The CMUnited-97 Simulator Team

Peter Stone and Manuela Veloso

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{pstone,veloso}@cs.cmu.edu
http://www.cs.cmu.edu/{~pstone,~mmv}

In *RoboCup-97: Robot Soccer World Cup I*,
H. Kitano (ed.), 1998. Springer Verlag, Berlin.

Abstract. The Soccer Server system provides a rich and challenging multiagent, real-time domain. Agents must accurately perceive and act despite a quickly changing, largely hidden, noisy world. They must also act at several levels, ranging from individual skills to full-team collaborative and adversarial behaviors. This article presents the CMUnited-97 approaches to the above challenges which helped the team to the semifinals of the 29-team RoboCup-97 tournament.

1 Introduction

The Soccer Server system [5] used at RoboCup-97 [2] provides a rich and challenging multiagent, real-time domain. Sensing and acting is noisy, while inter-agent communication is unreliable and low-bandwidth.

In order to be successful, each agent in a team must be able to sense and act in real time: sensations arrive at unpredictable intervals while actions are possible every 100ms. Furthermore, since the agents get local, noisy sensory information, they must have a method of converting their sensory inputs into a good world model.

Action capabilities range from low-level individual skills, such as moving to a point or kicking the ball, to high-level strategic collaborative and adversarial reasoning. Agents must be able to act autonomously, while working together with teammates towards their common overall goal. Since communication is unreliable and perception is incomplete, centralized control is impossible.

This article presents the CMUnited-97 approaches to the above challenges which helped the team to the semifinals of the 29-team RoboCup-97 simulator tournament. Section 2 introduces our overall agent architecture which allows for team coordination. Section 3 presents our agents' world model in an uncertain environment with lots of hidden state. Section 4 lays out the agents' hierarchical behavior structure that allows for machine learning at all levels of behavior from individual to collaborative to adversarial. Our team's flexible teamwork structure, which was also used by the CMUnited-97 small-size robot team [7], is presented in Section 5. Section 6 concludes.

2 Team Member Architecture

Our new teamwork structure is situated within a team member architecture suitable for domains in which individual agents can capture locker-room agreements and respond to the environment, while acting autonomously. Based on a standard agent architecture, our team member architecture allows agents to sense the environment, to reason about and select their actions, and to act in the real world. At team synchronization opportunities, the team also makes a locker-room agreement for use by all agents during periods of low communication. Figure 1 shows the functional input/output model of the architecture.

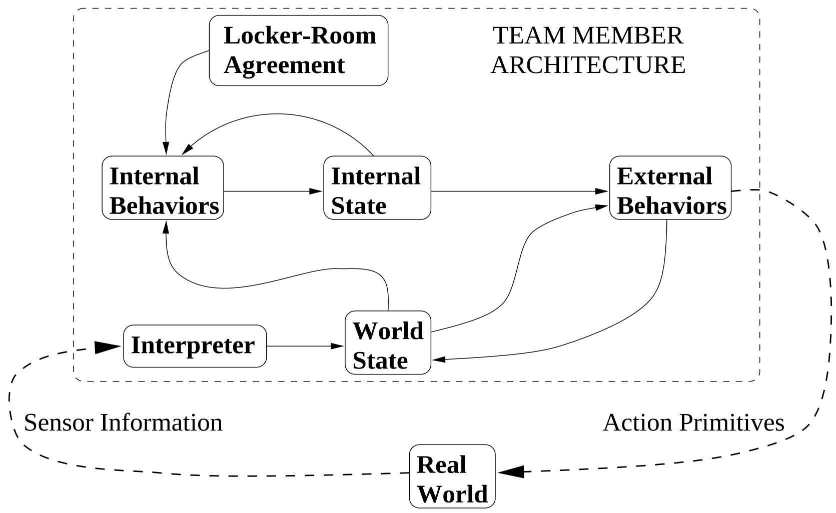

Fig. 1. The team member architecture for PTS domains.

The agent keeps track of three different types of state: the *world state*, the *locker-room agreement*, and the *internal state*. The agent also has two different types of behaviors: *internal behaviors* and *external behaviors*.

The World State reflects the agent's conception of the real world, both via its sensors and via the predicted effects of its actions. It is updated as a result of processed sensory information. It may also be updated according to the predicted effects of the external behavior module's chosen actions. The world state is directly accessible to both internal and external behaviors.

The Locker-Room Agreement is set by the team when it is able to privately synchronize. It defines the flexible teamwork structure as presented below as well as inter-agent communication protocols. The locker-room agreement may change periodically when the team is able to re-synchronize; however, it generally remains unchanged. The locker-room agreement is accessible only to internal behaviors.

The Internal State stores the agent's internal variables. It may reflect previous and current world states, possibly as specified by the locker-room agreement. For example, the agent's role within a team behavior could be stored as part of the internal state, as could a distribution of past world states. The agent updates its internal state via its internal behaviors.

The Internal Behaviors update the agent's internal state based on its current internal state, the world state, and the team's locker-room agreement.

The External Behaviors reference the world and internal states, sending commands to the actuators. The actions affect the real world, thus altering the agent's future percepts. External behaviors consider only the world and internal states, without direct access to the locker-room agreement.

Internal and external behaviors are similar in structure, as they are both sets of condition/action pairs where conditions are logical expressions over the inputs and actions are themselves behaviors as illustrated in Figure 2. In both cases, a behavior is a directed acyclic graph (DAG) of arbitrary depth. The leaves of the DAGs are the behavior types' respective outputs: internal state changes for internal behaviors and action primitives for external behaviors.

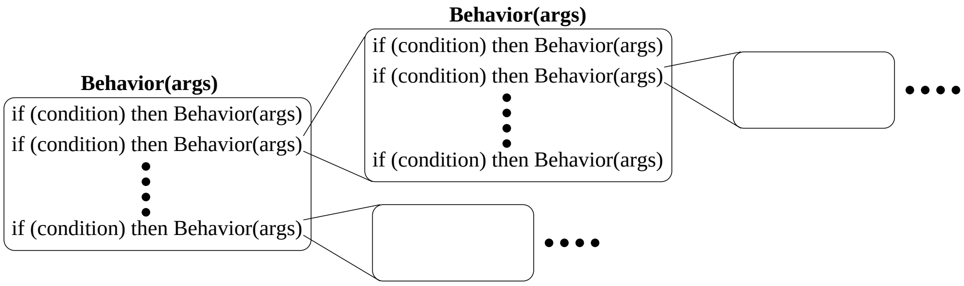

Fig. 2. Internal and external behaviors are organized in a directed acyclic graph.

Our notion of behavior is consistent with that laid out in [4]. In particular, behaviors can be nested at different levels: selection among lower-level behaviors can be considered a higher-level behavior, with the overall agent behavior considered a single "do-the-task" behavior. There is one such *top-level* internal behavior and one top-level external behavior; they are called when it is time to update the internal state or act in the world, respectively. The team structure presented in Section 5 relies and builds upon this team member architecture.

3 Predictive Memory

Based on the sensory information received from the environment, each agent can build its own world state. We developed a predictive memory model that builds a probabilistic representation of the state based on past observations. By making the right assumptions, an effective model can be created that can store and update knowledge even when there are inaccessible parts of the environment. The agent relies on past observations to determine the positions of objects that are not currently visible. We conducted experiments to compare the effectiveness of this approach with a simpler approach, which ignored the inaccessible parts of the environment. The results obtained demonstrate that this predictive approach does generate an effective memory model, which outperforms a non-predictive model [1].

4 Layered Learning

Once the world model is successfully created, the agents must use it to respond effectively to the environment. As described in Section 2, internal behaviors update the internal state while external behaviors produce executable actuator commands. Spanning both internal and external behaviors, *layered learning* [6] is our bottom-up hierarchical approach to client behaviors that allows for machine learning at the various levels (Figure 3). The key points of the layered learning technique are as follows:

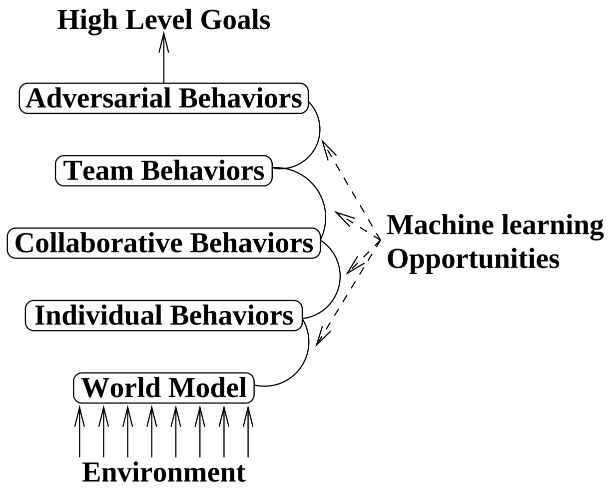

Fig. 3. An overview of the Layered Learning framework. It is designed for use in domains that are too complex to learn a mapping straight from sensors to actuators. We use a hierarchical, bottom-up approach

- The difficult aspects of the domain determine which behaviors are to be learned.
- The learned behaviors are combined in a vertical fashion, one being used as a part of the other.

Table 1 illustrates possible behavior levels within the robotic soccer domain. Because of the complexity of the domain, it is futile to try to learn intelligent behaviors straight from the primitives provided by the server. Instead, we identified useful low-level skills that must be learned *before* moving on to higher level strategies. Using our own experience and insights to help the clients learn, we acted as human coaches do when they teach young children how to play real soccer.

Layered Strategic Level	**Behavior Type**	**Examples**
Robot–ball	individual	intercept
Action selection	individual	pass or dribble
One-to-one player	collaborative	pass, aim
One-to-many player	collaborative	pass to teammate
Team formation	team	strategic positioning
Team-to-opponent	adversarial	strategic adaptation

Table 1. Examples of different behavior levels.

The low-level behaviors, such as ball interception and passing, are external behaviors involving direct action in the environment. Higher level behaviors, such as strategic positioning and adaptation, are internal behaviors involving changes to the agent's internal state. The type of learning used at each level depends upon the task characteristics. We have used neural networks and decision trees to learn ball interception and passing respectively [6]. These off-line approaches are appropriate for opponent-independent tasks that can be trained outside of game situations. We are using on-line reinforcement learning approaches for behaviors that depend on the opponents. Adversarial actions are clearly opponent-dependent. Team collaboration and action selection can also benefit from adaptation to particular opponents.

5 Flexible Team Structure

One approach to task decomposition in the Soccer Server is to assign fixed positions to agents.[1] Such an approach leads to several problems: i) short-term inflexibility in that the players cannot adapt their positions to the ball's location on the field; ii) long-term inflexibility in that the team cannot adapt to opponent strategy; and iii) local inefficiency in that players often get tired running across the field back to their positions after chasing the ball. Our introduced formations allow for flexible teamwork and combat these problems.

[1] One of the teams in Pre-RoboCup-97 (IROS'96) used and depended upon these assignments: the players would pass to the fixed positions regardless of whether there was a player there.

The definition of a position includes *home coordinates*, a *home range*, and a *maximum range*, as illustrated in Figure 4(a). The position's home coordinates are the default location to which the agent should go. However, the agent has some flexibility, being able to set its actual home position anywhere within the home range. When moving outside of the max range, the agent is no longer considered to be in the position. The home and max ranges of different positions can overlap, even if they are part of the same formations.

A formation consists of a set of positions and a set of units. The formation and each of the units can also specify inter-position behavior specifications for the member positions. Figure 4(b) illustrates the positions in one particular formation, its units, and their captains. Here, the units contain defenders, midfielders, forwards, left players, center players, and right players.

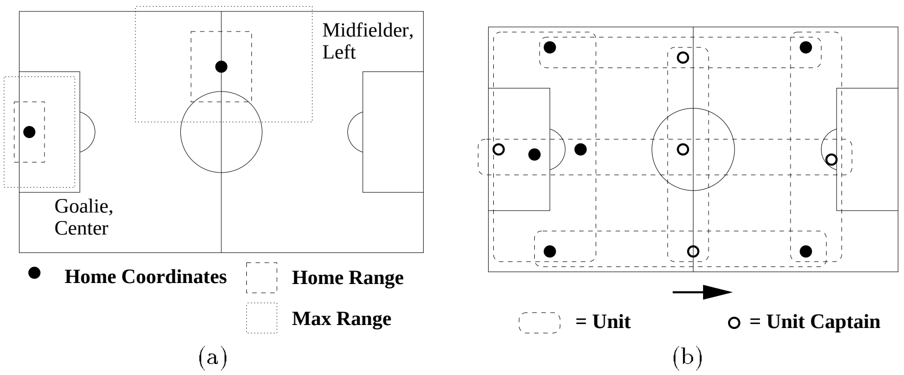

Fig. 4. (a) Different positions with home coordinates and home and max ranges. (b) Positions can belong to more than one unit.

Since the players are all autonomous, in addition to knowing its own position, each one has its own belief of the team's current formation along with the time at which that formation was adopted, and a map of teammates to positions. Ideally, the players have consistent beliefs as to the team's state, but this condition cannot be guaranteed between synchronization opportunities.

Our team structure allows for several significant features in our simulated soccer team. These features are: (i) the definition of and switching among multiple formations with units; (ii) flexible position adjustment and position switching; (iii) and pre-defined special purpose plays (set plays).

5.1 Dynamic Switching of Formations

We implemented several different formations, ranging from very defensive (8-2-0) to very offensive (2-4-4).[2]

The full definitions of all of the formations are a part of the locker-room agreement. Therefore, they are all known to all teammates. However during the periods of full autonomy and low communication, it is not necessarily known what formation the rest of the teammates are using. Two approaches can be taken to address this problem:

- **static formation** - the formation is set by the locker-room agreement and never changes;
- **run-time switch of formation** - during team synchronization opportunities, the team sets globally accessible run-time evaluation metrics as formation-changing indicators.

The CMUnited RoboCup-97 team switched formations based on the amount of time left relative to the difference in score: the team switched to an offensive formation if it was losing near the end of the game and a defensive formation if it was winning. Since each agent was able to independently keep track of the score and time, the agents were always able to switch formations simultaneously.

5.2 Flexible Positions

In our multiagent approach, the player positions itself flexibly such that it *anticipates* that it will be useful to the team, either offensively or defensively.

Two ways in which agents can use the position flexibility is to react to the ball's position and to mark opponents. When reacting to the ball's position, the agent moves to a location within its range that minimizes its distance to the ball. When marking opponents, agents move next to a given opponent rather than staying at the default position home. The opponent to mark can be chosen by the player (e.g., the closest opponent), or by the unit captain which can ensure that all opponents are marked, following a preset algorithm as part of the locker-room agreement.

Homogeneous agents can play different positions. But such a capability raises the challenging issue of when the players should change positions. The locker-room agreement provides procedures to the team that allow for coordinated role changing. In our case, the locker-room agreement designates an order of precedence switching among positions within each unit. By switching positions within a formation, the overall joint performance of the team is improved. Position-switching saves player energy and allows them to respond more quickly to the ball.

5.3 Pre-Planned Set Plays

The final implemented improvement facilitated by our flexible teamwork structure is the introduction of set plays, or pre-defined special purpose plays. As a part of the locker-room agreement, the team can define multi-step multiagent

[2] Soccer formations are typically described as goalie-defenders-midfielders--forwards [3].

plans to be executed at appropriate times. Particularly if there are certain situations that occur repeatedly, it makes sense for the team to devise plans for those situations.

In the robotic soccer domain, certain situations occur repeatedly. For example, after every goal, there is a kickoff from the center spot. When the ball goes out of bounds, there is a goal-kick, a corner-kick, or a kick-in. In each of these situations, the referee informs the team of the situations. Thus all the players know to execute the appropriate set play. Associated with each set-play-role is not only a location, but also a behavior. The player in a given role might pass to the player filling another role, shoot at the goal, or kick the ball to some other location.

We found that the set plays significantly improved our team's performance. During the RoboCup-97 competitions, several goals were scored off of of set plays.

6 Conclusion

The Soccer Server system provides a wide range of AI challenges. Here we have described our approaches to problems ranging from world modelling to multiagent cooperation. Machine leaning techniques are used throughout to improve performance of individual and team behaviors. Our successful implementation reached the semifinals of RoboCup-97.

Acknowledgements

This research is sponsored in part by the Defense Advanced Research Projects Agency (DARPA), and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-95-1-0018 and in part by the Department of the Navy, Office of Naval Research under contract number N00014-95-1-0591. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Air Force, of the Department of the Navy, Office of Naval Research or the United States Government.

References

1. Mike Bowling, Peter Stone, and Manuela Veloso. Predictive memory for an inaccessible environment. In *Proceedings of the IROS-96 Workshop on RoboCup*, pages 28–34, Osaka, Japan, November 1996.
2. Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.
3. Michael L. LaBlanc and Richard Henshaw. *The World Encyclopedia of Soccer*. Visible Ink Press, 1994.

4. Maja J. Mataric. Interaction and intelligent behavior. MIT EECS PhD Thesis AITR–1495, MIT AI Lab, August 1994.
5. Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996. Soccer server is available at URL http://ci.etl.go.jp/ noda/soccer/server/index.html.
6. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.
7. Manuela Veloso, Peter Stone, Kwun Han, and Sorin Achim. The CMUnited-97 small-robot team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 242–256. Springer Verlag, Berlin, 1998.