A Role-Based Decision-Mechanism for Teams of Reactive and Coordinating Agents ⋆

Silvia Coradeschi and Lars Karlsson

Department of Computer and Information Science
Linköping University, Sweden
silco@ida.liu.se, larka@ida.liu.se

Abstract. In this paper we present a system for developing autonomous agents that combine reactivity to an uncertain and rapidly changing environment with commitment to prespecified tactics involving coordination and team work. The concept of roles in a team is central to the paper. Agents act and coordinate with other agents depending on their roles. The decision-mechanism of an agent is specified in terms of prioritized rules organized in a decision tree. In the decision-tree both reactivity to changes in the environment and commitment to long term courses of actions (behaviors) are present. The coordination between agents is mainly obtained through common tactics, strategies, and observations of actions of team members, rather than explicit communication. Coordinated behaviors can be specified in terms of roles and are encoded simultaneously for all participating agents.

1 Introduction

An important issue in the development of multi-agent systems is how to create a decision mechanism for an agent that combines the ability to react to a dynamically changing environment and to coordinate with other agents. A special kind of coordination is the one required in performing team work. The importance to have the ability to operate in teams has been emphasized by several authors, see for example [11] and [2]. In a team, an agent needs special capabilities to coordinate with other agents in the team. In particular an agent acts and coordinates mostly depending on his roles in the team. The concept of role is therefore crucial in the development of agents working in teams. Roles have been considered in the context of the Belief, Desire, and Intentions Multi-Agent System [6], but mainly in the perspective of team formation and changes of roles in a team, [13] and [1]. In the case of modeling team and team tactics [14] just very simple cases have been examined (two types of roles statically assigned to the agent during the team formation). Our interest is in creating a general role-based decision-mechanism for multiple automated agents that act in complex and real-time environments in which team work is required.

Our multi-agent system has initially been developed in the context of a cooperation project with Saab Military Aircraft with the aim to investigate the design and implementation of automated agents for air combat simulation [4]. In particular the aim was to

⋆ To appear in "RoboCup-97: The First Robot World Cup Soccer Games and Conferences" Springer Verlag Lecture Notes in Artificial Intelligence (LNAI) 1998.

develop means for specifying the behavior for automated pilots. The goal was to combine on one side reactivity to the uncertain and dynamic battlefield environment and on the other side commitment to prespecified tactics involving coordination and team work. The behavior specification was not to be done by experts in computer science and AI, but mainly by experts in air combat and military air technology, and therefore, the system was aimed to be relatively easy to use and conceptually simple.

The system has proven to be easy to apply in other domains. We have been able to use it without any important modification in the soccer RoboCup simulated domain [3] and we intend to use some of the ideas of the system in a project for the design of an autonomous aircraft for traffic surveillance. In the following we use mainly examples from the soccer domain as most people have a better knowledge in this domain than in the air combat domain. Also in the TacAir-Soar project [12] the development of intelligent agents for the air combat domain is now coupled with the development of intelligent agents for the soccer domain. Special consideration has been given to team work, but the interest has been more centered in the tracking of teams [10] and commitment inside the team [11] than in the specification of agents through roles.

In both the soccer domain and the air combat domain, coordination between agents is essential, but it cannot be achieved by mainly communication and negotiation [8]. In fact, giving the real-time constrains in these domains, the time for communication is limited and the presence of hostile agents makes the communication risky and unreliable (jamming). Several techniques have been developed for communication-poor coordination including a physics oriented approach to coordination [9], focal points [5] and observation-based coordination [7]. In general these techniques aim at coordinating isolated agents performing rather simple tasks e.g. filling holes of various sizes in a planar surface, trying to choose the same object out of a set of objects and moving in the same location or in an opposite location with respect to another agent in a discrete time, two dimensional, simulated grid world. In our case the agents are organized in teams, have roles in the team and perform complex and in general not repetitive tasks.

What the agents do and how they coordinate with other agents is mostly determined by their roles in the team. An agent can have multiple roles, for example he can be captain of the team and midfielder. Roles can be specialized, for example a left-midfielder is a specialization of a midfielder. The decision-mechanism is specified in term of roles and the agents will execute the behaviors and actions that are proper to their roles. Specifying coordinated behavior requires to take in consideration all the roles involved in the coordination, but then for each agent the importance of the coordinated behaviors should be evaluated in the context of the roles of that agent. Therefore coordinated behavior is specified in decision-trees for multiple roles, section 6.1, and then the parts of the behavior related to a role are collected, added to the specification of the role and reconsidered in the context of the role. The coordination is then obtained with a totally distributed control.

2 Overview of the system

The basic element of the decision-mechanism in our system is the concept of *behavior module* (behavior for short). A behavior specifies what actions and subbehaviors should be executed and under what conditions to fulfill some specific aims or purposes. It can either be constantly present or it can be started under certain circumstances, go on for a period of time and then terminate. Behaviors have different complexities; they are used for specifying coordination and the behavior proper of a role, but also for specifying simple routine as for example the positioning in the penalty area of a soccer player. Behaviors can recursively contain behaviors. The same structure is therefore repeated at several levels of the decision mechanism. A behavior is characterized by a priority, which indicates the importance of the behavior, and a decision-tree in which is coded what the behavior does. A decision-tree is a tree of hierarchically structured and prioritized decision rules. Visiting the decision-tree several actions and behaviors can be selected as candidates to be performed. The selection of those that will actually execute is done based on priority values and compatibility. Behaviors and actions are compatible if they can be executed concurrently. The priority values can change dynamically as different circumstances can alter the importance of actions and behaviors during the simulation.

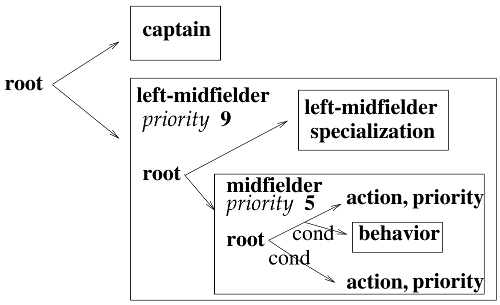

Fig. 1. Decision-tree of a left-midfielder that is also captain of the team

Each agent is equipped with a state describing the information that he uses for making decisions, and a decision-tree that specifies the general behavior of the agent (section 3). This decision-tree is the result of the composition of the behavior modules that specify the roles of the agent. The composition of behavior modules consists of creating a decision-tree with a module in each leaf and supplying a priority value for each behavior module. Conditions required to enter the behavior module can be specified, for example a corner-kick shooter behavior is entered if a corner kick is called. In Fig. 1 the decision-tree of a left-midfielder that is also captain of the team is sketched. The left-midfielder role is a specialization of the midfielder role and it is specified as a composition of a behavior module for the midfielder role and a behavior module for the left-midfielder specialization.

Coordinated behaviors are specified through decision-trees for multiple roles, section 6.1. These decision-trees have the same structure as the decision-tree of an agent, but each action and condition is associated with a specific role.

3 The decision-mechanism

A decision-tree consists of a hierarchy of decisions with one decision for each node. Two kinds of conditions can be associated with a node: conditions for entering the node and modifier conditions that change the priorities of actions and behaviors. Modifier conditions have the form *(condition* $\rightarrow$ *number)*. If the condition is true, the number is added to the priority of the actions or behaviors associated with the branches. The conditions are tested in the state of the agent. The state contains the characteristics of the agent specified before the start of the simulation, the information the agent receives directly from the simulator and derived information. In order of not being purely reactive, an agent should keep track of what has happened in the past. One part of the state represents therefore the "memory" of the agent. In the memory are recorded, among other things, important past events, decisions previously taken, and messages received via communication channels. Conditions can be primitive and composite. Composite conditions are formed by applying the operators *and3, or3, not3, for-all* and *there-is* to other composite or primitive conditions. *And3, or3, not3* are operators in a three-valued logic: true, false and unknown. The value of a condition in fact can be unknown, for example the agent may not know his relative position with respect to the ball.

At the leaves of the decision-tree there are actions and behaviors with a basic priority value. At each step of the simulation the state is updated with information that is received from the simulator and interpreted. Then the decision-tree is visited and as a result a list of actions and a list of behaviors are created. The actions in the list are the actions that the agent will consider to perform. To this list, the actions still in progress are added. The agent then selects the actions to actually execute and then sends them to the simulator or to update the state. Some of the actions in the list are mutually exclusive as they use the same resource and in this case the agent selects those he will actually perform depending on their priority values. Some actions can be done concurrently and in this case the agent performs them in parallel. The test to check if two actions are compatible is based on the physical resources they use. Two actions are compatible if they do not use the same resources. For example the action of passing is compatible with the action of sending a message. Operators for sequential and concurrent composition of actions are also defined. If the agent decides to perform sequences of actions he is not committed to complete the sequence of actions as the sequence can be interrupted if something else becomes more important.

Behaviors are selected for execution with a mechanism similar to the one for actions and the result of the selection is recorded in the state. The behaviors continue executing until they have fulfilled their aim, they have failed or they are interrupted to give space for something more important. If a behavior is executing then the tree inside the behavior is visited and the actions are selected in the same way as we have explained earlier in this section. The actions selected are then added to the list of actions of the decision-tree that directly contains the behavior. The selection mechanism is recursively applied until the actions of the main tree are selected and executed.

Let us consider a simple example of decision-tree (Fig. 2). This is part of the tree of an attacker. One branch leads to the action of going near the opponent the attacker is marking, in case the distance to the opponent is more then 30. The other branch is entered when a midfielder has the ball, *there-is x midfielder (has-ball x))*. If the attacker

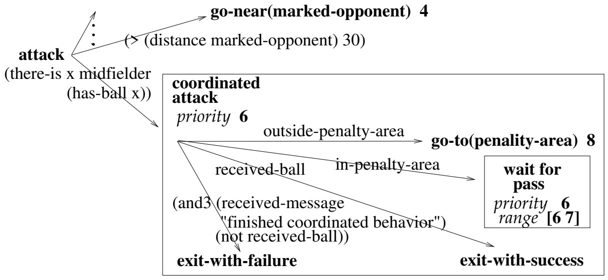

Fig. 2. Part of the decision-tree of an attacker

is outside the penalty area he goes to the penalty area, and when he is inside the penalty he waits for a pass. The decision-tree of the **wait for pass** behavior has been omitted due to lack of space. The behavior **coordinated-attack** is exited with success if the attacker receives the ball; with failure if a message from the midfielder is received that the coordinated attack is finished and the agent has not received the ball.

The conditions on the branches can refer to both information obtained from perception and information stored in the agent's internal state. An example of the latter is when a player follows an opponent due to a commitment to mark this opponent. This commitment is registered in the player's state. Making the commitment to mark a particular opponent constitute an internal action that update the state of the agent. The external actions presented in the decision tree are interpreted in terms of the low-level actions available in the simulator. For instance, **go-to(penalty-area)** makes the agent turn and move towards the penalty area. Also the information received from the simulator is interpreted. For instance, at each simulation cycle information about the ball and other players is received by the agent and using this information a list of team mates that are nearer than him to the ball is created. This permits the user to work on a level of abstraction that is more meaningful and convenient than working with the low-level data and actions of the simulator. He/she can focus more on what to do than the details of how to do it.

4 Behavior modules

Behavior modules play an important role in the system. They can be used to specify agents' roles and coordination, but also simple routines as **wait-for-pass** and in general, long term course of actions. An example of a behavior is **coordinated attack** in Fig.2. An important aspect that emerge in the use of behaviors is the need for modularity, that is the internal specification of the behavior should not be visible in the rest of the decision-tree of which it is part. In fact, roles and coordinated behaviors can be specified independently of the decision-tree of which they will be part. Coordinated behaviors can be specified for instance through decision-trees for multiple roles (see section 6.1). Also simple routines should be reusable in different decision-trees. On the other hand the actions selected in a behavior and specially their priorities have influence in the rest of the tree and in other behaviors. We consider these issues in the subsection *Priorities in behaviors*.

Complex behaviors can contain parts that should be executed in parallel or in sequence. In analogy with sequential and concurrent actions, sequential and concurrent behaviors are defined in the last part of this section.

Priorities in behaviors There are several possibilities in deciding the priorities of the actions selected in a behavior. The simpler method is to assign to the selected actions the priority of the behavior. This gives modularity in the assignment of priorities as the priorities inside the behavior are not visible externally. The problem with this approach is that not all the actions in a behavior have the same importance and a non-essential action in one behavior can be preferred over an essential action in another behavior and make this last behavior terminate. What we would like is that the priority that the action has inside the behavior would influence which priority the action will have when competing with the actions selected in the decision-tree containing the behavior. At the same time we would like to have an external control of the priorities of the actions coming from a behavior in order to prioritize correctly one behavior over another and to have more freedom in deciding the priorities inside a behavior. We define together with the priority of the behavior also a range inside which the priority of the actions selected in the behavior can vary. Given the priority that the selected action has inside the behavior, its external priority is the priority of the behavior increased or decreased in proportion to the internal priority of the action and the range in which the priority is allowed to vary. In the example in Fig.2 the action **go-to(penalty-area)** has internally priority 8 and externally priority 7. The action **go-near(x)**, where x is the opponent that the attacker is marking, has priority 4 and the range [5 7] of the coordinated attack behavior guarantees that the actions in the coordinated attack will always have higher priority than the **go-near(x)** action. This solution in most cases prevents that actions with lower priorities interfere with higher priority behaviors. However, it still may happen that actions external to a behavior interfere with the behavior, even if the behavior has higher priority than the actions. Let us consider for example the **wait-for-pass** behavior. In this behavior the agent mainly turns in the direction of the player with the ball to spot as soon as possible if the player is passing to him. Between one turning action and another, no actions are selected in the behavior. Therefore the system can decide to perform the action **go-near(x)**. That is of course not what we want. A solution we intend to consider is to allow a behavior to reserve a resource, in this case the moving resource. The resource can be released only if it is an action that has higher priority of the behavior that claims the resource.

Interruption and termination of behaviors Behaviors can terminate, successfully or unsuccessfully, or can be interrupted. When a behavior is selected the decision-tree inside the behavior is visited and the actions selected compete with the other actions selected in the visit of the decision-tree containing the behavior. Each behavior has exiting conditions that are used to exit the behavior when it is successfully terminated and when the behavior cannot be continued and terminates with failure. In the example in

Fig.2, the behavior is successfully terminated if the attacker receives the ball. A behavior can be interrupted if an incompatible behavior with higher priority is selected or if an action selected in the behavior cannot be executed as it is incompatible with another selected action with higher priority. In this last case the behavior can be exited or not depending on whether the action is essential for the behavior. If an action essential for the behavior cannot be performed, recovery actions to be performed before exiting the behavior can be specified. The criterion used for checking the compatibility of actions, looking at the resources they use, cannot be transferred directly to the case of behaviors. The compatibility of behaviors is in fact more conceptual than due to physical limitations. For example a coordinated attack from the left is incompatible with a coordinated attack from the right, but defending a part of the field by a defender can be compatible with making the team advance passing to the midfielders. The relation of compatibility between behaviors is decided when all the behaviors are created. When a new behavior is reached during the visit of the decision-tree, it competes with the other behaviors selected during the visit and with the behaviors currently executing. The compatible behaviors with higher priority will be the ones executing in the next step of the simulation. Exiting actions can be specified for the interrupted behaviors.

Sequential and concurrent behaviors Behaviors can be sequentially and concurrently composed and these composite behaviors can be selected in a decision-tree like single behaviors. In both concurrent and sequential behaviors the interruption or the termination with failure of one or more behaviors represents a problem. In the system, if a behavior is interrupted or unsuccessfully terminated the sequence of behaviors is aborted and all the concurrent behaviors are terminated. However, there can be cases in which such a drastic solution is not necessary. To give more flexibility to the system we intend to give the possibility to specify in terms of conditions $\rightarrow$ actions what to do in case of failure or termination of behaviors.

5 Roles and construction of decision-trees for roles

Roles give an essential contribution to efficiency and effectiveness of teams. In absence of roles in a soccer team all players would tend, for example, to get the ball, leaving their penalty area unguarded, or would try to defend the penalty area without leaving some players in the central part of the field ready for a counterattack. With roles, the resources of the team are distributed depending on the need and can be utilized in an effective way. Furthermore, the coordination is facilitated by the presence of roles both with respect to specification of coordination and to recognition of coordinated tactics. In fact knowing the role of an agent gives an indication of the capabilities of the agent and of the kind of coordinated behavior he is expected to perform. For example an attacker has already a specification of tactics for scoring. Therefore, this part does not need to be considered in the specification of a coordinated attack. Also, knowing that the player that has the ball is a left-midfielder gives an indication to an attacker that wants to be coordinated with him, on what direction the attack will take. In general the roles of the agent are maintained during the entire execution, but in some special cases the roles of the agent can also change during the execution. For example, if the team is losing, one midfielder can become an attacker.

An important aspect of roles is the inheritance relation between roles. For example, a left-midfielder is also a midfielder and a midfielder is also a player. This gives the possibility of easily specializing roles without having to redefine the parts common to several roles. Also an agent can have several roles in a team, that can be active in the same time as captain and midfielder, or that can alternate depending on the circumstances as midfielder and corner-kick shooter. There are therefore two kinds of composition of roles, one for specializing preexistent roles and one for combination of roles in the same agent. Examples of both cases can be found in Fig. 1. The problem in both cases is how to compose roles that are in general independently specified, without having to modify manually one or both the roles in order to deal with possible conflicts of actions and behaviors between roles. For example, a midfielder could try to pursue the ball in whatever position in the field the ball is, but a left-midfielder would be more inclined to pursue the ball if it is in his zone. One simple solution would be to just prioritize one role over the other, but in this case not very important actions in one role could prevent the execution of important actions in another role. The composition of roles is therefore done in term of composition of behaviors. In behaviors, as explained in section 4, a basic priority establishes the importance of the behavior, but the priority that a selected action has in this behavior, influences the priority that this action has compared with actions selected in other behaviors. With this priority assignment the actions selected in one behavior get a priority proportional to the importance that they have in the behavior. The specialization of roles is done by composition of the behavior module of the old role, and the behavior module containing the specializing part of the role. Priority and priority range of the behaviors is then decided.

The roles and the priority and priority range that the behaviors containing the roles will have, are decided when the agent is defined. The agent then executes the decision-trees created as a combination of his roles. In case an agent changes a role during the execution, he starts executing the decision-tree of the new role and stops executing the decision-tree of the previous role. As roles can have subroles in common (midfielder and attacker have for instance the subrole “player” in common), when an agent changes role, the roles’ relations are examined and just the behaviors that are in subroles not common with the new behavior are exited.

6 Coordination among agents

In both the soccer and the air combat domain team work is essential, but the possibility of communication, due to real time constrains and possibility of interception, is limited. Coordination is therefore obtained by common tactics and strategies learned and practised during training. Each agent knows the tactics of the team and his role in the tactics and tries to act according to this knowledge. The agent should initiate tactics depending on the situation and recognize the tactics that the other team members have initiated in order to do his part in it. The agent has three ways of recognizing the tactics that team mates are applying: observation of the general situation and knowledge of the tactics appropriate for the situation, observation of indicative actions of the team mates, and

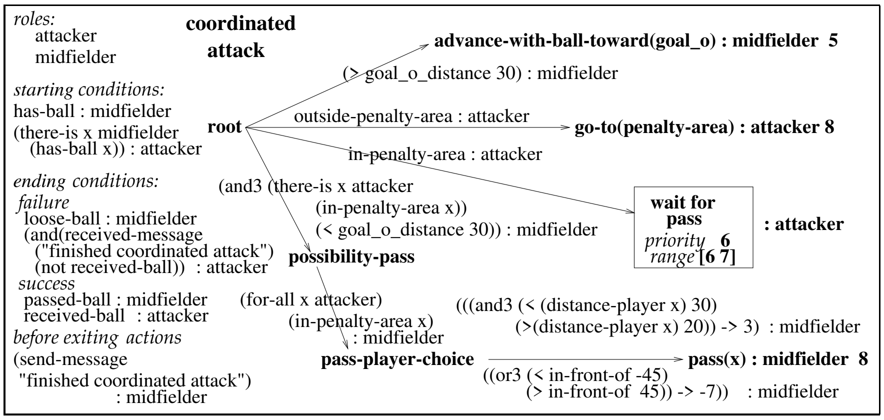

Fig. 3. Example of decision-tree for multiple agents

explicit communication. In general there can be a combination of these three aspects.

In our approach the knowledge required for applying the tactics is coded in behavior modules and the observations of the situation and of the actions of other agents influence the decisions about which tactics to adopt. Explicit communication is performed through communication actions present in the decision tree of the module. The message is stored in the state of the agent and will influence his future decisions.

6.1 Decision-tree for multiple roles

The specification of coordinated behavior is facilitated if the behavior of all the roles involved can be specified in the same structure. For this reason we introduce decision-trees in which the behaviors of a number of roles are specified. Every agent that has the appropriate role can then apply the behavior. In the decision-tree for coordinated behavior all actions and conditions are associated with roles.

Let us consider an example of a decision-tree for multiple agents. The behavior we want to specify is a simple case of coordinated attack between an attacker and a midfielder, Fig. 3. The coordinated behavior can be started if the midfielder has the ball and it is terminated when the midfielder loses or passes the ball. The midfielder advances with the ball toward the opponents' goal (**advance-with-ball-toward(goal-o)**) and when he is near the opponents' goal and an attacker is in the penalty area (*(and3 (there-is x attacker (in-penalty-are x)) (< (goal_o_distance)30))* he considers the possibility of passing the ball to an attacker among those that are in the penalty area (*for-all x attacker (in-penalty-area x)*). The preference to which player to pass is based on the distance of the player, and on the distance to the opponent goal (if a player is further than him from the goal the priority of passing to him is decreased by 7). The attacker, if he is outside the penalty area goes to the penalty area, and when he is inside the penalty area waits for a pass.

Performing a coordinated behavior involves a commitment towards the other agents participating in the behavior. Therefore it can be necessary to perform some actions before exiting the behavior. In our example before exiting the coordinated behavior the midfielder sends a message to advise that the coordinated attack is finished. The receiving of this message is in the ending condition of the coordinated behavior for the attacker.

The coordinated decision-trees are used in the specification of coordinated behavior, but then the parts of the trees regarding each role are transferred to the decision-trees of the role. The decision-trees of the roles are the ones that are actually executed. The branches regarding each role in a coordinated decision-tree are collected in a behavior module and this module is then added to the decision-tree of the role. Fig. 2 shows the module extracted from the coordinated attack decision-tree and how it is inserted in the decision-tree of an attacker. The figure has been explained already in section 3. We can here just point out that the starting condition of the coordinated attack is the condition in the branch for entering the behavior and the ending conditions are the condition for performing the action **exit-with-failure** and **exit-with-success**. A priority and priority range are assigned to the behavior in the role decision-tree as they depend on the importance that the behavior has in the context of the decision-tree.

7 Summary

We have described a decision mechanism for autonomous agents which attempts to provide a relatively simple way to specify coordination and team work in autonomous agents. A key concept in the paper is the concept of a role. Agents act and coordinate with other agents mostly depending on their roles. Roles can be specialized and several roles can be combined in the same agent through composition of behaviors. Behavior modules represent an important aspect of the system. They are used to specify coordination and the behavior proper of a role, simple routines and in general long term courses of action. The decision-mechanism is based on a decision-tree where actions and behaviors are assigned dynamically changing priorities and decisions are based both on sensory input and the internal state of the agent. Coordination between agents is mainly obtained through common tactics and strategies and observations of actions of team members. Explicit communication is also possible, but due to real time constraints and the possibility of interception by the opponents should be reduced to short and simple messages. Coordinated behaviors are specified in terms of roles and are encoded in decision-trees for multiple roles.

The system has been partially implemented and tested. It could unfortunatly not be tested in the context of the RoboCup competition in IJCAI97. We are now working in completing the implementation and in further testing.

8 Acknowledgments

We would like to thank D. Driankov, P. Lambrix and T. Vidal for their helpful comments. Supported by the Wallenberg Foundation project “Information Technology for Autonomous Aircraft”.

References

1. S. Ch'ng and L. Padgham. Role organisation and planning team strategies. In *Proc. of the 20th Australian Computer Science Conference*. Sydney, Australia, 1997.
2. P.R. Cohen and H.J. Levesque. Teamwork. In *Nous*, number 35. 1991.
3. S. Coradeschi and L. Karlsson. A decision-mechanism for reactive and cooperating soccer-playing agent. In *Linköping Electronic Articles in Computer and Information Science*, volume 2. 1997. http://www.ep.liu.se/ea/cis/1997/001/. Presented at "RoboCup Workshop: Soccer as a problem for Multi-Agent Systems" in the Second International Conference on Multi-Agent Systems (ICMAS-96), Kyoto, Japan.
4. S. Coradeschi, L. Karlsson, and A. Törne. Intelligent agents for aircraft combat simulation. In *Proc. of the 6th Conf. on Computer Generated Forces and Behavioral Representation*. Orlando, FL, 1996.
5. M. Fenster, S. Kraus, and J. S. Rosenschein. Coordination without communication: Experimental validation of focal point and techniques. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, CA, 1995.
6. M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proc. of IJCAI'89*. Detroit, Michigan, 1989.
7. M. J. Huber and E. H. Durfee. Deciding when to commit to action during observation-based coordination. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, CA, 1995.
8. J. E. Laird, R. M. Jones, and P. E. Nielsen. Coordinated behavior of computer generated forces in TacAir-Soar. In *Proc. of the 4th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL, 1994.
9. O. Shehory and S. Kraus. Cooperative goal-satisfaction without communication in large-scale agent-systems. In *Proceeding of ECAI'96*. Budapest, Hungary, 1996.
10. M. Tambe. Tracking dynamic team activity. In *Proceedings of AAAI'96*. Portland, Oregon, 1995.
11. M. Tambe. Teamwork in real-world, dynamic environments. In *Proc. of the 2nd International Conference on Multi-agent Systems (ICMAS-96)*. Kyoto, Japan, 1996.
12. M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1), 1995.
13. G. Tidhar, A. S. Rao, and E. A. Sonenberg. Guided team selection. In *Proceedings of the 2nd International Conference on Multi-agent Systems (ICMAS-96)*. Kyoto, Japan, 1996.
14. G. Tidhar, M. Selvestrel, and C. Heinze. Modeling teams and team tactics in whole air mission modelling. In *Proceedings of the 8th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*. Gordon and Breach Science Publishers, Melbourne, Australia, 1995.