Chapter 1

CONFLICTS IN AGENT TEAMS

Hyuckchul Jung and Milind Tambe
Information Sciences Institute, University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{*jungh,tambe*}*@isi.edu*

Abstract Multi-agent teamwork is a critical capability in a large number of applications. Yet, despite the considerable progress in teamwork research, the challenge of intra-team conflict resolution has remained largely unaddressed. This chapter presents a system called CONSA, to resolve conflicts using argumentation-based negotiations. The key insight in CONSA(*COllaborative Negotiation System based on Argumentation*) is to fully exploit 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 fully brought to bear during conflict resolution to improve argumentation flexibility. Furthermore, since teamwork conflicts often involve past teamwork, recently developed teamwork models can be exploited to provide agents with reusable argumentation knowledge. Additionally, CONSA also includes argumentation strategies geared towards benefiting the team rather than the individual, and techniques to reduce argumentation overhead. We present detailed algorithms used in CONSA and shows a detailed trace from CONSA's implementations.

1. INTRODUCTION

Teamwork is a critical capability in a large number of multi-agent applications, such as virtual environments for education and training[12], robotic teams[7] 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 instance, recent advances in teamwork models[6, 12], 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.

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

This chapter focuses on a system we have developed to resolve conflicts in agent teams, called CONSA: *COllaborative Negotiation System based on Argumentation*. In argumentation, agents negotiate by providing arguments (which may be justifications or elaborations) in support of their proposals to one another. CONSA builds on past work in argumentation[2, 8, 10, 11], but advances the state of the art by fully exploiting the benefits of argumentation in a team setting. Thus, one key idea in CONSA 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 negotiation flexibility. For instance, if a team member privately discovers an event that renders the current team conflict irrelevant, it will be committed to informing its team members — it will not just withdraw privately from negotiations. Additionally, with an explicit common team goal, novel argumentation strategies emerge, e.g., agents may attempt to improve the quality of teammates' arguments. Furthermore, since team conflicts are often rooted in past teamwork, CONSA enables agents to argue effectively about teamwork, by exploiting the teamwork models in a novel way, i.e., not only as a guide to agent behavior during conflict resolution, but as a source for reusable argumentation knowledge. Finally, CONSA is integrated within existing agent teams in complex environments, and has focused on practical issues, such as minimizing the resources consumed in negotiations.

This chapter is organized as follows: Section 2. provides background and motivation. Section 3. provides details on STEAM which CONSA is based on. Section 4. describes the representation and evaluation of arguments in CONSA. Section 5. explains CONSA's novel argumentation approach in detail. Section 6. shows how CONSA's implementation works in a specific example. An earlier and shorter version of this chapter has appeared in [13]. This chapter presents significant additional details, missing in that work.

2. DOMAINS AND MOTIVATIONS

The motivation for current research on negotiation is based on our previous work in complex, multi-agent domains such as real-world battlefield simulations[12]. 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. The second domain is Robocup[7] where we have twice successfully participated in the RoboCup tournaments. These agent teams have been developed based on a teamwork model called STEAM [12]. STEAM is based on the joint intentions[3] and SharedPlans[5] 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 of these applications. Yet, 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, i.e., the positions are planned to be at least one kilometer apart from each other. However, despite careful planning, individual pilots may autonomously react to unexpected enemy vehicles, and end up in conflicting positions (e.g., much less than 1 km apart). Figure 1.1 is a snapshot of ModSAF(*Modular Semi-Automated Forces*)[1] simulator for pilot agents and illustrates this conflict case. Contour lines show the terrain of the virtual environment in the ModSAF simulator. To attack nearby enemy units(100A12, 100A11, etc.), two helicopter pilot agents, *cheetah424* and *cheetah425*, pop up from a masking position. Unfortunately, they are too close to each other. They need to negotiate to resolve their position conflict.

- The proceed case: In planning the positions described above, the commander pilot plans one position(e.g. position to hide behind a small hill) per team member, and communicates it to the relevant team member via radio. In one run, a message was lost due to radio interference, i.e., the commander thought the position was communicated, but a team member M1 never received it. Thus, when the commander asked the team to proceed because it believed all of the positions were successfully communicated, there was a conflict with M1.

- The enemy position case: Two scout helicopter agents may have conflicting beliefs about the closest enemy unit seen. For instance, one scout may report completion of scouting and the closest enemy unit seen as part of this report, while the second scout may 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 may differ, leading them to have conflicting beliefs about whether the ball is a threat.

In addressing such conflict resolution problems, our goal is to avoid any specialized solutions, and focus instead on a general approach that would be applicable to a wide variety of conflicts.

Figure 1.1 Snapshot of firing position case.

3. TEAMWORK MODEL

Before we discuss CONSA, it is useful to briefly overview teamwork models, particularly the STEAM[12] model, since it is the basis of our team implementations. STEAM consists of two components, both currently realized in the Soar[9] architecture. The first is an enhanced agent architecture with explicit representation of team's joint intentions, mutual beliefs and team goals. Figure 1.2 shows an operator hierarchy (i.e., a reactive plan hierarchy) for a synthetic helicopter pilot developed using STEAM. Team operators (reactive team plans), which explicitly 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 to enable individual agents' flexible teamwork. Of particular importance here are two of the classes of domain-independent actions. The

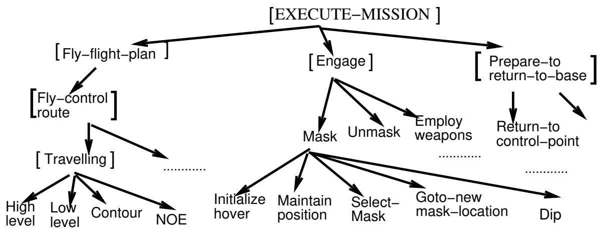


Figure 1.2 Portion of pilot operator hierarchy.

first is coherence-preserving actions, derived from the joint intention theory [3]. These 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 re-planning 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 [OP]. Three primitive role-relationships (and their combinations) can currently be specified in STEAM. An AND-combination implies that the achievement of 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$_{Mj}$.

4. ARGUMENT REPRESENTATION & 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[14] argumentation pattern (henceforth TAP), chosen for its generality. In a TAP, an argument consists of the following elements: (i) *claim:* a conclusion whose merit an agent seeks to establish. (ii) *data:* the facts that are a foundation for the claim. (iii) *warrants:* the authority (e.g., a rule) for taking the step from the data to the claim. (iv) *qualifications:* degree of force which 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) may 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 may itself be another claim (belief), with its own supporting TAP, so that a recursive tree of TAP structure may 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 data will lead to a "high" strength for the claim.

When an agent sends a proposal to its team, team members must determine if their own beliefs conflict with the proposal. Figure 1.3 presents 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$_i \in \Theta$, while 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 1 checks the input TAP tree Θ for conflicts with the agent's own claims. If a conflict is found, step 2 compares the strengths of the conflicting claims, rejecting the other agent's claim if own claim is found stronger. Step 3 now compares the input claims from Θ for coincidence or agreement. For simplicity, this algorithm assumes a single coincidence. 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. When no coincidence or conflict is found in Θ itself, CONSA will not immediately accept Θ. Since leaf nodes in Θ may hold undesirable implications, CONSA derives implications from Θ(step 4). While 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 claim in the *compare-strengths* procedure in Figure 1.3, 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.

5. CONSA APPROACH

Figure 1.4 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 counter-proposals, terminating arguments in the termination phase.

Evaluate-proposal(Input: TAP-tree Θ; Output: Ω)

1. In parallel, for all claims α_i in TAP-tree Θ do:
   { Check α_i for conflict with own claims;
   If α_i conflicts with own claim β_j,
   add tuple (α_i,β_j) to conflict-set CS; }
2. For all tuples (α_i,β_j) $\in$ CS, starting from tuple with lowest $\alpha_i \in \Theta$ do:
   { **Compare-strengths**(α_i,β_j);
   If β_j is stronger, add (reject(α_i),β_j) to Ω; }
3. In parallel, for all claims α_i in TAP-tree Θ do:
   { Check α_i for coincidence with own beliefs;
   If coincidence with own claim β_i,
   { **Compare-strengths**(support(α_i), support(β_i));
   If support(β_i) is stronger,
   add (accept(α_i,support(β_i)) to Ω; } }
4. If Ω is empty, check derivations of leaf claims in Θ;
5. Output Ω; if Ω is empty, no conflicts or coincidence found.

Figure 1.3 CONSA's algorithm for evaluating proposal.

Opening and closing stages: In CONSA's opening stage, the conflict detection step (2-a) requires it to address two different types of conflicts. In particular, based on the description of the teamwork model (Section 3.), conflicts can be of two types: (1) Team members may have conflicting beliefs about jointly initiating or terminating a team operator, e.g., one agent believes the team operator must be terminated, while the other believes it cannot be terminated; or (2) Agents executing individual operators may unintentionally conflict with each other's role performance. Thus, in the examples from Section 2., 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 via the **Evaluate-proposal** algorithm in Figure 1.3. 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 instance, in the firing position case, the *lateral-range* (distance) between Mj (the agent performing this role) and any other teammate must be at least one kilometer.

Having detected a conflict in Step 2-a, we temporarily skip over step 2-b to focus on step 2-c. Here, a team member Mj, who 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

1. A team member Mi generates a proposal α.
2. *Opening stage:*
   (a) A team member Mj detects a conflict with α.
   (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 1.3).
   (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 1.4 Three stages of argumentation in CONSA.

a conflict for Mj's role. For instance, 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 of STEAM's flexible teamwork capabilities are brought to bear, to guide agents' behavior during conflict resolution. For instance, agents jointly establish the conflict-resolution team operators, using protocols that ensure synchronization and agreement among team members. In particular, teammates may disagree about the existence of the conflict, or they may 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 towards each other. If a dynamic event causes a team member to privately discover that their conflict is resolved or unresolvable or irrelevant, it will be committed to make this 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 on-going negotiations, but do so responsibly while informing 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 may send back a counter-proposal to the team, who may continue this cycle of proposals and counter-proposals. Refutation may be done via rebutting or undercutting[10]. 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 may accept the proposal, but may have a better justification for the proposal than the one offered by the sender. For instance, in the "enemy position" case from Section 2., 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 1.3) checks for claim coincidence.

Second, teamwork models provide reusable argumentation knowledge. In particular, team conflicts are sometimes rooted in past teamwork, as for instance 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 instance, the warrants shown below, extracted from STEAM's role relationships, are employed in CONSA. Here, warrant $\omega 1$ states that if a team operator τ is an AND-combination, and all of its roles are not achieved, then the team operator is not achieved. $\omega 2$ is a variation for 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) $\wedge$ All-roles-fulfilled(τ) $\rightarrow$ achieved(τ)

Real-time, efficient argumentation: There are three techniques used in CONSA to reduce resources utilized in argumentation and enhance its real-time performance (shown in steps 2-c and 3-d of Figure 1.4). One technique is decision-theoretic reasoning of the cost-benefit analysis of argumentation, i.e., to avoid continuing to argue if the costs outweigh the likely benefits. Specifically, prior to initiating each proposal, an agent weighs the choices of

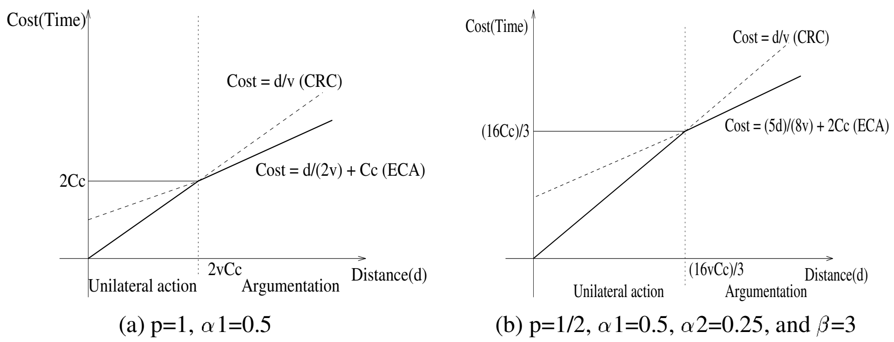


Figure 1.5 Threshold between unilateral action and argumentation

(i) continuing argumentation with teammates to optimally settle the conflict, or (ii) taking possible unilateral action to resolve the conflict. Agents continue the argumentation process only if the expected utility of argumentation is greater. For instance, in the firing position case, helicopters may not be 1 kilometer apart as desired. Given this conflict, an agent has the choice of initiating argumentation to determine the distance "n" each helicopter must move to minimize travel time (example discussed below in detail), or else to just moving a greater distance "$2n$" by itself to resolve the conflict.

Suppose the cost of the unilateral action choice (e.g., moving "$2n$") is CRC - the conflict resolution cost(this cost is the joint cost to the collaborating agents). In contrast, if an agent were to rely on argumentation, several outcomes are possible. First, if a teammate accepts the agent's proposal in the first cycle, the cost will be (1-α1)*CRC + Cc (α1: expected savings ratio for the collaborating agents together with argumentation after one cycle, Cc: communication cost). Second, if the proposal is not accepted at the first cycle and there are many cycles of sending proposals and counter proposals, it will cost (1-α2)*CRC + β*Cc (α2: expected savings ratio after multiple cycles, β: the expected number of cycles to settle the current conflict). With the probability(p) of teammate's accepting a proposal in one cycle, the expected cost of argumentation(ECA) is p*((1-α1)*$CRC + Cc) + (1-p)$*((1-α2)*CRC + β*Cc). Thus, an agent enters argumentation if $ECA < CRC$. Going back to the firing position case, the cost is time to resolve the conflict. With the unilateral action choice (ii), one pilot agent moves all the distance to resolve the constraint. In this case, the cost(CRC) is d/v (d: distance to be moved, v: helicopter's velocity). Now, to evaluate the argumentation option (i), if the agent assumes that there is no restriction for both agents' movements and expects that the other agent can accept its proposal at the first cycle with the probability(p) of 1, then it suggests 50:50 movements to minimize the time. This makes α1 equals 0.5. Figure 1.5 (a) shows that, with these p and

$\alpha 1$, an agent should argue if $d > 2 * v * Cc$. If the helicopter's velocity is 50m/sec and Cc is 5sec, agents enter into argumentation only if the distance to move is greater than 500m. Otherwise, it avoids argumentation and moves the distance by itself. On the other hand, if agents are not sure of the restrictions of the other agent, agents cannot expect that the other agent will accept its proposal at the first cycle. When an agent suggests 50:50 movements, there can be two possibilities. One possibility is that the other agent accepts the proposal at the first cycle and both agents move each half of the distance to be moved, which makes $\alpha 1$ 0.5. The other one is that an agent cannot accept its teammate's proposal and additional negotiation cycles are required. In the latter case, if three cycles are usually expected before reaching an agreement and the agreement is likely to be 75:25 movements because of the restriction of the other agent, then $\alpha 2$ is 0.25(one agent moves 75% of the distance and its time to move comes to be a team's cost) and β is 3. If an agent assumes that the probability(p) of the other agent's accepting its proposal in the first cycle is 0.5, the expected utility of argumentation is $(5 * d)/(8 * v) + 2Cc$. Therefore, agents enter into argumentation only if $d > (16 * v * Cc)/3$ (Figure 1.5 (b)). During the argumentation, Such evaluation continues at each cycle.

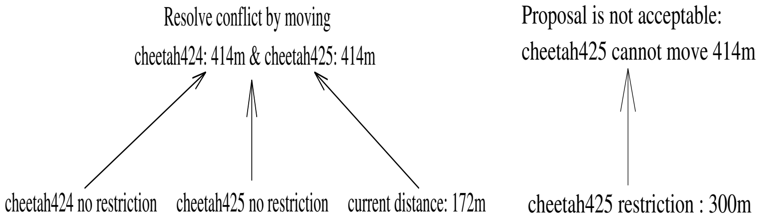
The second technique is ordering of arguments. If there are multiple arguments applicable, CONSA will communicate the strongest first, in order to speed up the argumentation process. Here the strength of the arguments is compared based on the **compare-strengths** procedure discussed in Figure 1.3. For instance, an agent will prefer an argument based on own role performance over one based on a teammate's role performance, since own role-performance provides a stronger argument given role expertise heuristic discussed earlier. Interestingly, this heuristic contrasts with [8], where in a non-collaborative setting, the weaker arguments are presented first to wear down an opponent. Ordering will also choose the highest claim in the proposed argument that can be attacked.

To reduce communication overhead during argumentation, CONSA also uses pruning to avoid communication of commonly held warrants. In particular, if an agent believes that particular warrants are mutually believed, and uniquely applicable in a given circumstance, it will prune the warrant from the communicated TAP tree.

6. DETAILED EXAMPLE OF IMPLEMENTATION

CONSA is currently realized in the Soar[9] architecture in 132 rules. Our implementation has enabled agents to negotiate to resolve conflicts in all the cases from Section 2.. Table 1.1 shows a part of the Soar outputs from CONSA's execution trace in the "firing position case". Here, *cheetah424* and *cheetah 425* are names of two pilot agents, and the number(e.g. "2288") in each line is the decision cycle number in Soar. "S114", "S124" etc. are the Soar states which represent a problem solving situation. "O264", "O267" etc. are the executed SOAR operators which make changes to the states. For instance, at decision cycle 2288, *cheetah424* executes an operator "O264" to resolve the firing position conflict: executing the operator enables *cheetah424* to start the argumentation processes with *cheetah425*. *Cheetah424* and *cheetah425* belong to a team of five pilot agents. Since the firing position conflict is a type 2 conflict (in Section 5.) between *cheetah424* and *cheetah425*, the other agents are not involved in this argumentation.

While executing operator "O264" at decision cycle 2288, *cheetah424* does decision theoretic reasoning (in Section 5.). Since it assumes that both agents have no restriction for their movements, it concludes that it is beneficial to share the distance to be moved through argumentation. At this step, if the communication cost is too high, *cheetah424* avoids argumentation and decides to take unilateral action. At decision cycle 2290 of *cheetah424* and 2289 of *cheetah425*, by executing the operator "establish-jpg", both agents agree to jointly activate and terminate a team operator "resolve-role-conflict". This process is supported by STEAM[12]. Next, *cheetah424* computes both agents' movements based on its beliefs for both of them. Based on the current distance and the assumption of no restriction, *cheetah424* sends a proposal suggesting 50:50 movements with justifications at decision cycle 2362. Figure 1.6 (a) shows the tree structure of the proposal. Leaf nodes are the justifications.



(a) *cheetah424*'s proposal (b) *cheetah425*'s counter-proposal

Figure 1.6 Structure of proposal with justifications

Cheetah425 receives the proposal and starts to check whether the claims in the justifications are consistent with its own beliefs or not by executing "inconsist-control" operator at decision cycle 2366. Let's assume that *cheetah425* cannot move more than 300 meters because of close enemy units. At decision cycle 2369, *cheetah425* finds a conflict between its belief(movement restriction of 300 meters) and *cheetah424*'s belief(middle leaf node in Figure 1.6 (a)). *Cheetah425* decides to reject the proposal by comparing strengths of both beliefs at decision cycle 2369 and 2371. Algorithm in Figure 1.3 explains

Table 1.1 Traces of firing position case.

cheetah424	*cheetah425*
...	...
Position conflict detected.	Position conflict detected.
...	...
2288:O: O264 resolve-role-conflict	2287:O: O403 resolve-role-conflict
Decision theoretic reasoning.	...
2289:==>S: S114 (operator no-change)	2288:==>S: S124 (operator no-change)
2290: O: O267 establish-jpg	2289: O: O406 establish-jpg
...	...
Established JPG.	Established JPG.
...	...
2362: O: O273 send-proposal	...
Movements cheetah424:414&cheetah425:414	...
...	Message from cheetah424.
...	2366: O: O413 inconsist-control
...	Begin inconsistency checking.
...	2367: ==>S: S129 (operator no-change)
...	2368: O: O415 make-decision
...	2369: O: O416 compare-claims
2368: O: O218 wait	Compare the claims in arguments.
...	2370: ==>S: S130 (operator tie)
...	2371: O: O417 evaluate
...	Evaluating the claims.
...	Reject the proposal.
...	...
...	2372: O: O432 send-counter-proposal
...	Cheetah425 restriction: 300 meters.
Message from cheetah 425.	...
2373: O: O282 inconsist-control	...
Begin inconsistency checking.	...
2374: ==>S: S119 (operator no-change)	...
2375: O: O285 make-decision	...
2376: O: O286 compare-claim	...
Compare the claims in arguments.	...
2377: O: O287 evaluate	2380: O: O414 inconsist-wait
Evaluating the claims.	...
Accept the counter-proposal.	...
Belief change: cheetah 425's free-move 300	...
Decision theoretic reasoning.	...
2378: O: O289 send-proposal	...
Movements cheetah424:528&cheetah425:300	...
...	Message from cheetah424.
...	2388: O: O416 compare-claims
...	Compare the claims in arguments.
...	2389: O: O417 evaluate
...	Evaluating the claims.
...	Accept the proposal.
...	...

this evaluation process. *Cheetah425* sends a counter-proposal of disagreement with justification(leaf node of Figure 1.6 (b)).

At decision cycle 2377, after receiving the counter-proposal, *cheetah424* evaluates it and changes its belief of the movement restriction for *cheetah425*. *Cheetah424* uses decision theoretic reasoning again before sending another proposal. If *cheetah425* can only move small distance(e.g., 100 meters), there is no benefit from argumentation and *cheetah424* takes unilateral action without argumentation. However, in this trace, *cheetah424* decides to start another cycle of argumentation. At decision cycle 2378, *Cheetah424* sends a counter-counter-proposal of 62:38 movements and *cheetah425* accepts the proposal at decision cycle 2389. Now, both agents make the agreed movements and terminate the jointly committed team operator “resolve-role-conflict”.

7. RELATED WORK

Previous work in argumentation-based negotiation has often assumed non-cooperative agents. For instance, [8, 11] uses several argument types borrowed from human argumentation in non-cooperative situations, e.g., threat, promise of a future reward, and appeal to self interest. An example from [8] is negotiation among two robots on Mars. Here, to persuade a robot R2, a robot R1 threatens it (R2) that R1 will break R2’s camera lens or antenna, if R2 does not comply. Such arguments appear inappropriate in team settings, e.g., if R1 and R2 are a team, and if R1 carries out its threat, then it will have a teammate (R2) without a lens or antenna. Other explicitly non-collaborative argumentation work appears in the legal domain, e.g., DART[4], which is also based on Toulmin’s representation schema. In contrast, [10] does not explicitly assume collaborativeness or non-collaborativeness in agents.

CONSA differs from this work in its explicit exploitation of the team setting in argumentation. As seen earlier, it exploits teamwork models: (i) to guide flexible agent behavior in negotiation and (ii) 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 work in argumentation does assume collaborativeness on part of the participating agents[2]. While they use SharedPlans [5] in negotiations, they appear to treat SharedPlans as a data-structure, rather than a teamwork model. Thus, unlike CONSA, they do not use SharedPlans either for prescribing agents’ behaviors in negotiations, or as source of reusable argumentation knowledge.

8. SUMMARY AND FUTURE WORK

Multi-agent 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. To address the problem, this chapter describes an implemented system called CONSA for collaborative negotiation via argumentation. While CONSA builds on previous work in argumentation, it advances the state of the art via the following key ideas: (i) 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; (ii) Since team conflicts are often about past teamwork, CONSA exploits teamwork models to provide agents with reusable argumentation knowledge; (iii) CONSA focuses on collaborative argumentation strategies such as improve-support; (iv) As an implemented system in a dynamic environment, CONSA uses a decision theoretic approach, argument ordering and pruning to reduce the cost of negotiation. We have presented detailed algorithms and initial results from CONSA's implementation. Areas of future work include understanding CONSA's implications for argumentation in self-interested agents.

9. ACKNOWLEDGEMENTS

This research is funded by DARPA ITO award number F30602-99-2-0507.

References

[1] 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.

[2] J. Chu-Carroll and S. Carberry. Generating information sharing subdialogues in expert-user consulation. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1995.

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

[4] K. Freeman and A. Farley. Towards formalizing dialectical argumentation. In *Proceedings of the Conference on Cognitive Science Society*, 1993.

[5] B. Grosz. Collaborating systems. *AI magazine*, 17(2), 1996.

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

[7] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on autonomous agents*, 1997.

[8] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104:1–70, 1998.

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

[10] S. Parsons and N. R. Jennings. Negotiation through argumentation — a preliminary report. In *Proceedings of the International Conference on Multi-agent Systems*, 1996.

[11] K. Sycara. Persuasive argumentation in negotiation. *Theory and Decision*, 28(3), 1990.

[12] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.

[13] M. Tambe and H. Jung. The benefits of arguing in a team. *AI Magazine*, 20(4), 1999.

[14] S. Toulmin. *The uses of argument*. Cambridge Univ Press, London, 1958.