# Using an Explicit Teamwork Model and Learning in RoboCup: An Extended Abstract

Stacy Marsella, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem, Randall Hill
Gal A. Kaminka, Zhun Qiu, Milind Tambe

Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292, USA
robocup-sim@isi.edu

## 1 Introduction

The RoboCup research initiative has established synthetic and robotic soccer as testbeds for pursuing research challenges in Artificial Intelligence and robotics. This extended abstract focuses on teamwork and learning, two of the multi-agent research challenges highlighted in RoboCup. To address the challenge of teamwork, we discuss the use of a domain-independent explicit model of teamwork, and an explicit representation of team plans and goals. We also discuss the application of agent learning in RoboCup.

The vehicle for our research investigations in RoboCup is **ISIS** (**ISI S**ynthetic), a team of synthetic soccer-players that successfully participated in the simulation league of RoboCup'97, by winning the third place prize in that tournament. In this position paper, we briefly overview the ISIS agent architecture and our investigations of the issues of teamwork and learning. The key novel issues for our team in RoboCup'98 will be a further investigation of agent learning, and further analysis of teamwork related issues.

## 2 The ISIS Architecture

An ISIS agent uses a two-tier architecture, consisting of a higher-level that makes decisions and a lower-level that handles various time critical functions tied to perception and action.

ISIS's lower-level, developed in C, communicates inputs received from the RoboCup simulator (after sufficient pre-processing), to the higher level. The lower-level also rapidly computes some recommended directions for turning and kicking, to be sent to the higher-level. For instance, a group of C4.5 rules compute a direction to intelligently shoot the ball into the opponents' goal (discussed further in Section 4). The lower-level also computes a plan to intercept the ball consisting of turn or dash actions.

The lower-level does not make any decisions with respect to its recommendations however. Instead, all such decision-making rests with the higher level, implemented in the Soar integrated AI architecture[11, 14]. Once the Soar-based

M. Asada and H. Kitano (Eds.): RoboCup-98, LNAI 1604, pp. 237-245, 1999.

higher-level reaches a decision, it communicates with the lower-level, which then sends the relevant information to the simulator.

The Soar architecture involves dynamic execution of an operator (reactive plan) hierarchy. An operator begins execution (i.e., the operator is activated), when it is selected for execution, and it remains active until explicitly terminated. Execution of higher-level abstract operators leads to subgoals, where new operators are selected for execution, and thus a hierarchical expansion of operators ensues. Operators in Soar are thus similar to reactive plans in architectures such as RAP[4].

## 3 Teamwork

There are two key aspects of ISIS's approach to teamwork. The first is the explicit representation of team activities via the use of explicit representation of *team operators* (reactive team plans). Team operators explicitly express a team's joint activities, unlike the regular "individual operators" which express an agent's own activities. Furthermore, while an individual operator applies to an agent's private state (an agent's private beliefs), a team operator applies to an agent's *team state*. A team state is the agent's (abstract) model of the team's mutual beliefs about the world, e.g., the team's currently mutually believed strategy. An ISIS agent can also maintain subteam states for subteam participation. Each team member maintains its own copy of the team state, and any subteam states for subteams it participates in. That is, there is no shared memory among the team members.

The second key aspect of teamwork in ISIS is its novel approach to coordination and communication via the use of a general-purpose teamwork model. In particular, to surmount uncertainties that arise in dynamic environments and maintain coherence in teamwork, team members must be provided the capability of highly flexible coordination and communication. To this end, general-purpose explicit models of teamwork have recently been proposed as a promising approach to enhance teamwork flexibility[6, 19]. Essentially, teamwork models provide agents with the capability of first principles reasoning about teamwork to provide teamwork flexibility. Such teamwork models also enable code reuse.

We investigate the use of STEAM[17, 18, 19], a state-of-the-art general-purpose model of teamwork. STEAM models team members' responsibilities and commitments in teamwork in a domain-independent fashion. As a result, it enables team members to autonomously reason about coordination and communication, improving teamwork flexibility. Furthermore, due to its domain-independence, STEAM has been demonstrated to be reusable across domains. STEAM uses the formal *joint intentions* framework[3, 7] as its basic building block, but it is also influenced by the SharedPlans theory[5], and includes key enhancements to reflect the constraints of real-world domains. For instance, the Joint intentions theory requires that agents attain mutual belief in establishing and terminating joint intentions, but does not specify how mutual belief should be attained. STEAM uses decision-theoretic reasoning to select the appropriate

method for attaining mutual belief. Thus, it does not rely exclusively on explicit communication (e.g, "Say" in RoboCup simulation) for attaining mutual belief in the team; instead, it may rely on plan-recognition.

A typical example of STEAM in operation is the DEFEND-GOAL team operator executed by the defender subteam. In service of DEFEND-GOAL, players in this subteam normally execute the SIMPLE-DEFENSE team operator to position themselves properly on the field and to try to be aware of the ball position. Of course, each player can only see in its limited cone of vision, and particularly while repositioning itself, can be unaware of the approaching ball. Here is where teamwork can be beneficial. In particular, if any one of these players sees the ball as being close, it declares the SIMPLE-DEFENSE team operator to be irrelevant. Its teammates now focus on defending the goal in a coordinated manner via the CAREFUL-DEFENSE team operator. Should any one player in the goalie subteam see the ball move sufficiently far away, it again alerts its team mates (that CAREFUL-DEFENSE is achieved). The subteam players once again execute SIMPLE-DEFENSE to attempt to position themselves close to the goal. In this way, agents coordinate their defense of the goal.

### 3.1 New Issues

Several issues have been brought forward due to our application of the teamwork model in RoboCup. First, RoboCup is a highly dynamic, real-time domain, where reasoning from first principles of teamwork via the teamwork model can sometimes be inefficient. Therefore, to improve efficiency, we plan to compile the teamwork model, so that the typical cases of reasoning about teamwork are speeded up. This can be achieved via machine learning methods such as chunking[11] (a form of explanation-based learning)[10]. The teamwork model itself will be retained however, since unusual cases may still arise, and require the first principles teamwork reasoning offered by the teamwork model.

Second, the teamwork model may sometimes enforce a rigid coherence constraint on the team, always requiring mutual belief to be attained. However, it is not always straightforward to attain such mutual belief. In RoboCup simulation, the shouting range (i.e., the range of the "say" message) is limited. A player may not necessarily be heard at the other end of the field. A tradeoff in the level of coherence and team performance is therefore necessary. We plan to investigate this tradeoff, and suggest corresponding modifications to teamwork models.

Finally, we are extending the capabilities of agents to deal with potential inconsistencies in their beliefs. Currently in STEAM, if an agent discovers new information relevant to the team goal, it will use decision-theoretic reasoning to select a method for attaining mutual belief. In particular, it may inform other agents by communicating the belief. However, the recipients of such communications accept the acquired belief without examination even if there are inconsistencies between this belief and a recipient's existing "certain" beliefs. This can be a problem.

For instance, in RoboCup if a defender sees the ball as being close and tells other defenders, all the defenders will establish a joint intention to approach the

ball; if at the same time the player is informed by another defender who stands far away from the ball that the ball is far, an inconsistency occurs. The joint goal will be terminated, and there will be recurring processes to form and end the joint intentions, and the whole team will get stuck at this point.

To address this problem, we have taken an approach whereby agents model the beliefs of other agents in order to detect inconsistencies and to decide whether to negotiate. A key idea in our approach is that negotiation itself takes time, which can be a significant factor in RoboCup. Thus, it is often the case that an agent should decide not to argue with its disagreeing teammates and, instead, go along with the temporary inconsistency.

To detect an inconsistency, the agent receiving a message has first to compare its own beliefs with the belief conveyed in the message sent. Since the conveyed belief may not conflict explicitly with the agent's existing beliefs, the agent uses belief inference rules to figure out the relevant beliefs of the message sender. Further, since both senders and recipients may have beliefs which they are more or less certain about, the detection of inconsistency takes into account whether beliefs are "certain" (e.g., supported by direct sensory evidence as opposed to plausible default beliefs).

To decide whether to negotiate over inconsistencies, an agent uses a decision-theoretic method to select an approach to addressing the inconsistencies. Briefly, the agent always has three choices. First of all it can just keep its own belief and work on it without any argument with its teammates. Second, it can choose to accept without argument what the sender says. Finally, the third option is to expend the effort both to detect a clash in beliefs and to negotiate a resolution. In that case, the negotiation may lead to agreement with the sender's observation or alternatively may persuade the sender to accept the recipient's belief.

Under different circumstances, the cost and utility of a decision will vary a lot. In the RoboCup case proposed above, the cost of the "negotiation" may be substantial, not only because of the large amount of resources consumed, but more importantly because of the time pressure in the soccer competition. Furthermore, since the situation may change greatly (the ball may have rolled to a far-away position) after the agents are set with their arguments, the possibility and benefit of the "right" decision will decrease significantly. All these lead to a rather low expected utility of negotiation. So in this specific case, the player involved will either stick to its own belief or turn to other's observation directly, rather than bothering to argue to resolve the disagreement.

## 4 Lower-level skills and Learning

Inspired by previous work on machine learning in RoboCup[15, 9], we focused on techniques to improve individual players' skills to kick, pass, or intercept the ball. Fortunately, the two layer ISIS architecture helps to simplify the problem for skill learning. In particular, the lower-level in ISIS is designed to provide several recommendations (such as various kicking directions) to the higher-level, but it need not arrive at a specific decision (one specific kicking direction). Thus,

an individual skill, such as a kicking direction to clear the ball, can be learned independently of other possible actions.

Learning has currently been applied to (i) selection of an intelligent direction to shoot a ball when attempting to score a goal and (ii) selection of a plan to intercept an incoming ball.

Scoring goals is clearly a critical soccer skill. However, our initial hand-coded, approaches to determining a good direction to kick the ball, based on heuristics such as "shoot at the center of the goal", or "shoot to a corner of the goal", failed drastically. In part, this was because heuristics were often foiled by the fact that small variations in the configuration of players around the opponent's goal or a small variation in the shooter's position may have dramatic effects on the right shooting direction.

To address these problems, we decided to rely on automated, *offline* learning of the shooting rules. A human expert created a set of shooting situations, and selected the optimal shooting direction for each such situation. The learning system trained on these shooting scenarios. C4.5[12] was used as the learning system, in part because it has the appropriate expressive power to express game situations and can handle both missing attributes and a large number of training cases.

In our representation, each C4.5 training case has 39 attributes, such as the shooters angles to the other visible players. The system was trained on over roughly 1400 training cases, labeled by our expert with one of UP, DOWN, and CENTER (region of the goal) kicking directions. The result was that given a game situation characterized by the 39 attributes, the decision tree selected the best of the three shooting directions. The resulting decision tree provided a 70.8%-accurate set of shooting rules.

These learned rules for selecting a shooting direction were used successfully in RoboCup'97. The higher-level typically selected this learned shooting direction when players were reasonably close to the goal, and could see the goal.

In contrast to the *offline* learning of shooting direction, we have also begun to explore *online* learning of intercept plans. In our initial implementation, ISIS players used a simple hand-coded routine to determine the plan for intercepting the ball. Our experience at RoboCup97 was that the resulting intercept plans work fine under some playing conditions, but fail under others. The result often depends on such external factors as network conditions, frequency of perceptual updates from the soccer server and the style of play of the opposing team. Unlike real soccer players, our ISIS players' intercept skills were not adapting very well to differing external factors.

To address this problem, we are exploring how players can adapt their intercept online, under actual playing conditions. Of course, an adaptive intercept has inherent risks. In the course of a game, there are not many opportunities to intercept the ball, and worse, inappropriate adaptations can have dire consequences. Therefore, it is important for adaptation to be done rapidly, reasonably and smoothly.

To assess these risks and the overall feasibility of an adaptive intercept, we

have started to explore simple adaptive approaches. In keeping with the risks, our current approach uses hill-climbing search in the space of plans where evaluation of success or failure is driven by an "oracle", ISIS's higher-level, decision-making tier. In addition, we have adopted a conservative approach of using distinct searches for distinct input conditions, so for instance balls that are moving towards the player may be treated separately from balls moving away.

As a player uses the intercept assigned by some input condition, failure to meet expectations will result in a new intercept plan for this input condition if there has been a history of similar failures. ISIS's higher-level drives the evaluation since it has the necessary context to model the failure. For instance, failure due to a blocking player is treated differently from failure due to an improper turn angle.

This work is preliminary and has not been fully evaluated. However, it has been tested under fixed, ideal conditions (e.g., reliable perceptual updates). In these tests, the method exhibits consistent and rapid convergence on simple turn and dash plans that are at least as good as the manually derived plans used at RoboCup97. In addition to more extensive evaluation under varying conditions, we are now considering enhancements such as allowing the learning under one input condition to influence similar input conditions.

## 5 Evaluation

There are several aspects to evaluation of ISIS. As mentioned earlier, ISIS successfully participated in RoboCup'97, winning the third-place prize in the simulation league tournament, in the 29 teams that participated. Overall at RoboCup97, ISIS won six out of the seven games in which it competed, outscoring its components 37 goals to 19.

Another key aspect of evaluation is measuring the contribution of the explicit teamwork model (STEAM) to ISIS. STEAM's contribution is both in terms of improved teamwork performance and reduced development time. To measure the performance improvement due to STEAM, we experimented with two different settings of communication cost in STEAM. At "low" cost, ISIS agents communicate a significant number of messages. At "high" communication cost, ISIS agents communicate no messages. Since the portion of the teamwork model in use in ISIS is effective only with communication, a "high" setting of communication cost essentially nullifies the effect of the teamwork model.

For each setting of communication cost, ISIS played 7 games against a fixed opponent team of roughly equivalent capability. With low communication cost, ISIS won 3 out of the 7 games. It scored 18 goals against the opponents, and had 22 goals scored against it. With high communication cost, ISIS won none out of the 7 games it played. It scored only 3 goals, but had 20 goals scored against it.

The results clearly illustrate that the STEAM teamwork model does make a useful contribution to ISIS's performance. Furthermore, by providing general teamwork capabilities, it also reduces development time. For instance, without STEAM, all of the communication for jointly initiating and terminating all of

the team operators (about 20 in the current system) would have had to be implemented via dozens of domain-specific coordination plans.

## 6 Related Work

In terms of work within RoboCup, ISIS was the only team at RoboCup'97 that investigated the use of a general, domain-independent teamwork model to guide agent's communication and coordination in teamwork. Some researchers investigating teamwork in RoboCup have used explicit team plans and roles, but they have relied on *domain-dependent* communication and coordination. Typical examples include [2, 1]. Other investigations of teamwork in RoboCup have used implicit or emergent coordination. A typical example is Yokota et al.[20].

Our application of learning in ISIS agents is similar to some of the other investigations of learning in RoboCup agents. For instance, Luke et al.[8] use genetic programming to build agents that learn to use their basic individual skills in coordination. Stone and Veloso[16] present a related approach, in which the agents learn a decision tree which enables them to select a recipient for a pass.

In terms of related work outside RoboCup, the use of a teamwork model remains a distinguishing aspect of teamwork in ISIS. The STEAM teamwork model used in ISIS, is among just a very few implemented general models of teamwork. Other models include Jennings' *joint responsibility* framework in the GRATE* system[6] (based on Joint Intentions theory), and Rich and Sidner's COLLAGEN[13] (based on the SharedPlans theory), that both operate in complex domains. STEAM significantly differs from both these frameworks, via its focus on a different (and arguably wider) set of teamwork capabilities that arise in domains with teams of more than two-three agents, with more complex team organizational hierarchies, and with practical emphasis on communication costs (see [19] for a more detailed discussion).

## 7 Summary

We have discussed teamwork and learning, two important research issues in multi-agent systems. The vehicle for our research is ISIS, an implemented team of soccer playing agents, that successfully participated in the simulation league of the RoboCup'97 soccer tournament. We have taken a principled approach in developing ISIS, guided by the research opportunities in RoboCup. Despite the significant risk in following such a principled approach, ISIS won the third place in the 29 teams that participated in the RoboCup'97 simulation league tournament.

There are several key issues that remain open for future work. One key issue is improved agent- or team-modeling. One immediate application of such modeling is recognition that an individual, particularly a team member, is unable to fulfill its role in the team activity. Other team members can then take over the role

of this failing team member. Team modeling can also be applied to recognize opponent behaviors and counter them intelligently.

## Acknowledgement

This research is supported in part by NSF grant IRI-9711665. We thank Bill Swartout, Paul Rosenbloom and Yigal Arens of USC/ISI for their support of the RoboCup activities described in this paper.

## References

1. T. F. Bersano-Begey, P. G. Kenny, and E. H. Durfee. Agent teamwork, adaptive learning, and adversarial planning in robocup using a prs architecture. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.
2. S. Ch'ng and L. Padgham. Team description: Royal merlbourne knights. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.
3. P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.
4. J. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1987.
5. B. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.
6. N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.
7. H. J. Levesque, P. R. Cohen, and J. Nunes. On acting together. In *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, Calif.: AAAI press, 1990.
8. S. Luke, Hohn C., J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.
9. H. Matsubara, I. Noda, and K. Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass play in soccer. In S. Sen, editor, *AAAI Spring Symposium on Adaptation, Coevolution and Learning in multi-agent systems*, March 1996.
10. T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
11. A. Newell. *Unified Theories of Cognition*. Harvard Univ. Press, Cambridge, Mass., 1990.
12. J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.
13. C. Rich and C. Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the International Conference on Autonomous Agents (Agents'97)*, 1997.
14. P. S. Rosenbloom, J. E. Laird, A. Newell, , and R. McCarl. A preliminary analysis of the soar architecture as a basis for general intelligence. *Artificial Intelligence*, 47(1-3):289–325, 1991.

15. P. Stone and M. Veloso. Towards collaborative and adversarial learning: a case study in robotic soccer. In S. Sen, editor, *AAAI Spring Symposium on Adaptation, Coevolution and Learning in multi-agent systems*, March 1996.
16. P. Stone and M. Veloso. Using decision tree confidence factors for multiagent control. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.
17. M. Tambe. Teamwork in real-world, dynamic environments. In *Proceedings of the International Conference on Multi-agent Systems (ICMAS)*, December 1996.
18. M. Tambe. Agent architectures for flexible, practical teamwork. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 1997.
19. M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.
20. K. Yokota, K. Ozako, Matsumoto A., T. Fujii, Asama H., and I. Endo. Cooperation towards team play. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.