AI Magazine Volume 20 Number 4 (1999) (© AAAI)

The Benefits of Arguing in a Team

Milind Tambe and Hyuckchul Jung

■ In a complex, dynamic multiagent setting, coherent team actions are often jeopardized by conflicts in agents' beliefs, plans, and actions. Despite the considerable progress in teamwork research, the challenge of intrateam conflict resolution has remained largely unaddressed. This article presents CONSA, a system we are developing to resolve conflicts using argumentation-based negotiations. CONSA focuses on exploiting the benefits of argumentation in a team setting. Thus, CONSA casts conflict resolution as a team problem, so that the recent advances in teamwork can be brought to bear during conflict resolution to improve argumentation flexibility. Furthermore, because teamwork conflicts sometimes involve past teamwork, teamwork models can be exploited to provide agents with reusable argumentation knowledge. Additionally, CONSA also includes argumentation strategies geared toward benefiting the team, rather than the individual, and techniques to reduce argumentation overhead.

Teamwork is a critical capability in a large number of multiagent applications, such as virtual environments for education and training (Tambe 1997), robotic teams (Kitano et al. 1997), and teams on the internet. In these applications, agents must act together despite the uncertainties of their complex dynamic environment. Considerable progress has indeed been made in teamwork research. For example, recent advances in teamwork models (Tambe 1997; Jennings 1995), which explicitly outline agents' commitments and responsibilities in teamwork, have significantly improved flexibility in teamwork coordination and communication. However, this research has so far not addressed the challenge of resolving conflicts within a team.

However, as agent applications advance to meet the requirements of scale and autonomy, interagent conflicts become increasingly inevitable. For example, while autonomously reacting to dynamic events, agents can unintentionally interfere in others' actions, or faulty sensors can provide them with conflicting information or lead them to conflicting inferences. Although such conflict resolution is difficult in general, it is even more problematic in teams if intrateam conflicts are not anticipated.

To address the problem of conflict resolution in team settings, we are building a system called CONSA (COllaborative negotiation system based on argumentation). In argumentation, agents negotiate by providing arguments (which can be justifications or elaborations) in support of their proposals to one another. CONSA builds on past work in argumentation (Kraus, Sycara, and Evenchik 1998; Parsons and Jennings 1996; Chu-Carroll and Carberry 1995), but our focus here is to exploit the benefits of argumentation in a team setting. Thus, given CONSA's roots in past teamwork research (Tambe 1997), a key idea is to cast conflict resolution as an explicit common team goal. As a result, the recent advances in teamwork models are brought to bear during conflict resolution, improving flexibility of agent behaviors during negotiations. For example, in dynamic environments, it's important to react to unanticipated events that can occur during negotiations. Thus, if a team member privately discovers an event that renders the current team conflict irrelevant, it will now react by appropriately informing its teammates. Additionally, with an explicit common team goal, novel argumentation strategies emerge; for example, agents might attempt to improve the quality of teammates' arguments. Furthermore, because team conflicts can be rooted in past teamwork, CONSA enables agents to argue effectively about teamwork by exploiting the teamwork models in a novel way, that is, not only as a guide to agent behavior during conflict resolution but as a source for reusable argumentation knowledge. Finally, CONSA is being built within existing agent teams in complex environments and has focused on practical issues, such as minimizing the resources consumed in negotiations.

Domains and Motivations

The motivation for current research on negotiation is based on our previous work in complex, multiagent domains such as real-world battlefield simulations (Tambe 1997). We have built different teams of synthetic pilot agents that participate in combat simulations in these environments. These pilot-agent teams include companies of attack helicopter pilots and divisions of transport and escort helicopter pilots. A second domain for our work is RoboCup (Kitano et al. 1997), where we have twice successfully participated in the RoboCup tournaments. These agent teams have been developed based on a teamwork model called STEAM (Tambe 1997). STEAM is based on the joint intentions (Cohen and Levesque 1991) and SHAREDPLANS (Grosz 1996) theories of teamwork but with practical extensions for monitoring and replanning as well as decision-theoretic communication selectivity. STEAM has provided significant teamwork flexibility in all these applications. However, STEAM does not address the problem of conflicts in agents' beliefs and relevant negotiations to resolve such conflicts, limiting teamwork flexibility in key instances. We describe here just a few key examples that outline some of the basic issues for collaborative negotiations:

The firing-position case: Individual pilots in a helicopter team typically attack the enemy from firing positions. These positions are planned by a commander agent, who ensures that they do not conflict; that is, the positions are planned to be at least 1 kilometer (km) apart from each other. However, despite careful planning, individual pilots can autonomously react to unexpected enemy vehicles and end up in conflicting positions (for example, much less than 1 km apart).

The proceed case: In planning the positions described previously, the commander pilot plans one position (for example, position to hide behind a small hill) for each team member and communicates it to the relevant team member by radio. In one run, a message was lost because of radio interference; that is, the commander thought the position was communicated, but a team member *M*1 never received it. Thus, when the commander asked the team to proceed because it believed all the positions were successfully communicated, there was a conflict with *M*1.

The enemy-position case: Two scout helicopter agents can have conflicting beliefs about the closest enemy unit seen. For example, one scout might report completion, but the second scout might see an even closer enemy unit than the one reported.

The ball-position case: In our player team in RoboCup soccer simulation, defenders inform each other if the ball is close by and, hence, a threat. However, the players' belief of the ball's threat can differ, leading them to have conflicting beliefs about whether the ball is a threat.

These examples illustrate some of the key issues we are investigating in team negotiations. First, conflicts can arise between two team members' local actions, as in the firing position case, where an agent's local reaction has led to conflicts with another agent's local actions. In contrast, in the remaining three cases, conflicts in agents' beliefs affect the team's joint action: to proceed, report enemy location, or defend against the opponent team. Second, conflicts can or cannot be related to past teamwork. Thus, although in the proceed case, conflicts are related to team members' past actions in teamwork, it is not true of the enemy-position and ball-position cases. Third, negotiations might need to be performed under real-time pressure, as in the ball-position case, where negotiation delays are highly detrimental to team performance. These and other issues in negotiations are highlighted in further detail in the following sections. In addressing these issues, we aim to avoid any specialized solutions and focus instead on a general approach that would be applicable to a wide variety of conflicts.

Teamwork Model: A Brief Overview

Before we discuss CONSA, it is useful to briefly overview teamwork models, particularly the STEAM (Tambe 1997) model, because it is the basis of our team implementations. STEAM consists of two components, both currently realized in the SOAR (Newell 1990) architecture. The first is an *enhanced agent architecture* with explicit representation of the team's joint intentions, mutual beliefs, and team goals. Figure 1 shows an operator hierarchy (that is, a reactive plan hierarchy) for a synthetic helicopter pilot developed using STEAM. Team operators (reactive team plans), which explicitly

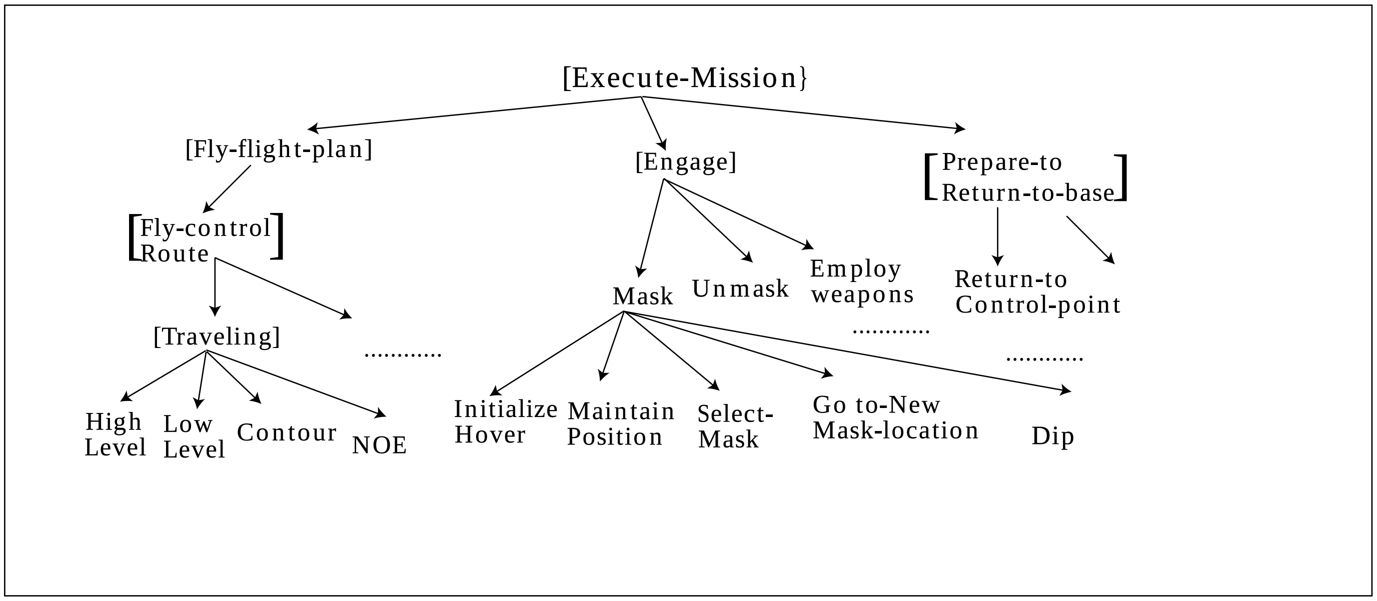


Figure 1. Portion of Pilot-Operator Hierarchy.

express a team's joint activities, are shown in [], such as [Engage]. At any time, one path through this hierarchy is active. This active hierarchy of operators is the team's joint intentions (team operators) and individual intentions (individual operators).

The second component of STEAM is the *domain-independent teamwork knowledge* that enables individual agents' flexible teamwork. Of particular importance here are two of the classes of domain-independent actions. The first is *coherence-preserving actions*, derived from the joint-intention theory (Cohen and Levesque 1991). These actions require agents to jointly activate and terminate team operators by establishing mutual beliefs in their initiation and termination; individual operators are executed without such mutual beliefs. The second class of domain-independent actions is *maintenance and repair actions* for replanning and team reorganization. These actions require an explicit specification of the dependency relationship of the joint intention on individual team members' activities, based on the notion of a role. A *role* constrains a team member *Mi* to some suboperator op_{Mi} of the team operator. Three primitive role relationships (and their combinations) can currently be specified in STEAM. An *And combination* implies that the achievement of a team operator requires achievement of each one of the roles. An *Or combination* requires success in at least one role for the team operator to be achieved. The role-dependency relationship states that an op_{Mi} depends on op_{Mj}.

Argument Representation and Evaluation

This section describes CONSA's underlying representation and algorithms to evaluate arguments, which are embedded in a larger CONSA process, discussed in the next section. CONSA's representation of arguments is based on Toulmin's (1958) argumentation pattern (henceforth TAP), chosen for its generality. In a TAP, an argument consists of the following elements: (1) *claim*, a conclusion whose merit an agent seeks to establish; (2) *data*, the facts that are a foundation for the claim; (3) *warrants*, the authority (for example, a rule) for taking the step from the data to the claim; and (4) *qualifications*, the degree of force that conferred on the claim based on the data and warrant.

In CONSA, claims are agents' individual or mutual beliefs. During argumentation, these claims form the proposals, with the supporting TAP as the argument for the proposal. For example, in RoboCup soccer, a claim (proposal) can be that "the ball is a threat," supported by data that "the ball is 30 meters from own goal," and a warrant that "if the soccer ball is within 35 meters of own goal, then it is very likely a threat." In CONSA, the data can itself be another claim (belief), with its own supporting TAP, so that a recursive tree of TAP structure can emerge in support of a claim. Finally, in CONSA, the qualifications on claims determine the strengths of arguments. Currently, claims have qualitative strengths: high, medium, and low. Thus, a strong warrant and strong data lead to a high strength for the claim.

```
Evaluate-proposal(Input: TAP-tree $\Theta$; Output: $\Omega$)

  1. In parallel, for all claims $\alpha_i$ in TAP-tree $\Theta$ do:
     (a)  { Check $\alpha_i$ for conflict with own claims;
            If $\alpha_i$ conflicts with own claim $\beta_j$, Compare-strengths($\alpha_i$, $\beta_j$,);
            If $\beta_j$, is stronger, add (reject($\alpha_i$), $\beta_j$)to $\Omega$ ;}
     (b)  { Check $\alpha_i$ for coincidence with own beliefs; If coincidence with own claim $\beta_i$
              { Compare-strengths(support($\alpha_i$), support($\beta_i$));
                If support($\beta_i$) is stronger, add (accept($\alpha_i$), support($\beta_i$)) to $\Omega$; } }

  2. Output $\Omega$; if $\Omega$ is empty, no conflicts or coincidence found.
```

Figure 2. A Simplified Version of CONSA's Algorithm for Evaluating a Proposal.

When an agent sends a proposal to its team, team members must determine if their own beliefs conflict with the proposal. Figure 2 presents a simplified version of CONSA's algorithm to make this determination. The input is a proposed TAP tree Θ, which forms the proposal (claim), with supporting arguments. The output is a set Ω of tuples ({reject(claim$_i$) or accept(claim$_i$)}, justification). Here, a *reject tuple* implies an agent's conflict with the claim $i \in \Theta$, but an *accept tuple* implies an improved justification in support of the claim. The justifications consist of TAPs. If Ω is empty, then no conflicts or improvements are found.

In the algorithm, step 1a checks the input TAP tree Θ for conflicts with the agent's own claims. If a conflict is found, strengths of the conflicting claims are compared, and the other agent's claim is rejected if its own claim is found stronger. Step 1b now compares the input claims from Θ for coincidence or agreement. If coincidence is found, then the supports of coincident claims are compared to determine the stronger support. If one is found, it is added to Ω. For expository purposes, two complicating factors addressed in CONSA are not shown here: First, CONSA can address the presence of multiple conflicts and coincidences. Second, when no coincidence or conflict is found in Θ itself, CONSA will not immediately accept Θ. Because leaf nodes in Θ can hold undesirable implications, CONSA derives implications from Θ. Although in general checking undesirable implications is difficult, CONSA currently executes one iteration of such derivations, checking for conflict or coincidence and adding the result to Ω.

To determine the strengths of the claims in the compare-strengths procedure in figure 2, CONSA relies on the supporting TAP structure. Given that the TAP structure can itself be recursive, claim strengths are evaluated recursively. For leaf-level claims, evidential rules are used. Here, CONSA exploits the benefits of argumentation in a team setting by relying on the following rules of evidence: Assertions from a team member regarding its own role and capability are judged to provide high-strength claims.

CONSA Approach

Figure 3 presents the overall CONSA negotiation process. Step 1 is a proposal generated by a team member. Steps 2, 3, and 4 are the opening, argumentation, and termination stages of CONSA's negotiation. In the opening stage, agents agree to jointly resolve the current conflict. In the argumentation stage, they cycle through proposals and counterproposals, terminating arguments in the termination phase.

Opening and Closing Stages

In CONSA's opening stage, the conflict-detection step (2a) requires it to address two different types of conflict. In particular, based on the description of the teamwork model (see Teamwork Model: A Brief Overview), conflicts can be of two types: (1) Team members can have conflicting beliefs about jointly initiating or terminating a team operator; for example, one agent believes the team operator must be terminated, but the other believes it cannot be

```
1. A team member $Mi$ generates a proposal $\alpha$.

2. Opening stage:
    (a) A team member $Mj$ detects a conflict with $\alpha$.
    (b) If $Mj$ believes joint action not beneficial to resolving conflict, terminate, return;
    (c) Else $Mj$ communicates with team members to establish team operator to resolve current conflict.

3. Argumentation stage
    (a) Any member $Mk$ in the current team operator may generate proposal to resolve conflict;
    (b) Other team members evaluate-proposal (see figure 2)
    (c) If no conflict/coincidence found, accept the proposal and go to step 4;
    (d) Else if proposal found to conflict/coincide; continue argument if cost-benefit-wise useful,
        else accept the proposal and goto step 4;

4. Closing stage
    (a) If suggested proposal accepted, then terminate conflict-resolution team operator;
    (b) Else if the conflict resolution found unachievable or irrelevant,
        terminate conflict-resolution team operator;
```

Figure 3. Three Stages of Argumentation in CONSA.

terminated. (2) Agents executing individual operators can unintentionally conflict with each other's role performance. Thus, in the examples from the section entitled Domains and Motivations, the firing-position case is a type-2 conflict, but the rest are type-1 conflicts. To detect a type-1 conflict, an agent must evaluate proposals sent by their teammates to jointly initiate or terminate team activities, detected by the evaluate-proposal algorithm in figure 2. In contrast, to detect type-2 conflicts, CONSA uses role constraints that explicitly specify the maintenance goals for the successful performance of the role. For example, in the firing-position case, the lateral range (distance) between *Mj* (the agent performing this role) and any other teammate must be at least 1 km.

Having detected a conflict in step 2a, we temporarily skip over step 2b to focus on step 2c. Here, a team member *Mj*, which has detected a conflict, initiates establishment of a team operator to resolve the current conflict. If the conflict is of type 1, *Mj* initiates the establishment of resolve-joint-conflict as a team operator, involving the entire team from the original joint activity. If the conflict is of type 2, *Mj* initiates the establishment of resolve-role-conflict as a team operator, but the involved team here is only *Mj* and the agent that caused a conflict for *Mj*'s role. For example, in the firing-position case, resolve-role-conflict is established as a team operator between *Mj* and *Mk* (the agent that caused the role conflict).

By casting conflict resolution itself as a team operator, all STEAM's flexible teamwork capabilities are brought to bear to guide agents' behavior during conflict resolution. For example, agents jointly establish the conflict-resolution team operators, using protocols that ensure synchronization and agreement among team members. In particular, teammates can disagree about the existence of the conflict, or they can be unable to negotiate if they are performing another higher-priority task. However, by using a team operator for conflict resolution, an agent *Mj* begins negotiations only after ensuring its teammates agree to, and are able to, engage in negotiations. Furthermore, STEAM's reasoning about commitments leads team members to behave responsibly toward each other. If a dynamic event causes a team member to privately discover that the conflict is resolved or unresolvable or irrelevant, it will be committed to make it mutually believed in the team. A team member cannot just on its own drop out from participation in the conflict resolution. The utility of such flexibility can be seen in the firing-position case. If a team member sees enemy vehicles approaching, it will terminate the current ongoing ne-

$\Rightarrow$ Mutually believed warrants: $\omega 1$, $\omega 2$, $\omega 3$, and $\omega 4$: $\neg$ Role-fulfilled(self)$\rightarrow$ $\neg$ All-roles-fulfilled(τ)

$\Rightarrow$ Commander pilot agent's initial claims: claim $\alpha 2$: All-roles-fulfilled(τ), claim $\gamma 1$: AND-combination(τ)

$\Rightarrow$ Pilot agent $M1$'s initial claims: claim $\beta 4$: $\neg$ Role-fulfilled (self), claim $\gamma 1$: AND-combination(τ)

Figure 4. Initial State: Commander Believes All-Roles-Fulfilled, M*1 Believes Own Role Not Fulfilled.*

gotiations but do so responsibly while it informs teammates of the situation.

Argumentation Stage

The *argumentation stage* involves an agent (sender) making a proposal to the agent-team (receiver) with an attached justification (argument). The receivers evaluate the proposal taking the justification into account and either accept or refute it. If refuting the proposal, a receiver can send back a counterproposal to the team, which can continue this cycle of proposals and counterproposals. Refutation can be done by rebutting or undercutting (Parsons and Jennings 1996). Briefly, rebutting refutes the teammate's claim (proposal) directly, with some justifications. In contrast, undercutting attacks the justification provided with the proposal rather than the proposal itself. In this argumentation stage, the teamwork setting provides two key novel ideas: First, it enables and requires a third strategy in addition to rebutting and undercutting, which we call *improve support*. In particular, an agent receiving a proposal from its team member can accept the proposal but might have a better justification for the proposal than the one offered by the sender. For example, in the enemy-position case, the second scout detected a closer enemy unit. The second scout agrees with the top-level claim that the scouting is completed, but it offers a higher-quality solution about the closer enemy unit, which allows the helicopter team's performance to improve. It is to enable this improve-support strategy that the evaluate-proposal algorithm (figure 2) checks for claim coincidence.

Second, teamwork models provide reusable argumentation knowledge. In particular, team conflicts are sometimes rooted in past teamwork, for example, in the proceed case. To argue effectively about teamwork, agents must be knowledgeable about teamwork. Here, STEAM provides general, reusable warrants for constructing TAPs. For example, the warrants shown here, extracted from STEAM's role relationships, are used in CONSA. Here, warrant $\omega 1$ states that if a team operator τ is an And combination, and all its roles are not achieved, then the team operator is not achieved. $\omega 2$ is a variation of an Or combination, and $\omega 3$ is that for an And combination.

$\omega 1$: Team-Operator(τ)
$\wedge$ AND-combination(τ)
$\wedge$ $\neg$ All-roles-fulfilled(τ) $\rightarrow$ $\neg$ achieved(τ)

$\omega 2$: Team-Operator(τ)
$\wedge$ OR-combination(τ)
$\wedge$ $\neg$ All-roles-unachievable(τ) $\rightarrow$
$\neg$ unachievable(τ)

$\omega 3$: Team-Operator(τ)
$\wedge$ AND-combination(τ)
$\wedge$ $\neg$ All-roles-fulfilled(τ) $\rightarrow$ $\neg$ achieved(τ)

Real-Time, Efficient Argumentation

Three techniques are used in CONSA to reduce resources used in argumentation and enhance its real-time performance (shown in steps 2b and 3d of figure 3). One technique is *decision-theoretic reasoning* of the cost-benefit analysis of argumentation. For example, in the ball-position case, the cost of arguing might outweigh the benefits (for example, the ball might be shot into the goal by the time the defenders complete their negotiations). Therefore, an agent will not negotiate with teammates even though it detects a conflict in the teammates' proposal. The second technique is *ordering of arguments*. If multiple arguments are applicable, CONSA will communicate the strongest first to speed the argumentation process. CONSA also uses pruning (see discussion later) to avoid communication of commonly held warrants.

Detailed Example of CONSA application

For a detailed example of CONSA's application, we take the simple proceed case we discussed earlier. Figure 4 shows the initial warrants and claims that are mutually known by the pilot agent team (of five agents). τ is the current team operator, an And combination. The initial proposal is generated by the commander agent (step 1 of figure 3) to suggest termination of the team operator τ. This proposal is $\alpha3 \Leftarrow \alpha2$, where $\alpha3$ is the claim "achieved(τ)," and $\Leftarrow$ stands for a justification.

*M*1 evaluates the proposal from the commander agent to detect conflicts (step 2a of figure 3). During this evaluation, using the evaluate-proposal algorithm from figure 2, no direct conflict or coincidence is found. However, deriving implication of $\alpha2$ leads to "Role-fulfilled(self)," which conflicts with $\beta4$, *M*1's own belief. However, $\beta4$ is evaluated to be stronger because *M*1 is an expert in its own role. *M*1 next uses $\omega4$ and $\omega1$ to construct an argument: $\neg\alpha3 \Leftarrow \neg \alpha2 \Leftarrow \beta4$. (Warrants $\omega1$ and $\omega4$ are pruned.) Essentially, *M*1 informs the commander agent that it disagrees that the team operator is achieved because its own role is not fulfilled. Because this conflict is of type 1, the argument from *M*1 is communicated to the entire team of pilot agents, which causes all members (including *M*1) to establish a team operator (resolve-joint-conflict); the team has thus entered the argumentation stage of CONSA. In this case, because $\beta4$ is in the area of expertise of *M*1, the commander (and other team members) evaluate $\beta4$ to have a high strength and accept it. They subsequently also accept $\neg \alpha3$ and $\neg \alpha2$ based on the support offered by $\beta4$. Thus, the proceed case is resolved by the commander accepting M1's assertion, and it communicates this acceptance to teammates.

Applying CONSA

CONSA is currently realized in the SOAR architecture in 109 rules. In the following discussion, we attempt a preliminary qualitative evaluation of CONSA. Our implementation has enabled agents to negotiate to resolve conflicts in the cases defined in Domains and Motivations in the following manner:

Firing-position case: An agent detects a conflict in its firing position because of its role-constraint violation (1-km lateral range). It then establishes a team operator (with the teammate that violates the role constraint) to resolve role conflict. It generates a proposal to suggest an equidistant move by each agent (500 meters) to meet the lateral range role constraint. This proposal is accepted by the second agent. (However, if the second agent cannot move, it rejects this proposal, causing the first agent to move 1 km on its own.)

Proceed case: As discussed previously, M1 persuades teammates that the current team activity is not achieved.

Enemy-position case: The second scout finds an *improve-support argument* to inform the team that it has better support (that is, a higher-quality solution), in the form of closer-range enemy that it spotted.

Ball-position case: As the cost of negotiation exceeds the likely benefits, agents avoid negotiations and act based on their own (divergent) beliefs.

We also attempted a preliminary test of CONSA's flexibility by creating some surprise variations of these cases.

proceed-1: The role relationship for the team operator τ was changed from an And combination to an Or combination. Here, despite team member *M*1's role not being fulfilled, *M*1 did not detect a conflict, and no arguments were generated. No conflict detection is correct because an Or combination does not require all roles to be fulfilled.

proceed-2: We gave one pilot agent (*M*1) two arguments to attack the commander's proposal—one based on its own role and one based on another teammate *M*3's role. Here, *M*1 correctly selected the stronger argument based on its own role to communicate first to the team.

firing-position-1: When the pilots established the resolve-role-conflict team operator to resolve firing-position conflicts, enemy vehicles were suddenly placed close by them. The pilot that noticed these vehicles first terminated the conflict-resolution team operator because it was irrelevant and informed its teammate.

firing-position-2: In a situation similar to the previous, we had one helicopter destroyed. The second terminated the negotiation because this team operator had become unachievable.

Related Work

Previous work in argumentation-based negotiation has often assumed noncooperative agents. For example, Kraus, Sycara, and Evenchik (1998) used several argument types borrowed from human argumentation in noncooperative situations, for example, threat, promise of a future reward, and appeal to self-interest. An example from Kraus, Sycara, and Evenchik (1998) is negotiation between two robots on Mars. Here, to persuade a robot *R*2, a robot *R*1 threatens it (*R*2) that *R*1 will break *R*2's camera lens or antenna if *R*2 does not comply. Such arguments appear inappropriate in team settings; for example, if *R*1 and *R*2 are a team and if *R*1 carries out its threat, then it will have a teammate (*R*2) without a lens or antenna. Other explicitly noncollaborative argumentation work appears in the legal domain, for example, DART (Freeman and Farley 1993), which is also based on Toulmin's representation schema. In contrast, Parsons and Jennings (1996) did not explicitly assume collaborativeness or noncollaborativeness in agents.

CONSA differs from this work in its explicit exploitation of the team setting in argumentation. As seen earlier, it exploits teamwork models to (1) guide flexible agent behavior in negotiation and (2) act as a source of reusable argumentation knowledge. It also adds argumentation strategies so agents can collaboratively improve each other's arguments. Also, CONSA includes techniques to avoid high overheads of negotiations. Chu-Carroll and Carberry's (1995) work in argumentation does assume collaborativeness on the part of the participating agents. Although they used SHAREDPLANS (Grosz 1996) in negotiations, they appeared to treat SHAREDPLANS as a data structure rather than a teamwork model. Thus, unlike CONSA, they do not use SHAREDPLANS either as a prescription for agents' behaviors in negotiations or as a source of reusable argumentation knowledge.

Summary and Future Work

Multiagent teamwork in diverse applications ranging from planning, design, education, and training faces the problems of conflicts in agents' beliefs, plans, and actions. Collaborative negotiation is, thus, a fundamental component of teamwork. We have begun to address this problem using an implemented system called CONSA for collaborative negotiation by argumentation. Although CONSA continues to build on previous work in argumentation, it exploits the benefits of a team setting with the following key ideas: First, CONSA casts conflict resolution as a team problem, bringing to bear some of the recent advances in flexible teamwork to improve the flexibility of agent behavior in conflict resolution. Second, because team conflicts are often about past teamwork, CONSA exploits teamwork models to provide agents with reusable argumentation knowledge. Third, CONSA focuses on collaborative argumentation strategies such as improve-support. Fourth, as an implemented system in a dynamic environment, CONSA uses a decision-theoretic approach, argument ordering, and pruning to reduce the cost of negotiation. Areas of future work include understanding CONSA's implications for argumentation in self-interested agents.

Acknowledgments

This research was sponsored in part by Air Force Office of Scientific Research contract F49620-97-1-0501 and in part by a subcontract from the Boeing Corp. We thank Zhun Qiu who implemented portions of the CONSA system described in this article.

References

Chu-Carroll, J., and Carberry, S. 1995. Generating Information-Sharing Subdialogues in Expert-User Consultation. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1243–1250. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Cohen, P. R., and Levesque, H. J. 1991. Teamwork. *Nous* 25(4): 487–512.

Freeman, K., and Farley, A. 1993. Toward Formalizing Dialectical Argumentation. In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, 440–445. Saline, Mich.: Cognitive Science Society.

Grosz, B. 1996. Collaborating Systems. *AI Magazine* 17(2): 67–85.

Jennings, N. 1995. Controlling Cooperative Problem Solving in Industrial Multiagent Systems Using Joint Intentions. *Artificial Intelligence* 75(2): 195–240.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1997. RoboCup: The Robot World Cup Initiative. In Proceedings of the First International Conference on Autonomous Agents, 340–347. New York: Association for Computing Machinery.

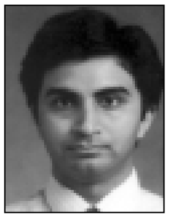
Kraus, S.; Sycara, K.; and Evenchik, A. 1998. Reaching Agreements through Argumentation: A Logical Model and Implementation. *Artificial Intelligence* 104(1–2):1–69.

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

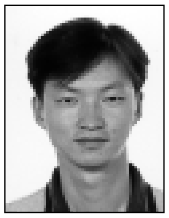
Parsons, Simon, and Jennings, Nicholas R. 1996. Negotiation through Argumentation—A Preliminary Report. In Proceedings of the International Conference on Multiagent Systems, 267–274. Washington, D.C.: IEEE Computer Society.

Tambe, M. 1997. Toward Flexible Teamwork. *Journal of Artificial Intelligence Research (JAIR)* 7: 83–124.

Toulmin, S. 1958. *The Uses of Argument.* London: Cambridge University Press.



Milind Tambe is a project leader for the Information Sciences Institute at the University of Southern California (USC) and a research assistant professor with the Computer Science Department at USC. He received his Ph.D. in 1991 from the School of Computer Science at Carnegie Mellon University. His interests are in the area of multiagent systems, specifically multiagent teamwork, learning, negotiation, agent modeling, and real-time performance, and he has published extensively in these areas. He is currently on the editorial board of the *Journal of Artificial Intelligence Research* and *Autonomous Agents and Multiagent Systems.* He is the program cochair for the International Conference on Multiagent Systems 2000 and has served as finance and local arrangements chair for the International Conference on Autonomous Agents and the senior program committee member for the American Association for Artificial Intelligence conference. His e-mail address is tambe@isi.edu.



Hyuckchul Jung received his Master's degree in computer science from Seoul University in Korea. He is currently a graduate research assistant at the Information Sciences Institute at the University of Southern California. His interest is in multiagent negotiation. His e-mail address is jungh@isi.edu.