and in fact, must fulfill the RIGHT role. Additionally, team models also execute fewer operator hierarchies, e.g., instead of executing four separate operator hierarchies corresponding to four individual opponents, **D** executes only one team operator hierarchy. Even if subteam hierarchies are generated, they are still fewer than the number of agent hierarchies. Finally, team models provide robustness due to abstraction from individual agents to teams and subteams. Thus, tracking is not disturbed if agents switch allegiance from one subteam to another, or heretofore unseen agents emerge in a team, etc., unless this forms new subteams.

Team models and RESC$_{team}$ are applicable for tracking even if an agent is a collaborative participant in the team. Consider the scenario in Figure 3-c, which shows a team of simulated helicopters executing its mission[36], again in the real-world combat simulation environment[5]. Helicopter radio communications are often restricted to avoid detection by enemy. It is thus essential for a helicopter pilot agent to infer relevant information from the actions of its teammates, e.g., the team has reached a pre-specified holding area since

teammates have begun hovering. To track team activities, a team member executes a team model, using RESC$_{team}$. In Figure 3-c, the tracker happens to be a subordinate in the team, and the result of its tracking is the operator hierarchy shown. That is, the tracker believes its own team as jointly engaged in *execute-mission*. In service of mission execution, the team is flying a flight plan via a technique called *travelling*. *Travelling* involves a LEAD role, and two other FOLLOWER roles, causing the operator hierarchy to branch out.

The key point in Figure 3-c are the two types of uniformities shown. First, team models and RESC$_{team}$ are shown to be uniformly applicable in a collaborative situation. Second, the process of an agent's generation of its own actions and its tracking of its teammates' actions are also shown to be uniform. The tracker executes the *follow-leader* operator branch to *generate* its own behaviors, while executing the *fly-contour* branch to track the leader's flying along the terrain contour. (The third dashed branch is used to track the other follower).

There is yet a third point of uniformity in Figure 3-c: *self tracking*. In particular, while tracking others, an agent often monitors the progress of their low-level actions closely, to quickly detect any deviation from predicted action. However, such detailed monitoring of progress of its own actions is often absent (they are assumed to progress satisfactorily). Team models provide a uniform framework to view such monitoring. Indeed, upon experimentally enabling such detailed self-monitoring (self-tracking), a pilot agent was able to track situations where its commands were not controlling its simulated helicopter as expected. The key issue in self tracking is that when it fails, an agent may modify its behaviors, rather than its self model — an issue for future work.

3.1 The Joint Intentions Framework

While team tracking has received little attention in the literature, researchers are investigating teamwork both theoretically[10, 6, 16], and practically[14, 27]. Perhaps the most well understood among the theories is the joint intentions framework[6]. Very briefly, this theory states that a team jointly intends an activity if it is jointly committed to completing that activity (commitments have a common escape clause q). Joint commitment implies that (at least initially) team members have a mutual belief that they are each committed to that activity. Furthermore, a team jointly intending an activity, leads subteams to intend to do their share in that activity, subject to the joint intention remaining valid.

In a team model, a team operator selected in an operator hierarchy (as in Figure 3) is or tracks a joint intention in the above sense. Thus, team models are among the first practical applications of the joint intention theory ([14] describes another); and their application here is certainly novel — tracking team activities. This application raises one issue: joint intentions pack with them the responsibility of a team member when it privately comes to believe that the team's jointly intended activity is unachievable (or achieved) — this team member is left with the commitment to communicate this private belief to its teammates[6]. However, if communication itself is very costly — breaking radio silence may be risky for a helicopter pilot agent — such a commitment may be inappropriate. Thus, when tracking, RESC$_{team}$ does not assume that all subteams are aware of a subteam's deviation from its role(more in Section 4.2).

3.2 Team Mind?

Team models appear to track a "team mind". However, the notion of a team mind is controversial, given that the external world consists only of individual minds[30]. Team models are neutral in this controversy, and can also be reconciled with the view of individual minds. In particular, a dynamic team model may be considered a result of combining (partial) team models attributed to team members. For instance, in tracking a pincer, each subteam may be considered as executing at least the relevant half of the team model — where, as shown by the dashed lines, a subteam may or may not track its sibling subteam(Figure 4).

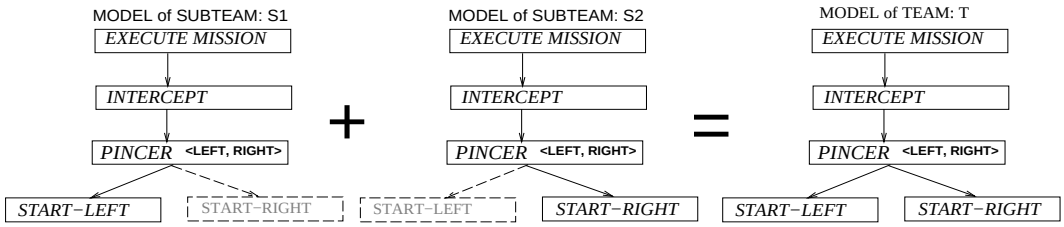

Figure 4: A Team model may be considered a combintion of models attributed to subteams.

4 Further Enhancements in Team Tracking

4.1 Efficient Role Assignments

While team operators already constrain tracking effort by enforcing jointness and role coherency, role assignment in team operators can potentially be inefficient. Given a team operator with $\mathcal{R}$ roles, a tracker may need to test all $\mathcal{R}!$ permutations of subteam to role assignments; generating $\mathcal{R}$ children operators in each test. Furthermore, such a team operator could itself be defined in multiple ways, based on multiple possible role combinations. For example, *half-pincer*<STRAIGHT,RIGHT> and *half-pincer*<STRAIGHT,LEFT> are two separate definitions of half-pincer — in one role combination, one subteam flies straight to attack, while a second subteam attacks from the left; while the second combination involves an attack from the right. If there are $\mathcal{C}$ such role combinations, the total operators executed are $\mathcal{R}\times\mathcal{R}!\times\mathcal{C}$. Furthermore, observations of match failure are often not instantly available, so that testing each role assignment may take time. Thus testing role assignments in real-time can potentially be difficult.

To alleviate this inefficiency, multiple definitions of a team operator (if any), corresponding to its multiple role combinations, are unified together, with explicit constraints to define allowable role combinations. The role assignment problem for this unified operator is now cast as a single constraint satisfaction problem (CSP)[18]. In this CSP, subteams are variables and possible roles are the domains of those variables. Constraints are the explicit constraints among roles, i.e., their allowable combinations. Figure 5 shows the role assignment for the *half-pincer* cast as a CSP. Subteams $\mathcal{S}_1$ and $\mathcal{S}_2$ are variables, with the roles of

half-pincer as their domains. The static constraints specify that in a half-pincer, one subteam must take on the role STRAIGHT, the other must take on either RIGHT or LEFT. The dynamic observations of subteam actions provide unary constraints, as they rule out non-matching values from the domains.

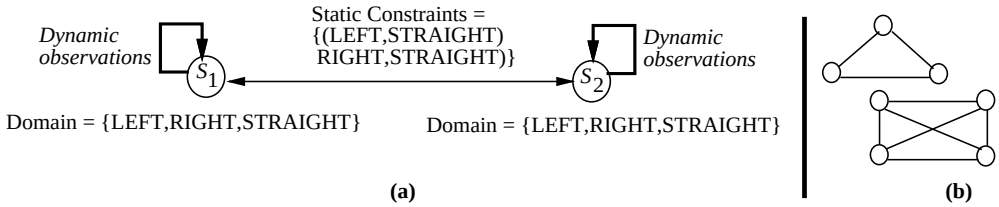

Figure 5: (a) Detailed illustrative CSP; (b) other constraint graphs tested.

Unifying role combinations and casting the problem as a CSP provides several benefits. First, while detecting match failures is the simplest consistency check (*node consistency*), other constraint propagation techniques such as *arc consistency* or *path-consistency*[18, 8] can acclerate the process of role assignment (or detecting failures). For instance, in Figure 5, if node-consistency rules out the role STRAIGHT for subteam $\mathcal{S}_2$, then arc-consistency will automatically rule out roles LEFT and RIGHT for $\mathcal{S}_1$. This is important, given complex constraint graphs tested (Figure 5-b), and the absence of immediate observations. Additionally, testing the node consistency of variables independently converts the multiplicative effect of generating combinations of role assignments into an additive effect. In general, given this mapping, more of tracking could be cast as a CSP — an issue for future work.

Based on the above, RESC$_{team}$ was modified so that when executing a team operator with more than one role, it does not assign subteams to roles and then test that assignment (unless the role assignments are known). This enables roles to be assigned to subteams via CSP. So far, only arc-consistency has been incorporated in RESC$_{team}$. RESC$_{team}$, being a repair-based approach, solves this CSP via a repair-based approach[22]. It commits to one assignment of roles to subteams, while obeying static constraints. If dynamic observations rule out a role for a particular subteam, arc-consistency rules out inconsistent roles for other subteams. Roles are then reassigned to failed subteam assignments. (See Section 6 experimental results.)

4.2 Minimum Cost Repair in Tracking

Repairing role assignments to subteams is, however, more complex than indicated in the previous section. In particular, team tracking raises a novel issue — ambiguity about a subteam's degree of adherence to its role. If a subteam is strongly adherent it is very likely to fulfill its role in the joint activity; but if it is weakly adherent, it may deviate. Thus, when a subteam is observed to not fulfill its role, there are two possible explanations: (i) the subteam being strongly adherent is fulfilling its role, but there is an error in tracking the entire team tactic; or (ii) this single subteam being weakly adherent is deviating from its

role to respond to some event.[2] As a concrete example consider the situation in Figure 1-d, where we assumed that a subteam being weakly adherent, responded to a missile firing by abandoning the team's on-going pincer. However, if this subteam were known to be strongly adherent, then it would have never deviated from its role, and thus there is an error in tracking the pincer — the whole team was not executing a pincer. A symmetrical issue arises if a subteam is seen to fulfill its role despite a reason to deviate. This could be because it is strongly adherent; but if it is weakly adherent, then there is an error in tracking.

This ambiguity greatly increases the search space of repairs in RESC$_{team}$. To tame the search, RESC$_{team}$ works with the heuristic of *minimal cost repair* — where cost measures the amount of repair effort required to continue error-free tracking. Zero repair to the team model is naturally considered lowest cost repair. Repairing a single subteam's model (i.e., operators involving just a single subteam) within the team model is considered to have higher cost (given a possible reason for the subteam deviation). Repairing and retracking operators involving the entire team is considered to be even higher cost.

The tracker uses this minimal repair cost heuristic in attributing a degree of adherence to subteams. Thus, if a subteam is fulfilling its role despite events that could cause deviation from that roles — e.g., if a missile has been fired at the subteam, or if other subteams have abandoned their roles — RESC$_{team}$ attributes strong adherence to the subteam; this attribution suggests lowest cost (zero) repair for continued error-free tracking. If, however, a subteam does not fulfill its role during such an event, then RESC$_{team}$ attributes weak adherence to the subteam — the event supports a cheap repair to explain this single subteam's deviation. Thus, RESC$_{team}$ assumes that this single subteam has abandoned its role in the joint activity. In terms of the CSP implementation, this abandonment implies disabling outward constraint propagation from this subteam variable. Thus, this subteam's future actions are not tracked as part of their joint activity; however, the other subteam is assumed to continue. In contrast, if a subteam deviates *without* an event that could explain its deviation, RESC$_{team}$ assumes an error in tracking the whole team (applies normal CSP).

The rationale behind the above heuristic is two-fold. Pragmatically, in real-time environments, it helps to reduce computational (repair) expense. Theoretically, it leans towards parsimonious explanations (the heuristic was inspired by minimality of fault models[32]). This heuristic is actually more general, and in fact, it is already incorporated in *current-state backtracking*: repair the lowermost operators in an operator hierarchy before higher level ones.

5 Mental Simulations of Invisible Subteams

One important benefit of recognizing the jointness of a team is anticipating/tracking an unobservable subteam's activity (see Section 1). RESC$_{team}$ assumes that an invisible subteam will fulfill its role in its on-going joint activity; it thus mentally simulates the effects of those actions. This simulation is carried out in an abstract visualization subcontext, and its results are input to tracking, as though it is information received via observation. RESC$_{team}$ uses

[2]Such ambiguity in the level of adherence is present even in tracking individuals; the difficulty in team tracking is that the whole team may be implicated.

that to predict/track the next possible action for the subteam. Currently, visualizations are attempted at fixed time intervals; a more sophisticated simulation is also left for future work.

6 Implementation Results and Evaluation

To evaluate RESC_{team}, we have implemented experimental variants of Soar-based pilot agents for both simulated fighters[34] and helicopters[36]. The original pilot agents have participated in various large-scale exercises, including one involving expert human pilots[34]. Our experimental pilots (called $\text{pilot}^{tracker}$) incorporate RESC_{team} (contain over 1000 rules). These agents can track teams: opponents' teams in the case of fighter pilots and their own team in case of helicopter pilots. Promising results (see below) with these $\text{pilot}^{tracker}$ agents has led the techniques, including team models, to be ported to the original agents.

We have run the $\text{pilots}^{tracker}$ agents in several combat simulation scenarios outlined by our human experts. Figure 6-a compares a fighter $\text{pilot}^{tracker}$'s performance when tracking with team models versus when using individuals' models. The scenario in Figure 1 is used as a basis for comparison, with four agents in the opponents' team. Figure 6-a shows the percent of its total time[3] that $\text{pilot}^{tracker}$ spends in acting and tracking. Thus, when using team models in tracking, a $\text{pilot}^{tracker}$ spends only 18% of its time is tracking. In contrast, it spends 71% of its time in tracking when using individual agents' models. Basically, individual agents' models *fail* to correctly track the team's pincer. This failure is not simply in terms of inexpressivity, but also, unable to exploit the teams' jointness, they engage in a large unconstrained tracking effort. Thus, they run out of time, before they can each at least individually detect the pincer. Similarly, with team models, $\text{pilot}^{tracker}$ spends only 7% of its time in deciding on its own actions(SELF), since it can quickly and accurately track its opponents. In constrast, $\text{pilot}^{tracker}$ incorrectly readjusts its own maneuvers when using individual models; hence spends 28% of time deciding on its own actions.

Thus, when using team models, a $\text{pilot}^{tracker}$ spends 25% (18% TRACKING + 7% SELF) of time in mental activity, and the rest it waits. Waiting is essential because $\text{pilot}^{tracker}$'s maneuvers take time, e.g., to complete a turn, or reach missile range. When using individual agents' models, most of the cycles are spent tracking.

Figure 6-b provides similar comparative numbers for a team consisting of three opponents. The key point here is that team models are not wedded to a specific numbers of agents in a team. In these scenarios, $\text{pilot}^{tracker}$ assumes enemy fighters within three kilometers of each other as part of a single team (based on knowledge provided by human experts). When a subgroup of fighters separates by more than three kilometers subteams are assumed to have formed. In the helicopter domain, the team is known in advance.

Focusing only on role-assignment, Table 1 presents the reduction in tracking effort due to team model and the CSP formalism. Column 1 names different tactics[31]. Column 2 indicates the number of agents assumed involved in the tactic. Column 3 estimates the (worst-case) total number of operators searched assuming roles assigned to individual agents rather than subteams. Column 4 shows the factor reduction in the operators searched when

[3]Time is measured in simulation cycles. Since inefficiency percolates to all levels of the simulation when using individual models, they lead to fewer total cycles.

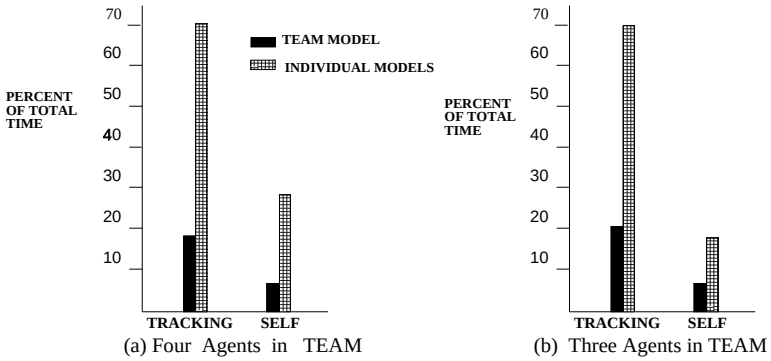

Figure 6: Comparing the percent of total time in acting and tracking when using team models versus when using individuals' models.

role assingment is based on teams/subteams. The final column shows the *actual* results from the RESC$_{team}$ implementation, with unification of operator roles, and CSP. The reduction in tracking effort is substantial, and it will only grow with increasing numbers of agents.

Tactic. name	Number of agents	Total operators agent models	Factor reduction via team model	Factor reduction via team model+CSP
Pincer	4	56	14	14
Half-pincer	4	112	14	22
Posthole	4	112	14	22
Pincer-trail	4	144	8	24

Table 1: Reduction in role assignment effort due to operator role-unification, team models and CSP.

7 Summary and Discussion

Animal and human (natural) world is full of collaborative and competitive team activities: a mongoose team surrounding a Cobra, a pack of cheetahs hunting a prey, games of soccer or cricket, an orchestra, a discussion, a coauthored paper, a play, etc. It is only natural that this teamwork is (and will be) reflected in virtual and robotic agent worlds, e.g., robotic collaboration by observation[19], RoboCup robotic (and virtual) soccer[17], social agents[23], virtual theatre[3, 11], virtual battlefields[34, 27]. If agents are to successfully inhabit such collaborative and competitive worlds, they must be proficient in understanding and tracking team activity. This paper has taken a step towards this goal and advanced the state of the art in agent tracking and plan recognition (which are currently focused on individual agent activities). Key contributions/ideas in this paper include: (i) the use of explicit team models for team tracking; (ii) uniform application of team models regardless of an agent's being a participant or non-participant in a team; (iii) demonstration of the key advantages of team models, specifically, efficiency, robustness and expressivity gained via jointness, and team

abstractions; (iv) Use of constraint satisfaction for improved tracking efficiency; (v) Use of a cost minimality criterion to constrain tracking search. We also touched on self tracking and mental simulations.

Team models are one of the first to practically instantiate the theoretical joint intentions framework (and certainly the first in service of tracking). While team models and other key ideas could be applied in different approaches to tracking, we presented one specific approach: RESC$_{team}$. Although based on RESC, RESC$_{team}$ includes several additions to address subteam formation(merging), role assignments and subteam deviations. RESC$_{team}$ has been applied to two different tasks in a simulated combat environment: (i) a collaborative task involving simulated helicopters; and (ii) an adversarial task involving fighter combat. These synthetic yet real-world teamwork tasks provide a solid foundation for further investigation of team tracking. Indeed, the field continues to investigate teamwork, and team tracking will need to reflect the advances made.

8 Acknowledgement

I thank Randy Hill for reading and commenting on two drafts of this paper, and Paul Rosenbloom for questioning the basic assumptions underlying it. Kevin Knight, Gal Kaminka and Micheal Wooldridge helped with useful comments.

This research was supported under subcontract to the University of Southern California Information Sciences Institute from the University of Michigan, as part of contract N00014-92-K-2015 from the Advanced Systems Technology Office (ASTO) of the Advanced Research Projects Agency (ARPA) and the Naval Research Laboratory (NRL); and under contract N66001-95-C-6013 from the Information Systems Office (ISO) of the Advanced Research Projects Agency (ARPA) and the Naval Command and Ocean Surveillance Center, RDT&E division (NRAD). Critical expertise and support has been provided by Bob Richards and Dave Sullivan of BMH Inc.

References

[1] J. R. Anderson, C. F. Boyle, A. T. Corbett, and M. W. Lewis. Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42:7–49, 1990.

[2] J. Azarewicz, G. Fala, R. Fink, and C. Heithecker. Plan recognition for airborne tactical decision making. In *Proceedings of the National Conference on Artificial Intelligence*, pages 805–811. Menlo Park, Calif.: AAAI press, 1986.

[3] J. Bates, A. B. Loyall, and W. S. Reilly. Integrating reactivity, goals and emotions in a broad agent. Technical Report CMU-CS-92-142, School of Computer Science, Carnegie Mellon University, May 1992.

[4] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 979–984. IJCAI-89, Morgan Kaufmann, August 1989.

[5] R. B. Calder, J. E. Smith, A. J. Courtemanche, J. M. F. Mar, and A. Z. Ceranowicz. Modsaf behavior simulation and control. In *Proceedings of the Conference on Computer Generated Forces and Behavioral Representation*, 1993.

[6] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.

[7] J. Cremer, J. Kearney, Y. Papelis, and R. Romano. The software architecture for scenario control in the Iowa driving simulator. In *Proceedings of the Conference on Computer Generated Forces and Behavioral Representation*, 1994.

[8] R. Dechter, Pearl., and J. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1988.

[9] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proceedings of the IEEE special issue on knowledge representation*, 74:1383–1398, 1986.

[10] B. J. Grosz and C. L. Sidner. *Plans for Discourse*, pages 417–445. MIT Press, Cambridge, MA, 1990.

[11] B. Hayes-Roth, L. Brownston, and R. V. Gen. Multiagent collaobration in directed improvisation. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95)*, 1995.

[12] R. Hill and W. L. Johnson. Situated plan attribution for intelligent tutoring. In *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, Calif.: AAAI press, 1994.

[13] M. Huber and E. Durfee. Deciding when to commit to action during observation-based coordination. In *Proceedings of the International Conference on Multi-agent systems (ICMAS)*, 1995.

[14] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.

[15] A. Kautz and J. F. Allen. Generalized plan recognition. In *Proceedings of the National Conference on Artificial Intelligence*, pages 32–37. Menlo Park, Calif.: AAAI press, 1986.

[16] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhard, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems, Lecture notes in AI 830*. Springer Verlag, New York, 1992.

[17] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife*, 1995.

[18] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 12(1), Spring 1992.

[19] Y. Kuniyoshi, S. Rougeaux, M. Ishii, N. Kita, S. Sakane, and M. Kakikura. Cooperation by observation: the framework and the basic task pattern. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1994.

[20] R. L. Lewis. An architecturally-based theory of human sentence comprehension. In *Proceedings of the Annual Conference of the Cognitive Science Society*, 1993.

[21] P. Maes, T. Darrell, B. Blumberg, and S. Pentland. Interacting with animated autonomous agents. In J. Bates, editor, *Proceedings of the AAAI Spring Symposium on Believable Agents*, 1994.

[22] S. Minton, M. D. Johnston, A. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the National Conference on Artificial Intelligence*, 1990.

[23] K. Nagao and A. Takeuchi. Social interaction: multi-model conversation with social agents. In *Proceedings of the National Joint Conference on Artificial Intelligence*, pages 22–28, 1994.

[24] A. Newell. *Unified Theories of Cognition.* Harvard Univ. Press, Cambridge, Mass., 1990.

[25] K. Pimentel and K. Teixeira. *Virtual reality: Through the new looking glass.* Windcrest/McGraw-Hill, Blue Ridge Summit, PA, 1994.

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

[27] A. S. Rao, A. Lucas, D. Morley, M. Selvestrel, and G. Murray. Agent-oriented architecture for air-combat simulation. Technical Report Technical Note 42, The Australian Artificial Intelligence Institute, 1993.

[28] A. S. Rao and G. Murray. Multi-agent mental-state recognition and its application to air-combat modelling. In *Proceedings of the Workshop on Distributed Artificial Intelligence (DAI-94).* Menlo Park, Calif.: AAAI press, Technical report WS-94-02, 1994.

[29] P. S. Rosenbloom, J. E. Laird, A. Newell, , and R. McCarl. A preliminary analysis of the soar architecture as a basis for general intelligence. *Artificial Intelligence*, 47(1-3):289–325, 1991.

[30] J. R. Searle. *Collective intention and action*, pages 401–415. MIT Press, Cambridge, MA, 1990.

[31] R. L. Shaw. *Fighter combat: tactics and maneuvers.* Naval Institute Press, Annapolis, Maryland, 1988.

[32] M. Stefik. *Introduction to Knowledge Systems.* Morgan Kaufmann, Palo Alto, CA, 1995.

[33] M. Tambe. Recursive agent and agent-group tracking in a real-time dynamic environment. In *Proceedings of the International Conference on Multi-agent systems (ICMAS)*, 1995.

[34] M. Tambe, W. L. Johnson, R. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1), Spring 1995.

[35] M. Tambe and P. S. Rosenbloom. RESC: An approach for real-time, dynamic agent tracking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[36] M. Tambe, K. Schwamb, and P. S. Rosenbloom. Building intelligent pilots for simulated rotary wing aircraft. In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, May 1995.

[37] B. Ward. *ET-Soar: Toward an ITS for Theory-Based Representations.* PhD thesis, School of Computer Science, Carnegie Mellon Univ., 1991.