CMUnited-98: RoboCup-98 Simulator World Champion Team [*†]

Peter Stone, Manuela Veloso, and Patrick Riley
Computer Science Department,Carnegie Mellon University
Pittsburgh, PA 15213
{pstone,veloso}@cs.cmu.edu, priley@andrew.cmu.edu
http://www.cs.cmu.edu/{~pstone,~mmv}, http://www.andrew.cmu.edu/~priley

July 24, 1999

Abstract

The CMUnited-98 simulator team became the 1998 RoboCup simulator league champion by winning all 8 of its games, outscoring opponents by a total of 66–0. CMUnited-98 builds upon the successful CMUnited-97 implementation, but also improves upon it in many ways. This article gives an overview of the CMUnited-98 agent skill and multi-agent coordination strategies, emphasizing the recent improvements.

1 Introduction

The CMUnited-98 simulator team became the 1998 RoboCup (Kitano *et al.* 1997) simulator league champion by winning all 8 of its games, outscoring opponents by a total of 66–0. CMUnited-98 builds upon the successful CMUnited-97 implementation (Stone & Veloso 1998a), but also improves upon it in many ways.

CMUnited-98 agents are capable of perception, cognition, and action. By perceiving the world, each fully distributed agent builds a model of the world's current state. Then, based on a complex set of behaviors, it chooses an action appropriate for the current world state. While acting autonomously, each agent contributes to the overall team's goal.

The agents operate in the RoboCup soccer server (Noda *et al.* 1998), a robotic soccer simulator. The simulator, acting as a server, accepts action commands from fully distributed clients (agents) throughout a 100 msec cycle and then updates the world state all at once at the end of the cycle. Agents receive sensory perceptions from the simulator asynchronously and at unpredictable intervals. A complete overview of the soccer server, including agent sensor and actuator capabilities, is given in (Stone 1998).

This article gives an overview of the CMUnited-98 agent-skills and multi-agent coordination strategies, emphasizing the recent improvements. The most notable improvements are the predictive low-level skills and the strategic agent positioning in anticipation of passes from teammates. The success of CMUnited-98 also depends on our previous research innovations including layered learning, a flexible teamwork structure, and a novel communication paradigm (Stone 1998).

2 Periodic Team Synchronization (PTS) Domains

We view robotic soccer as an example of a periodic team synchronization (PTS) domain. We define PTS domains as domains with the following characteristics:

- There is a team of autonomous agents A that collaborate towards the achievement of a joint long-term goal G.

[*] An extended version of this article appears in (Stone *et al.* 1999).

[†] This research is sponsored in part by the DARPA/RL Knowledge Based Planning and Scheduling Initiative under grant number F30602-98-2-0135. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the U. S. Government.

- Periodically, the team can synchronize with no restrictions on communication: the agents can in effect inform each other of their entire internal states and decision-making mechanisms with no adverse effects upon the achievement of G. These periods of full communication can be thought of as times at which the team is "off-line."

- In general (i.e., when the agents are "on-line"):

  – The domain is *dynamic* and *real-time* meaning that team performance is adversely affected if an agent ceases to act for a period of time: G is either less likely to be achieved, or likely to be achieved farther in the future. That is, consider agent a_i. Assume that all other agent behaviors are fixed and that were a_i to act optimally, G would be achieved with probability p at time t. If a_i stops acting for any period of time and then resumes acting optimally, either:

    * G will be achieved with probability p' at time t with $p' < p$; or
    * G will be achieved with probability p at time t' with $t' > t$.

  – The domain has *unreliable communication*, either in terms of transmission reliability or bandwidth limits. In particular:

    * If an agent $a_i \in A$ sends a message m intended for agent $a_j \in A$, then m arrives with some probability $q < 1$; or
    * Agent a_i can only receive x messages every y time units.

In the extreme, if $q = 0$ or if $x = 0$, then the periods of full communication are interleaved with periods of *no* communication, requiring the agents to act completely autonomously. In all cases, there is a cost to *relying* on communication. If agent a_i cannot carry on with its action until receiving a message from a_j, then the team's performance could suffer. Because of the unreliable communication, the message might not get through on the first try. And because of the dynamic, real-time nature of the domain, the team's likelihood of or efficiency at achieving G is reduced.

The soccer server provides a PTS domain since teams can plan strategies before the game, at halftime, or at other breakpoints; but during the course of the game, communication is limited.

3 Team Member Agent Architecture

At the core of the CMUnited-98 coordination mechanism is what we call the Locker-Room Agreement (Stone & Veloso 1999a). Based on the premise that agents can periodically meet in safe, full-communication environments, the locker-room agreement specifies how they should act when in low-communication, time-critical, adversarial environments. The locker-room agreement includes specifications of the flexible teamwork structure (Section 5.2) and the inter-agent communication paradigm (Section 5.4).

Our team member agent architecture is suitable for PTS domains. Individual agents can capture locker-room agreements and respond to the environment, while acting autonomously. Based on a standard agent paradigm, our team member agent 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 limited communication. Figure 1 shows the functional input/output model of the architecture.

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 interpreted 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 and the inter-agent communication protocols, if any. The locker-room agreement is accessible only to internal behaviors.

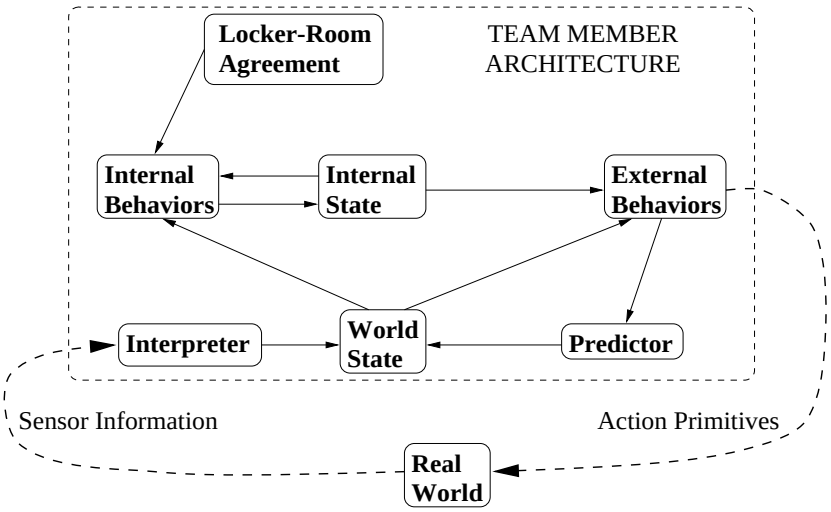

Figure 1: A functional input/output model of the team member agent architecture for PTS domains.

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. A window or distribution of past world states could also be stored as a part of the internal state. 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, and select the actions to send 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.

4 Agent Skills

The skills available to CMUnited-98 players include kicking, dribbling, ball interception, goaltending, defending, and clearing. The common thread among these skills is that they are all *predictive, locally optimal skills* (PLOS). They take into account predicted world models as well as predicted effects of future actions in order to determine the optimal primitive action from a local perspective, both in time and in space. In this section, we present dribbling and defending as examples of PLOS. Additional skills are described in detail in (Stone *et al.* 1999).

4.1 Dribbling

Dribbling is the skill which allows the player to move down the field while keeping the ball close to the player the entire time. The basic idea is fairly simple: alternate kicks and dashes so that after one of each, the ball is still close to the player.

Every cycle, the agent looks to see that if it dashes this cycle, the ball will be in its kickable area (and not be a collision) at the next cycle. If so, then the agent dashes, otherwise it kicks. A kick is always performed assuming that on the next cycle, the agent will dash. As an argument, the low-level dribbling code takes the angle relative to the direction of travel at which the player should aim the ball (see Figure 2). This is called the "dribble angle" and its valid values are $[-90, 90]$. Deciding what the dribble angle should be is discussed in Section 4.2.

First the predicted position of the agent (in 2 cycles) is calculated:

$$p_{new} = p_{current} + v + (v * pdecay + a)$$

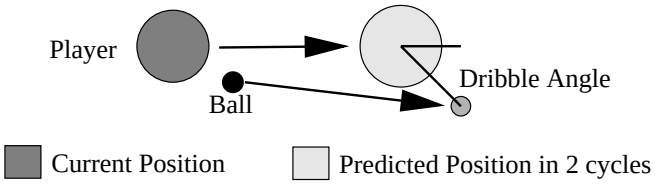

Figure 2: The basic dribbling skill.

where p_{new} is the predicted player position, $p_{current}$ is the current position of the player, v is the current velocity of the player, $pdecay$ is the server parameter `player_decay`, and a is the acceleration that a dash gives. The a value is usually just the dash power times the `dash_power_rate` in the direction the player is facing, but stamina may need to be taken into account.

Added to p_{new} is a vector in the direction of the dribble angle and length such that the ball is in the kickable area. This is the target position p_{target} of the ball. Then the agent gets the desired ball trajectory by the following formula:

$$traj = \frac{p_{target} - p_{ball}}{1 + bdecay}$$

where $traj$ is the target trajectory of the ball, p_{ball} is the current ball position, and $bdecay$ is the server parameter `ball_decay`. This process is illustrated in Figure 2.

If for some reason this kick can not be done (it would be a collision for example), then a turnball kick is done to get the ball in the right position. Then the next cycle, a normal dribble kick should work.

As can be seen from these calculations, the basic dribbling is highly predictive of the positions and velocities of the ball and player. It is also quite local in that it only looks 2 cycles ahead and recomputes the best action every cycle.

4.2 Smart Dribbling

The basic dribbling takes one parameter that was mentioned above: the dribble angle. Smart dribbling is a skill layered on the basic dribbling skill that decides the best dribble angle based on opponent positions. Intuitively, the agent should keep the ball away from the opponents, so that if an opponent is on the left, the ball is kept on the right, and vice versa.

The agent considers all nearby opponents that it knows about. Each opponent is given a "vote" about what the dribble angle should be; each opponent votes for the valid angle $[-90, 90]$ that is farthest from itself. For example, an opponent at 45 degrees, would vote for -90, while an opponent at -120 degrees would vote for 60. Each opponent's vote is weighted by the distance and angle relative to the direction of motion. Closer opponents and opponents more in front of the agent are given more weight.

4.3 Defending

CMUnited-98 agents are equipped with two different defending modes: opponent tracking and opponent marking. In both cases, a particular opponent player is selected as the target against which to defend. This opponent can either be selected individually or as a defensive unit via communication (the latter is the case in CMUnited-98).

In either case, the agent defends against this player by observing its position over time and position itself strategically so as to minimize its usefulness to the other team. When *tracking*, the agent stays between the opponent and the goal at a generous distance, thus blocking potential shots. When *marking*, the agent stays close to the opponent on the ball-opponent-goal angle bisector, making it difficult for the opponent to receive passes and shoot towards the goal. Defensive marking and tracking positions are illustrated in Figure 3.

When marking and tracking, it is important for the agent to have accurate knowledge about the positions of both the ball and the opponent (although the ball position isn't strictly relevant for tracking, it is used for the decision of whether or not to be tracking). Thus, when in the correct defensive position, the agent always turns to look at the object (opponent or ball) in which it is least confident of the correct position. The complete algorithm, which results in the behavior of doggedly following a particular opponent and glancing back and forth between the opponent and ball, is as follows:

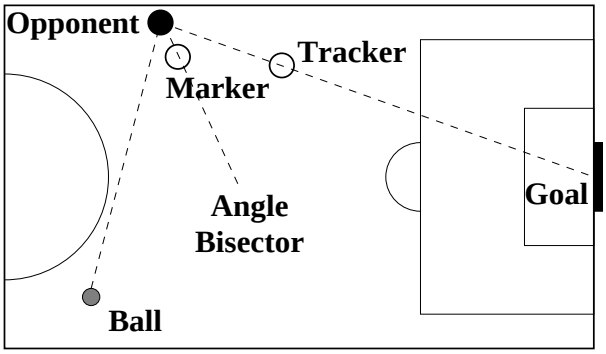

Figure 3: Positioning for defensive tracking and marking.

- If the ball position is unknown, look for the ball.
- Else, if the opponent position is unknown, look for the opponent.
- Else, if not in the correct defensive position, move to that position.
- else, look towards the object, ball or opponent, which has been seen less recently (lower confidence value).

This defensive behavior is locally optimal in that it defends according to the opponent's current position, following it around rather than predicting its future location. However, in both cases, the defensive positioning is chosen in anticipation of the opponent's future possible actions, i.e. receiving a pass or shooting.

5 Agent Coordination

If all players act individually — constantly chase the ball and try to kick towards the opponent goal — they will all get tired, there will be nowhere to pass, and the opponents will have free reign over most of the field. Building upon the innovations of the CMUnited-97 simulator team (Stone & Veloso 1998a), the CMUnited-98 team uses several complex coordination mechanisms, including reactive behavior modes, pre-compiled multi-agent plans and strategies, a flexible teamwork structure, a novel anticipatory offensive positioning scheme, and a sophisticated communication paradigm.

5.1 Behavior Modes

A player's top-level behavior decision is its behavior mode. Implemented as a rule-based system, the behavior mode determines the abstract behavior that the player should execute. For example, there is a behavior mode for the set of states in which the agent can kick the ball. Then, the decision of what to do with the ball is made by way of a more involved decision mechanism. On each action cycle, the first thing a player does is re-evaluate its behavior mode.

The behavior modes are:

Localize: Find own field location if it's unknown.

Face Ball: Find the ball and look at it.

Handle Ball: Used when the ball is kickable.

Active Offense: Go to the ball as quickly as possible. Used when no teammate could get there more quickly.

Auxiliary Offense: Get open for a pass. Used when a nearby teammate has the ball.

Passive Offense: Move to a position likely to be useful offensively in the future.

Active Defense: Go to the ball even though another teammate is already going. Used in the defensive end of the field.

Auxiliary Defense: Mark an opponent.

Passive Defense: Track an opponent or go to a position likely to be useful defensively in the future.

The detailed conditions and effects of each behavior mode are described in (Stone 1998). However, they will also become more clear in subsequent sections as the role-based flexible team structure is described in Section 5.2.

5.2 Roles, Formations, and Set-Plays

Like CMUnited-97, CMUnited-98 is organized around the concept of flexible formations consisting of flexible roles. Roles are defined independently of the agents that fill them: homogeneous agents (all except the goalie) can freely switch roles as time progresses. Each role specifies the behavior of the agent filling the role, both in terms of positioning on the field and in terms of the behavior modes that should be considered. For example, forwards never go into auxiliary defense mode and defenders never go into auxiliary offense mode.

A formation is a collection of roles, again defined independently from the agents. Just as agents can dynamically switch roles within a formation, the entire team can dynamically switch formations. After testing about 10 formations, the CMUnited-98 team ended up selecting from among 3 different formations. A standard formation with 4 defenders, 3 midfielders, and 3 forwards (4-3-3) was used at the beginnings of the games. If losing by enough goals relative to the time left in the game (as determined by the locker-room agreement), the team would switch to an offensive 3-3-4 formation. When winning by enough, the team switched to a defensive 5-3-2 formation.

As a part of the locker-room agreement, the team can also define multi-step multi-agent plans, or *set-plays*, 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.

For a detailed presentation of roles, formations, and set-plays, see (Stone & Veloso 1999a).

5.3 Strategic Positioning by Attraction and Repulsion (SPAR)

The flexible roles defined in the CMUnited-97 software were an improvement over the concept of rigid roles. Rather than associating fixed (x, y) coordinates with each position, an agent filling a particular role was given a range of coordinates in which it could position itself. Based on the ball's position on the field, the agent would position itself so as to increase the likelihood of being useful to the team in the future.

However, by taking into account the positions of other agents as well as that of the ball, an even more informed positioning decision can be made. The idea of *strategic position by attraction and repulsion* (SPAR) is one of the novel contributions of the CMUnited-98 software, both simulated and robotic (Veloso *et al.* 1999). SPAR was developed in parallel on our simulator and small-robot teams.

When positioning itself using SPAR, the agent uses a multi-objective function with attraction and repulsion points subject to several constraints. As described in detail in (Veloso *et al.* 1999) (in this same issue), we formulate the multi-objective function as a weighted single-objective function:

$$\max \sum_{i=1}^{n} w_{O_i}\, dist(P, O_i) + \sum_{i=1}^{n} w_{T_i}\, dist(P, T_i) - \; -w_B\, dist(P, B) - w_G\, dist(P, G)$$

In the simulator case, we use $w_{O_i} = w_{T_i} = x$, $w_B = 0$, and $w_G = x^2$. For example, the last term of the objective function above expands to $(dist(P, G))^2$.

One constraint in the simulator team relates to the position, or role, that the passive agent is playing relative to the position of the ball. The agent only considers locations that within one of the four rectangles, illustrated in Figure 5.3: the one closest to the position home of the position that it is currently playing. This constraint helps ensure that the player with the ball will have several different passing options in different parts of the field. In addition, players don't need to consider moving too far from their positions to support the ball.

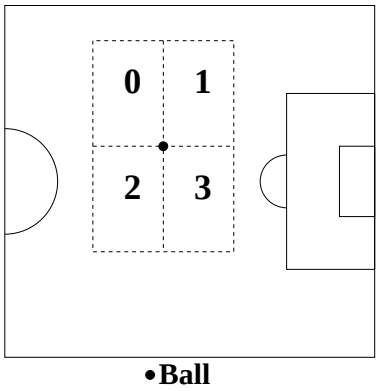

Figure 4: The four possible rectangles, each with one corner at the ball's location, considered for positioning by simulator agents when using SPAR.

Since this position-based constraint already encourages players to stay near the ball, we set the ball-attraction weighting function w_B to the constant function $y = 0$. In addition to this first constraint, the agents observe three additional constraints. In total, the constraints in the simulator team are:

- Stay in an area near home position;
- Stay within the field boundaries;
- Avoid being in an offsides position;
- Stay in a position in which it would be possible to receive a pass.

This last constraint is evaluated by checking that there are no opponents in a cone with vertex at the ball and extending to the point in consideration.

In our implementation, the maximum of the objective function is estimated by sampling its values over a fine-grained mesh of points that satisfy the above constraints.

Using this SPAR algorithm, agents are able to *anticipate* the collaborative needs of their teammates by positioning themselves in such a way that the player with the ball would have several useful passing options.

5.4 Communication

The soccer server provides a challenging communication environment for teams of agents. With a single, low-bandwidth, unreliable communication channel for all 22 agents and limited communication range and capacity, agents must not rely on any particular message reaching any particular teammate. Nonetheless, when a message does get through, it can help distribute information about the state of the world as well as helping to facilitate team coordination.

All CMUnited-98 messages include a certain amount of state information from the speaker's perspective. Information regarding object position and teammate roles are all given along with the confidence values associated with this data. All teammates hearing the message can then use the information to augment their visual state information.

The principle functional uses of communication in CMUnited-98 are

- To ensure that all participants in a set play are ready to execute the multi-step plan. In this case, since the ball is out of play, time is not a critical issue.
- To assign defensive marks. The captain of the defensive unit (the goaltender in most formations) determines which defenders should mark or track which opponent forwards. The captain then communicates this information periodically until receiving a confirmation message.

For a detailed specification of the communication paradigm as it was first developed for CMUnited-97, see (Stone & Veloso 1999a).

6 Layered Learning

Once the world model is successfully created, the agents must use it to respond effectively to the environment. As described in Section 3, internal behaviors update the internal state while external behaviors produce executable actuator commands. Spanning both internal and external behaviors, *layered learning* (Stone & Veloso 1998b; Stone 1998) is our bottom-up hierarchical approach to client behaviors that allows for machine learning at the various levels (Figure 5). The key points of the layered learning technique are as follows:

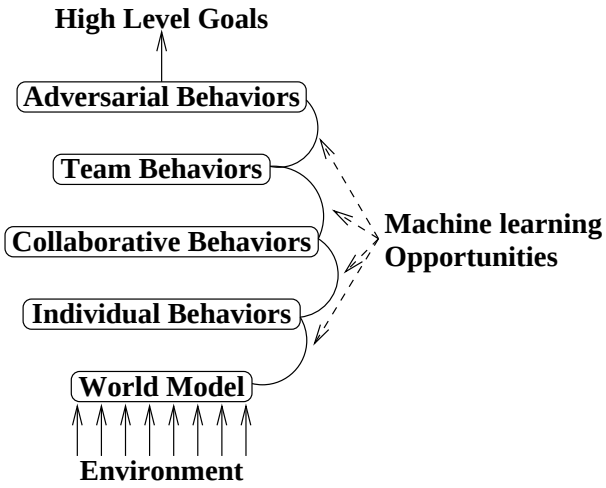

Figure 5: An overview of the Layered Learning framework in multi-agent domains. It is designed for use in domains that are too complex to learn a mapping straight from sensors to actuators. We use a hierarchical, bottom-up approach

- A bottom-up, hierarchical task decomposition is given.
- Machine learning exploits data to train and/or adapt. Learning occurs separately at each level.
- The output of learning in one layer feeds into the next layer.

Table 1 illustrates possible behavior levels within the robotic soccer domain.

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 (Stone & Veloso 1998b). 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 (Stone & Veloso 1999b). Adversarial actions are clearly opponent-dependent. Team collaboration and action selection can also benefit from adaptation to particular opponents.

7 Results

In order to test individual components of the CMUnited-98 team, it is best to compile performance results for the team with and without these components as we have done elsewhere (Stone & Veloso 1999a). However, competition against other, independently-created teams is useful for evaluating the system as a whole.

At the RoboCup-98 competition, CMUnited-98 won all 8 of its games by a combined score of 66–0, finishing first place in a field of 34 teams. Table 2 details the game results.

Opponent Name	Affiliation	Score
UU	Utrecht University, The Netherlands	22–0
TUM / TUMSA	Technical University Munich, Germany	2–0
Kasuga-Bitos II	Chubu University, Japan	5–0
Andhill'98	NEC, Japan	8–0
ISIS	Information Sciences Institute (USC), USA	12–0
Rolling Brains	Johannes Gutenberg-University Mainz, Germany	13–0
Windmill Wanderers	University of Amsterdam, The Netherlands	1–0
AT-Humboldt'98	Humboldt University of Berlin, Germany	3–0
TOTAL		66–0

Table 2: The scores of CMUnited-98's games in the simulator league of RoboCup-98. CMUnited-98 won all 8 games.

From observing the games, it was apparent that the CMUnited-98 low-level skills were superior in the first 6 games: CMUnited-98 agents were able to dribble around opponents, had many scoring opportunities, and suffered few shots against.

However, in the last 2 games, the CMUnited-98 strategic formations, communication, and ball-handling routines were put more to the test as the Windmill Wanderers (3rd place) and AT-Humboldt'98 (2nd place) also had similar low-level capabilities. In these games, CMUnited-98's abilities to use set plays to clear the ball from its defensive zone, to get past the opponents' offsides traps, and to maintain a cohesive defensive unit became very apparent.

Throughout the tournament, the CMUnited-98 software demonstrated its power as a complete multi-agent architecture in a real-time, noisy, adversarial environment.

8 Conclusion

For a more thorough understanding of the technical details contributing to the success of CMUnited-98, the reader is encouraged to study the detailed algorithmic descriptions provided in (Stone 1998) in conjunction with the CMUnited-98 source code (Stone *et al.* 1998). Other RoboCup researchers and multi-agent researchers in general should be able to benefit and build from the innovations represented therein.

The success of CMUnited-98 at RoboCup-98 was due to several technical innovations. Building on the contributions of CMUnited-97, including flexible formation, a novel communication paradigm, and machine learning modules, CMUnited-98 introduces strategic positioning using attraction and repulsion (SPAR). CMUnited-98 successfully combines low-level individual and high-level strategic, collaborative reasoning in a single multi-agent architecture.

References

Kitano, H.; Kuniyoshi, Y.; Noda, I.; Asada, M.; Matsubara, H.; and Osawa, E. (1997). RoboCup: A challenge problem for AI. *AI Magazine*, **18**(1):73–85.

Noda, I.; Matsubara, H.; Hiraki, K.; and Frank, I. (1998). Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, **12**:233–250.

Stone, P., and Veloso, M. (1998). The CMUnited-97 simulator team. In Kitano, H. (ed.), *RoboCup-97: Robot Soccer World Cup I*. (Berlin: Springer Verlag).

Stone, P., and Veloso, M. (1998). A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, **12**:165–188.

Stone, P., and Veloso, M. (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, . To appear.

Stone, P., and Veloso, M. (1999). Team partitioned, opaque transition reinforcement learning. In *Proceedings of the Second International Conference on Autonomous Agents*. (ACM Press), pp. 206–212.

Stone, P.; Veloso, M.; and Riley, P. (1998). CMUnited-98 source code. Accessible from http://www.cs.cmu.edu/~pstone/RoboCup/CMUnited98-sim.html.

Stone, P.; Veloso, M.; and Riley, P. (1999). The CMUnited-98 champion simulator team. In Asada, M., and Kitano, H. (eds), *RoboCup-98: Robot Soccer World Cup II*. (Berlin: Springer Verlag).

Stone, P. (1998). *Layered Learning in Multi-Agent Systems*. Ph.D. Dissertation, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Available as technical report CMU-CS-98-187.

Veloso, M.; Bowling, M.; Achim, S.; Han, K.; and Stone, P. (1999). CMUnited-98: RoboCup-98 samll-robot world champion team. *AI Magazine*, . To appear.