Tracking Dynamic Team Activity

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

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.

Introduction

In many multi-agent domains, the interaction among intelligent agents — collaborative or non-collaborative — is both dynamic and real-time. For instance, in education, intelligent tutoring systems interact with students to provide real-time feedback(Anderson *et al.* 1990). In entertainment, projects such as virtual immersive environments(Maes *et al.* 1994) and virtual theatre(Hayes-Roth, Brownston, & Gen 1995) involve real-time and dynamic interaction. Similarly, in training, dynamic, real-time simulations — e.g., traffic or air-traffic control(Pimentel & Teixeira 1994) and combat(Rao & Murray 1994; Tambe *et al.* 1995) simulations — involve 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(Kuniyoshi *et al.* 1994), *RoboCup* soccer(Kitano *et al.* 1995).

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(Anderson *et al.* 1990; Rao 1994; Tambe & Rosenbloom 1995). 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 or risk. The key to this capability is tracking an agent's flexible and reactive behaviors in dynamic, multi-agent domains. This contrasts with previous work in the related area of plan recognition(Kautz & Allen 1986), which mostly focuses on 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 various 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(Grosz & Sidner 1990; Cohen & Levesque 1991). For instance, ordinary automobile traffic is not considered teamwork, despite the simultaneous activity, coordinated by traffic signs(Cohen & Levesque 1991). Teamwork involves team members' joint goals and joint intentions, i.e., joint commitments to joint activities(Cohen & Levesque 1991). Consequently, tracking teamwork as independent activities of individual members is difficult. Consider the example of two children collaboratively building a tower of blocks(Grosz & Sidner 1990) — they cannot be tracked as building two individual towers of blocks with gaps in just the right places. Similarly, in (RoboCup) soccer, the collaborative pass play of two attackers cannot be tracked as independent activities. Robotic collaboration by observation(Kuniyoshi *et al.* 1994) would also require tracking such joint activities.

Thus, team tracking raises the novel challenge of tracking a team's joint goals and intentions. Previous approaches(Anderson *et al.* 1990; Rao 1994; Tambe & Rosen-

bloom 1995), that focus on tracking individual agents, fail to track such team activities. One basic problem is inexpressiveness. 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 simply does not express a team's joint goal and activities. Furthermore, by failing to exploit such jointness, these approaches also fail to adequately meet the demands of dynamic, real-time domains (the focus of our current work). The main difficulty in real-time domains is that the tracker (tracking agent) has to resolve ambiguities in multiple team members' actions in real-time. Here, 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. Unfortunately, unable to exploit such jointness, the individual-oriented approaches engage in unconstrained search; this can be particularly problematic when tracking large teams. Finally, the above approaches also fail to address the dynamism in teamwork, particularly, dynamic formation and dissolution of subteams for different subtasks.

Some recent work has attempted to go beyond individuals and track multiple agents. One such approach tracks a group of agents engaged in identical activity(Tambe 1995). 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, for instance, this approach fails to track teamwork where agents engage in non-identical activities. An alternative approach(Rao & Murray 1994), although focused on multi-agent domains, tracks mental states of individual agents rather than joint mental states of teams. As shown later, such individual oriented approaches are found lacking in both solution quality (due to lack of expressiveness) and efficiency (due to lack of jointness) when tracking teams.

As a concrete basis for the above discussion, consider the following example of real-world teamwork — a typical simulated air-combat scenario (Figure 1), from a real-world combat simulation environment(Calder *et al.* 1993). 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(Shaw 1988). 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.

By correctly tracking the pincer **D** effectively counteracts it — it turns away from the center (Figure 1-b). Although this puts the second subteam (**L** and **M**) outside of **D**'s radar sight, recognizing the pincer also enables **D** to anticipate this second subteam's possible actions. 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.

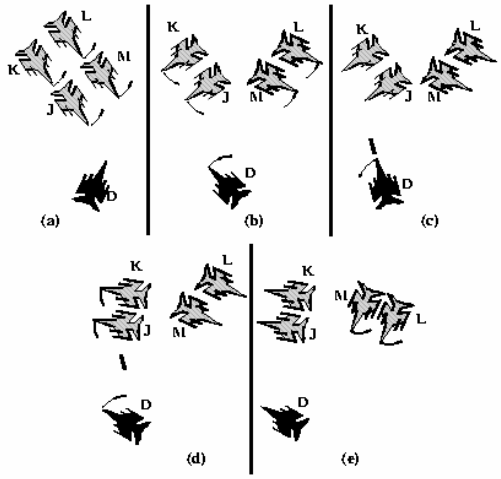


Figure 1: Simulated 1vs4 air-combat.

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. Meanwhile, 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 express 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 jointly executing a pincer. Yet, by focusing on individual models, **D** may be unable to express such jointness. In particular, **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?

The scenario also illustrates the difficulty in real-time ambiguity resolution. 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. In resolving such ambiguity, it is useful to exploit the team's jointness, i.e., to recognize that if one subteam is executing one half of the pincer, the other subteam must be executing the other half, and cannot be engaged in some unrelated activity.[1] Finally, the scenario also shows the dynamic formation of subteams and their sometimes

[1]This work will assume that subteams begin a joint activity together; over time though, a subteam may deviate.

unsychronized activities. Team members begin with almost identical activities(Figure 1-a), then dynamically split into subteams to begin a 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 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(Cohen & Levesque 1991). 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 the RESC(Tambe & Rosenbloom 1995) approach for tracking individual agents. The new approach, RESC$_{team}$, is aimed at real-time, dynamic team tracking.

Before describing team models and RESC$_{team}$ in more detail, the following section first provides an overview of RESC. The description below assumes as a concrete basis, pilot agents based on the Soar architecture(Tambe *et al.* 1995). We assume some familiarity with Soar's problem solving, specifically, applying operators to states to reach a desired state(Newell 1990).

RESC: Tracking Individual Agents

The RESC (REal-time Situated Commitments) approach to agent tracking(Tambe & Rosenbloom 1995) builds on *model tracing*(Anderson *et al.* 1990). 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 a repair-based approach to constraint satisfaction(Minton *et al.* 1990)).

A second key technique in RESC leads to its situatedness, i.e., ability to track the trackee's dynamic behaviors. A key assumption here is that the tracker 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.

As a concrete example of RESC, consider **D**'s tracking of **J** in Figure 1-d, assuming **J** is the *only* opponent present. To illustrate architectural reuse for 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, **D** selects the *intercept* operator in a subgoal 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*$_{\mathbf{D}}$. Operator$_{\mathbf{D}}$ will denote an arbitrary operator in **D**'s operator hierarchy. State$_{\mathbf{D}}$ will denote **D**'s state. Together, state$_{\mathbf{D}}$ and the operator$_{\mathbf{D}}$ hierarchy constitute **D**'s model of its present dynamic self, referred to as model$_{\mathbf{D}}$.

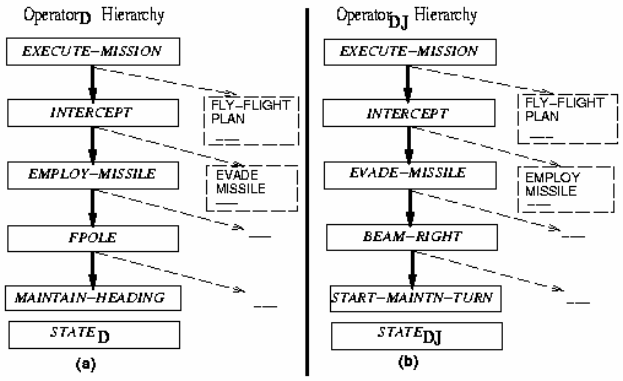


Figure 2: (a) Model$_{\mathbf{D}}$; (b) Model$_{\mathbf{DJ}}$. Dashed lines are unselected alternative operators. Model$_{\mathbf{D}}$ and Model$_{\mathbf{DJ}}$ need not be identical.

To reuse own architecture for tracking, **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 operator$_{\mathbf{DJ}}$ hierarchy and state$_{\mathbf{DJ}}$ constitute **D**'s model of **J** or model$_{\mathbf{DJ}}$, used to track **J**'s behavior. For instance, in the final subgoal, **D** applies the *start-&-maintain-turn*$_{\mathbf{DJ}}$ operator, which predicts **J**'s action. Thus,if **J** starts turning right towards beam, then there is a match with 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 operator$_{\mathbf{DJ}}$ hierarchy.

Such architectural reuse provides situatedness in RESC, e.g., 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(Hayes-Roth, Brownston, & Gen 1995; Rao 1994).) 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° beam turn or a 150° turn to run away. Yet, **D** commits to just one operator$_{DJ}$ hierarchy. This commitment may be inaccurate, resulting in a *match failure*, i.e., a difference between the model$_{DJ}$'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$_{DJ}$ hierarchy, within the context of the current continuously updated state. Thus, RESC attempts to generate a matching new operator$_{DJ}$ hierarchy without re-examining past states.

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; another approach may be to compute an average of individual attributes. While these approaches are desirable, in the absence of appropriate low-level sensors, their cost can be prohibitive. Therefore, the approach currently preferred in this work is to equate the divergent part to the state of a single *paradigmatic* (or representative) agent within the team, e.g., the team's orientation is the paradigmatic agent's orientation (as in (Tambe 1995)). 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 sub-teams, $\sigma_1...\sigma_N$, 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 (see the final section for further discussion 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 are the *roles*, which define activities that subteams 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. While the notion of roles has previously appeared in the context of teamwork(Kinny *et al.* 1992), it is exploited here via the specification of 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. 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 $pincer_{DT}$<LEFT,RIGHT>). Some abstract high level operators may require multiple definitions as they allow multiple role combinations. Such operators may essentially impose no role coherency constraints; hence their roles are not explicitly denoted.

Following is now the RESC$_{team}$ approach to team tracking:

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 RESC$_{team}$ always has a current best working hypothesis about the team activity — an anytime quality suitable for a real-time domain.

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. 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, which may not impose role-coherency constraints, are not susceptible to role failures). 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 the next Section).

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*$_{DT}$ denotes the operator that **D** uses to track $\mathcal{T}$'s joint mission execution. Since $\mathcal{T}$'s mission is not yet complete, **D** applies the *intercept*$_{DT}$ operator in the subgoal to track $\mathcal{T}$'s joint intercept. In the next subgoal, *employ-weapons*$_{DT}$ is applied. Following that, *get-firing-position*$_{DT}$ tracks **D**'s belief that $\mathcal{T}$ is attempting to get to a missile firing position, and so on. Each operator in this operator$_{DT}$ hierarchy indicates **D**'s model of $\mathcal{T}$'s joint commitment to that activity.

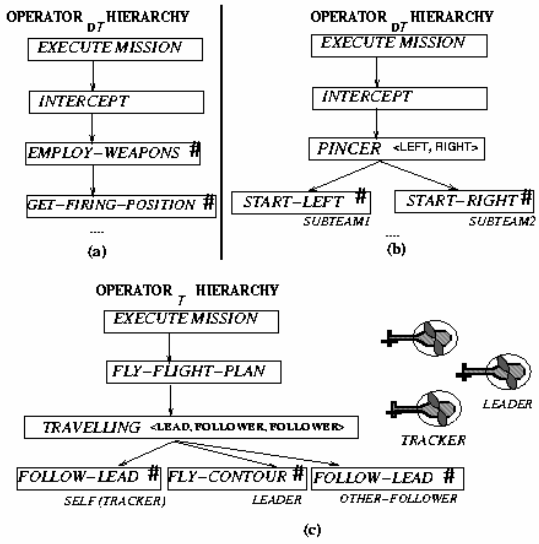


Figure 3: Team tracking via team operators. Some team operators require exactly one role. Rather than naming such a role, we denote it with a "#".

When the team in Figure 1-a splits into two subteams, role overload failure causes *employ-weapons*$_{DT}$ to fail. After current state backtracking, operator$_{DT}$ hierarchy in Figure 3-b results, which correctly tracks the on-going pincer. Here, *pincer*$_{DT}$<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 hierarchy 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(Tambe, Schwamb, & Rosenbloom 1995), again in the real-world combat simulation environment(Calder *et al.* 1993). 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 other branches to track teammates.

The Joint Intentions Framework

While team tracking has received little attention in the literature, researchers are investigating teamwork(Grosz & Sidner 1990; Cohen & Levesque 1991; Kinny *et al.* 1992). One leading theory of teamwork is the joint intentions framework(Cohen & Levesque 1991). 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 intentions framework; 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. 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 the next section).

Enhancements in Team Tracking

Efficient Role Assignments

While team operators do constrain tracking effort via 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 terms of multiple role combinations. Although in some such cases almost no role combinations may be disallowed, in other cases, there may be a fixed set of allowable 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 — and thus, real-time role assignment can be difficult.

To alleviate this inefficiency, any multiple definitions of a team operator, 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 constraint satisfaction problem (CSP)(Kumar 1992). 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 4 shows the role assignment for *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. Observations of subteam actions provide dynamic unary constraints.

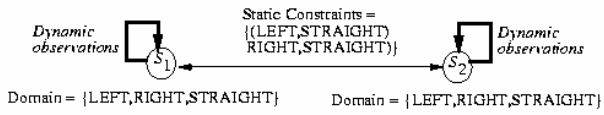


Figure 4: Detailed illustrative CSP (two variables).

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*(Kumar 1992) can accelerate the process of role assignment (or detecting failures). For instance, in Figure 4, 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 (up to four variables), and the absence of immediate observations. Additionally, independently testing the node consistency of variables 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 roles in a team operator are assigned to subteams via CSP (unless the role assignment is known). 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(Minton *et al.* 1990) — it commits to one assignment of roles to subteams, and dynamically repairs inconsistent assignments.

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. For example, in Figure 1-d, we assumed that a weakly adherent subteam 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 never deviate from its role, and thus there is an error in tracking — 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$_{team}$ uses 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. Repairing a single subteam's model (i.e., operators involving just a single subteam) within the team model is considered higher cost (given a possible reason for the subteam deviation). Repairing operators involving the entire team is considered to be even higher cost.

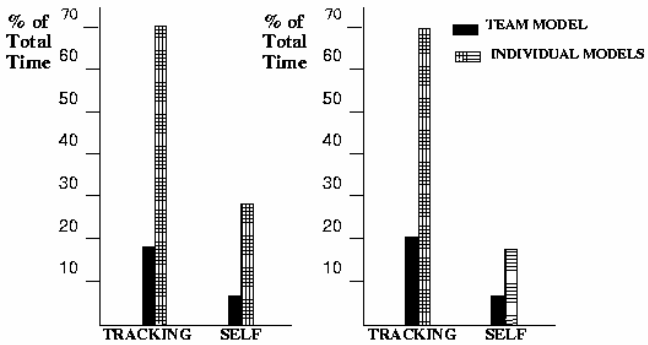
The tracker uses this minimal repair cost heuristic in attributing a degree of adherence to subteams. Thus, if a subteam is fulfilling its role despite events that could cause deviation from that role — e.g., if a missile has been fired at the subteam — RESC$_{team}$ attributes strong adherence to the subteam; this attribution suggests 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 CSP, this abandonment implies disabling outward constraint propagation from this subteam variable. Thus, this subteam is no longer assumed as part of the joint activity; however, the other subteam is assumed to continue. In contrast, if a subteam deviates *without* an explaining event, RESC$_{team}$ assumes team-wide error (applies normal CSP).

The pragmatic rationale behind the above heuristic is that it reduces real-time repair expense. Theoretically, it leans towards parsimonious explanations (it is based on minimality of fault models(Stefik 1995)). This heuristic is actually already incorporated into individual-agent RESC in that once committed to an interpretation, RESC avoids repairs until failure.

Implementation Results and Evaluation

We have implemented experimental variants of Soar-based pilot agents for both simulated fighters and helicopters. The original pilot agents have participated in various large-scale exercises, some involving expert human pilots(Tambe *et al.* 1995). Our experimental pilots (called pilot^{tracker}) incorporate RESC$_{team}$ (contain over 1000 rules). Promising results have led the team models to be ported to the original agents.

We have run the pilots^{tracker} agents in several combat simulation scenarios outlined by our human experts. Figure 5-a compares a fighter 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 5-a shows the percent of its total time that pilot^{tracker} spends in acting and tracking. Thus, when using team models in tracking, a 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, 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 contrast, pilot^{tracker} incorrectly readjusts its own maneuvers when using individual models; hence spends 28% of time deciding on its own actions. Figure 5-b provides similar comparative numbers for a team consisting of three opponents. (Thus, when using team models, a pilot^{tracker} spends 25% (18% TRACKING + 7% SELF) of time in mental activity, and the rest it waits for its maneuvers to complete. When using individual models, most of the time is spent tracking.)



(a) Four Agents in TEAM (b) Three Agents in TEAM

Figure 5: Comparing the efficiency of team models and individual models. Time measured in simulation cycles.

Focusing only on role-assignment, Table 1 presents the reduction in tracking effort due to team models and CSP (assuming four agents in each tactic). Column 1 names different tactics(Shaw 1988). Column 2 estimates the (worst-case) total number of operators searched assuming roles assigned to individual agents rather than subteams. Column 3 shows the factor reduction in the operators searched when role assignment is based on teams/subteams. The final column shows the *actual* results from the RESC$_{team}$ implementation, with operator role-unification and CSP. The reduction in tracking effort is substantial, and it will only grow with increasing numbers of agents.

Tactic name	Num operators indivdul models	Reduction team model	Reduction team+CSP
Pincer	56	14	14
Half-pincer	112	14	22
Posthole	112	14	22
Pincer-trail	144	8	24

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

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(Kuniyoshi *et al.* 1994), RoboCup soccer(Kitano *et al.* 1995), virtual theatre(Hayes-Roth, Brownston, & Gen 1995), virtual battlefields(Tambe *et al.* 1995). If agents are to successfully inhabit such worlds, they must understand and track team activity. This paper has taken a step towards this goal and advanced the state of the art in agent tracking and plan recognition. 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 minimal cost repair criterion to constrain tracking search.

Team models are one of the first to practically instantiate the theoretical joint intentions framework. 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 synthetic yet real-world environment: (i) a collaborative task involving simulated helicopters; and (ii) an adversarial task involving fighter combat.

One concern raised in generalizing this work to other domains is detecting that observed agents actually form a team. Currently, the team is either known in advance (e.g., helicopters) or it is detected based on agents' proximity to each other (e.g., pilot agents conclude that enemy fighters within 3-4 kilometers form a team). It appears that in many domains, teams are indeed known in advance (e.g., RoboCup Soccer or collaboration by observation); nonetheless, future work will hopefully uncover some general heuristics for team detection. Another issue for future work is tracking (apparantely) ill-structured team activity. To this end, we are currently testing RESC$_{team}$ in RoboCup soccer simulation(Kitano *et al.* 1995).

Acknowledgement: I thank Randy Hill and Paul Rosenbloom for valuable comments; Ramesh Patil, Kevin Knight, Gal Kaminka and Micheal Wooldridge provided useful feedback. This research was supported as part of contract N66001-95-C-6013 from ARPA/ISO. Domain expertise was provided by Bob Richards and Dave Sullivan of BMH Inc.

References

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

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

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

Grosz, B. J., and Sidner, C. L. 1990. Plans for discourse. Cambridge, MA: MIT Press. 417–445.

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

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

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

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.*

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

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

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

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

Newell, A. 1990. *Unified Theories of Cognition.* Cambridge, Mass.: Harvard Univ. Press.

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

Rao, A. S., and Murray, G. 1994. 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.

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).*

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

Stefik, M. 1995. *Introduction to Knowledge Systems.* Palo Alto, CA: Morgan Kaufmann.

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

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).

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

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