AI Magazine Volume 21 Number 3 (2000) (© AAAI)

The CMUNITED-99 Champion Simulator Team

Peter Stone, Patrick Riley, and Manuela Veloso

■ The CMUNITED-99 simulator team became the 1999 RoboCup simulator league champion by winning all 8 of its games, outscoring opponents by a combined score of 110–0. CMUNITED-99 builds on the successful CMUNITED-98 implementation but also improves on it in many ways. This article gives an overview of CMUNITED-99's improvements over CMUNITED-98.

The CMUNITED robotic soccer project is an ongoing effort concerned with the creation of collaborative and adversarial intelligent agents operating in real-time, dynamic environments. CMUNITED teams have been active and successful participants in all three international RoboCup (robot soccer world cup) competitions (Veloso, Pagello, and Kitano 2000; Asada and Kitano 1999; Kitano 1998). In particular, the CMUNITED-97 simulator team made it to the semifinals of the first RoboCup competition in Nagoya, Japan; the CMUNITED-98 simulator team won the second RoboCup competition in Paris, France (Stone, Veloso, and Riley 2000); and the latest CMUNITED simulator team won the third RoboCup competition in Stockholm, Sweden.[1]

The CMUNITED-99 simulator team is modeled closely after its two predecessors. Like CMUNITED-97 and CMUNITED-98, it uses layered learning and a flexible team structure (Stone 2000). In addition, many of the CMUNITED-99 agent skills, such as goal tending, dribbling, kicking, and defending, are closely based on the CMUNITED-98 agent skills. However, CMUNITED-99 improves on CMUNITED-98 in many ways. This article focuses on the research innovations that contribute to CMUNITED-99's improvements.

Coupled with the publicly available CMUNITED-99 source code (Stone, Riley, and Veloso 1999), this article is designed to help researchers involved in the RoboCup software challenge (Kitano et al. 1997) build on our successes. Throughout the article, we assume that the reader is familiar with the RoboCup simulator, or SOCCER SERVER (Noda et al. 1998). A detailed overview of the SOCCER SERVER, including agent perception and actuator capabilities, is given in Stone (2000).

The innovations touched on in this article are as follows: To partially automate the optimization of agents' individual skills, we created an offline agent training module, which is described in Offline Training. We greatly improved the agents' goal-scoring ability by introducing the ability to take advantage of models of teammates and opponents, particularly when near the opponent's goal, as presented in Using Models of Opponents and Teammates. The section entitled Layered Disclosure introduces the concept of layered disclosure, a key advance in our development methodology. Results and Conclusions summarizes CMUNITED-99's successful performance at RoboCup-99.

Offline Training

Individual agent skills, such as kicking and dribbling (running with the ball), are important prerequisites for team collaboration. For each of these skills, many parameters affect the details of the skill execution. For example, in the ball skill of dribbling, there are parameters that affect how quickly the agent runs, how far ahead it kicks the ball, and on which side of its body the agent keeps the ball while it dribbles.

The settings for these parameters usually involve a trade-off, such as speed versus safety or power versus accuracy. It is important to gain an understanding of what exactly these trade-offs are before "correct" parameter settings can be made.

We created a trainer client that connects to the server as an omniscient offline coach client. The trainer is responsible for three things: First is repeatedly setting up a particu-

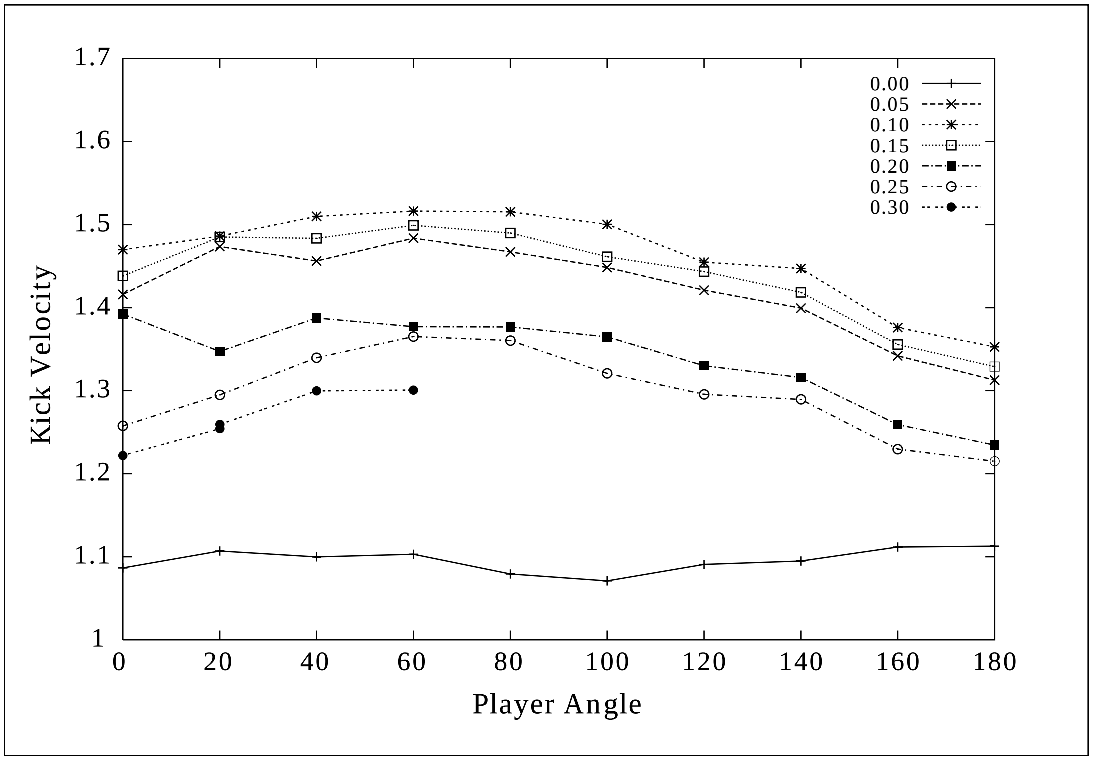


Figure 1. An Example of Training Data.

Two parameters are varied here: (1) the player angle (displayed on the *x* axis) and (2) the buffer around the player out of which the agent tries to keep the ball (the different lines). The goal is to maximize the kick velocity (displayed on the *y* axis).

lar training scenario. In the kicking skill, for example, the trainer repeatedly puts a single agent and the ball at a particular spot. The agent then tries to kick the ball as hard as possible toward a fixed target point. Second is recording the performance of the agent on the task. The trainer uses task-specific performance metrics to measure the agents' performance. In the kicking skill, for example, it records how quickly the ball is moving after being kicked, how accurately it goes in the intended direction, and how long it takes the agent to execute the kick. Third is iterating through different parameter settings. Using the server's communication mechanism, the trainer instructs the client on which parameter settings to use. The trainer records the performance of the agent for each set of parameter values.

Once the training scenario is set up, the system runs autonomously. Because most skills only involve one or two clients, we could afford to have the trainer iterate over many possible parameter values, taking several hours or days. The range of parameter settings to explore is determined a priori and searched exhaustively.

Once the trainer gathered the data, we depicted the results graphically and decided which parameters to use. An example for the kicking skill is shown in figure 1. The two parameters varied for the test shown are (1) the angle the agent is facing relative to the angle of the kick (the *x* axis) and (2) the distance buffer around the player outside of which the agent tries to keep the ball (the different lines).

Sometimes, the "optimal" parameter combination was fairly clear. For example, in figure 1, we were trying to maximize the kick velocity. Therefore, we selected a player angle of approximately 60 degrees and a buffer of 0.10. For some other skills, the data looked much noisier. In these cases, we could narrow our search down somewhat and get more data over the relevant parts of the parameter space.

We were sometimes limited by processing power in the breadth or resolution of the parameter space that we could examine. Given unlimited computing power, one might consider optimizing all the parameters in all the skills together in a full-game setting to capture their interactions. However, that not being the case, we were able to make progress by breaking the agents' behaviors into independent skills.

Skills that we tuned using the offline trainer include dribbling, kicking, goal tending, ball interception, and shooting.

Using Models of Opponents and Teammates

An important idea in many team ball sports such as soccer is the idea of a *breakaway*. Intuitively, a breakaway is when some number of offensive players get the ball and themselves past the defenders, leaving only perhaps a goalie preventing them from scoring.

When on a breakaway, an agent must decide when to shoot the ball. If the agent shoots too early, the goalie will have plenty of time to stop the ball. If the agent shoots too late, then the goalie might have time to get the ball before the kick is complete.

In CMUNITED-98, decisions about when to shoot were made based on one of three things: (1) the distance to the goal, (2) the number of opponents between the ball and the goal, and (3) a decision tree. All these methods have significant problems in this domain.

Distance to the goal completely ignores how the opponents are positioned. The number of opponents between the ball and the goal does not accurately reflect how good a position the defenders are in. Lastly, the decision tree was trained for passing (Stone 2000), so its performance on the related but different behavior of shooting is questionable. The decision tree uses over 100 input features and is trained offline to predict whether a pass to a given receiver will succeed or fail. Full details are available in Stone (2000).

CMUNITED-99 makes this decision in a more principled way by using a model of an "optimal" goalie. That is, we use a model of a goalie that reacts instantaneously to a kick, moves to exactly the right position to stop the ball, and catches with perfect accuracy.

The details of our method, called *ideal-model–based behavior outcome prediction* (IMBBOP), are introduced in Stone, Riley, and Veloso (2000). In short, the agent uses knowledge of the world dynamics, including the maximum speed that players can move, to predict how quickly the goal tender could get to the ball. Because this information does not depend on the behavior of any particular goal tender, it is effective against previously unseen opponents.

More specifically, we define the agent's behavior using the following predicates:

[blocking-point]: The point on the ball's path for which an optimal goalie heads

[ball-to-goalie-cycles]: The number of cycles for the ball to get to the blocking point

[goalie-to-ball-cycles]: The number of cycles for the goalie to get to the blocking point

[shot-margin]: Ball-to-goalie-cycles and goalie-to-ball-cycles description

During a breakaway, the agent shoots when either one of the following is true: (1) *shot margin cond*, when the shot margin gets below a certain threshold (one cycle in CMUNITED-99) or (2) *steal cond*, when the time that it would take for the goalie to proceed directly to the ball and steal it gets below a certain threshold (six cycles in CMUNITED-99). This time is again determined analytically using an optimal model of the goalie's movement capabilities.

Testing

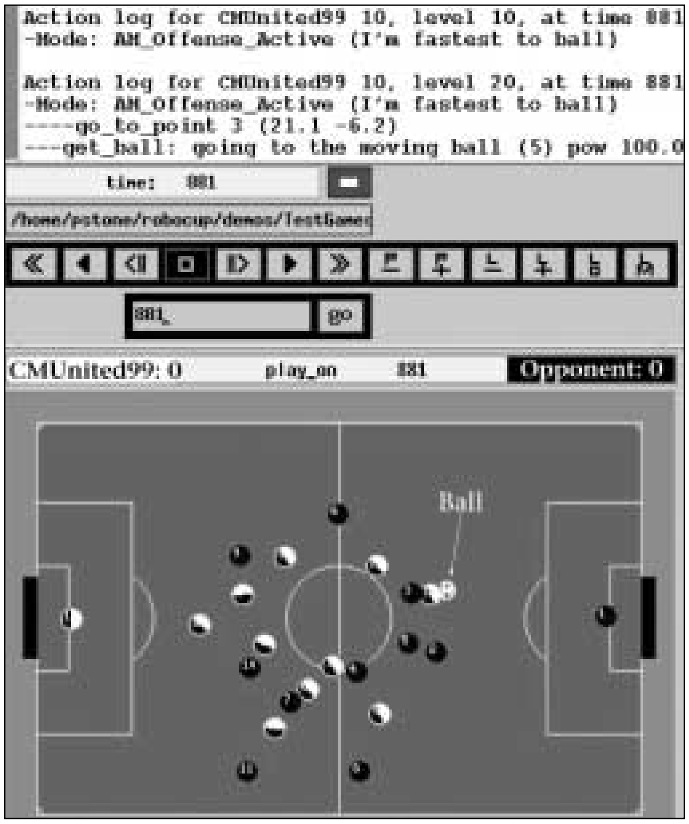
IMBBOP has proven to be useful to us in creating the CMUNITED-99 team of soccer-playing agents. Although CMUNITED-98 could rarely score when playing against itself (roughly one goal every three games), CMUNITED-99 scores about nine goals a game when playing against CMUNITED-98.

Because there were several improvements over CMUNITED-98 incorporated into CMUNITED-99, it is usually difficult to isolate a single change as being responsible for the team's overall improvement. However, in this case, there is clear evidence that incorporating IMBBOP into the agents' breakaway strategy is itself enough to lead to a significant improvement in the team's performance.

To demonstrate this claim, we played five versions of CMUNITED-99 against the CMUNITED-98 team. The only difference among these five versions was that their agents used the five different breakaway strategies: (1) Use both models to determine when to shoot conditions (shot margin cond and steal cond). (2) Use only the stealing ball model (condition steal cond). (3) Use only the shot-margin model (condition shot margin cond). (4) Shoot as soon as within 17 meters of the goal. (5) Shoot as soon as within 25 meters of the goal.

Each version played nine 10-minute games against CMUNITED-98. Table 1 displays the mean goals for each game scored by each of these versions as well as the standard deviation. CMUNITED-98 never scored a goal.

The three strategies (1–3) using some form of IMBBOP all performed significantly better than the two (4–5) that do not. Note that the CMUNITED-98 team used breakaway strategy 4 (always shooting from 17 meters). Although breakaway strategy 2, which only uses one of



```
Action log for CMUnited99 10, level 10, at time 881
-Mode: AM_Offense_Active (I'm fastest to ball)

Action log for CMUnited99 10, level 20, at time 881
-Mode: AM_Offense_Active (I'm fastest to ball)
----go_to_point 3 (21.1 -6.2)
---get_ball: going to the moving ball (5) pow 100.0
```

Figure 2. The Layered Disclosure Tool.

The terminal window at the top is displaying the high-level information for the agent with the ball (number 10 on the light-colored team) at the instant shown in the graphic display of the game. At level 10, only the agent's active mode (offense active) is shown. At level 20, information about its high-level actions is also included. In this case, the agent is trying to intercept the moving ball at a specific point.

the two types of opponent models, outperforms strategy 1, which uses both, the result is only of borderline significance. In addition, as noted earlier, each strategy will work against some specific goalie. When testing against different goalie types, we found that breakaway strategy 1 was most effective overall.

Because the RoboCup tournaments do not provide controlled testing environments, we cannot make any definite conclusions based on the competitions. However, when watching the games during RoboCup-99, we noticed many goals scored as a result of well-timed shots and passes near the opponent's goal.

Layered Disclosure

A perennial challenge in creating and using complex autonomous agents is following their choices of actions as the world changes dynamically and understanding why they act as they do. To this end, we introduce the concept of *layered disclosure* (Riley, Stone, and Veloso 2000) by which autonomous agents include in their architecture the foundations necessary to allow them to disclose to a person on request the specific reasons for their actions. The person can request information at any level of detail and either retroactively or while the agent is acting.

In developing layered disclosure, we began with the assumption that agents' actions and the actual states of the world over time are generally observable. However, several agent characteristics are generally unobservable, including sensory perceptions, internal states (current role in team, current task assignment, and so on), perceived current world states, and reasoning processes.

The goal of layered disclosure is to make these unobservable characteristics observable. Furthermore, to avoid being overwhelmed with data, the observer must be able to probe into the agent at an arbitrary level of detail or abstraction.

There are four main steps to realizing this goal: (1) The developer must organize the agent's perception-cognition-action process in different levels of detail. (2) The agent must store a log of all relevant information from its internal state, world model, and reasoning process. (3) This log must be synchronized with a recording of the observable world (or generated in real time). (4) An interface is needed to allow the developer to probe a given agent's

	Breakaway Strategy				
Goals/Game	1	2	3	4	5
Mean	8.9	10.6	8.6	3.6	3.6
Standard Deviation	+/– 1.5	+/– 1.3	+/– 2.6	+/– 1.4	+/– 1.0

Table 1. Goals Scored by CMUNITED-99 against CMUNITED-98 When Using the Different Breakaway Strategies.

Each trial represents nine 10-minute games. CMUNITED-98 never scored.

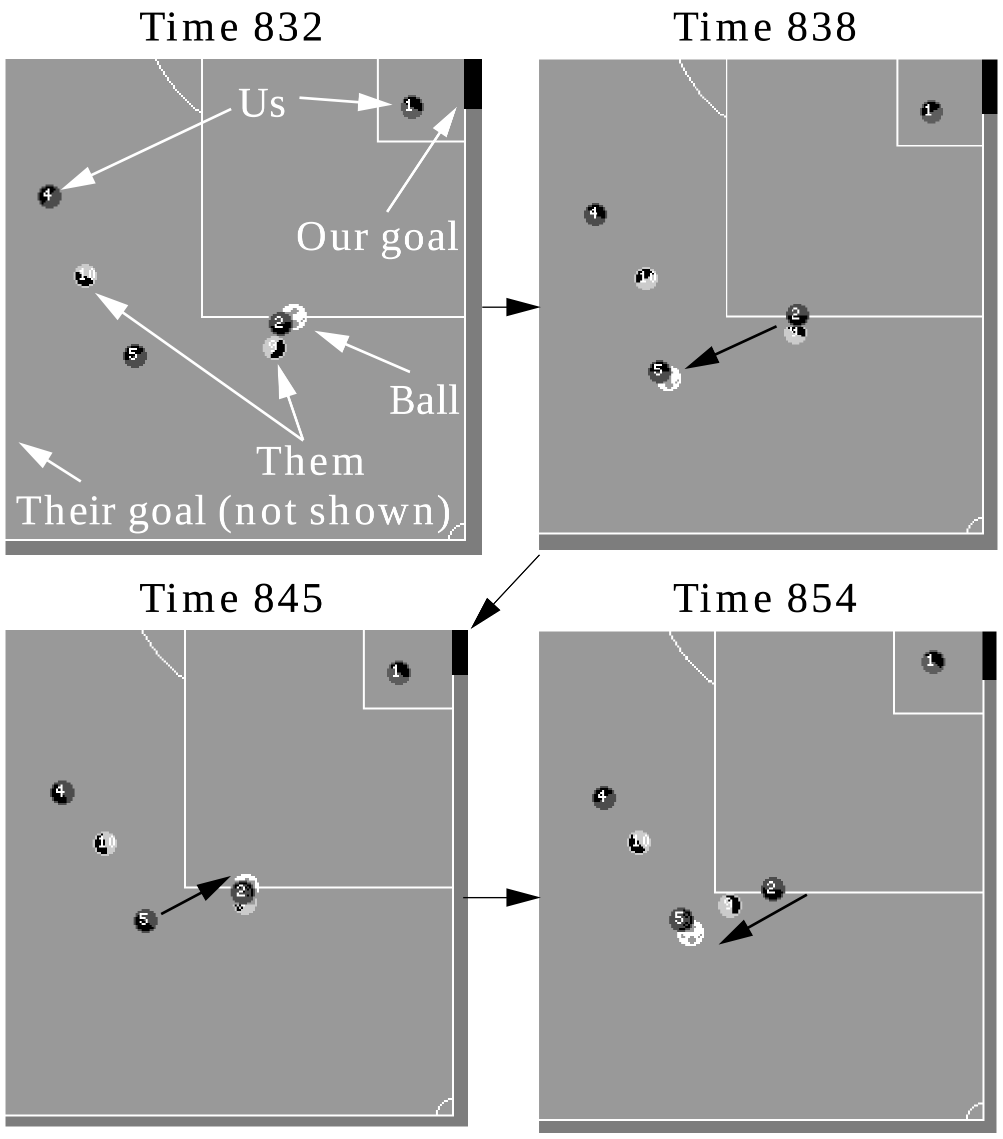


Figure 3. Undesired Passing Behavior.

internal reasoning at any time and any level of detail.

Layered disclosure was a significant part of the development of CMUNITED-99 and led to many of the improvements in the team over CMUNITED-98. During the course of a game, our agents store detailed records of selected information in their perceived world states, their determination of their short-term goals, and their selections of which actions will achieve these goals, along with any relevant intermediate decisions that lead to their action selections. After the game is over, it can be replayed using the standard log-player program

```
Action log for CMUnited99 2, level 30, at time 831
-Mode: AM With Ball
--Handling the ball
----OurBreakaway() == 0
---handle ball: need to clear
---clear ball: target ang == -93.0
-----starting kick to angle -93.0, translated to point (-6.4, -34.0)


Action log for CMUnited99 5, level 50, at time 838
------- Invalidating ball vel :0.36 > 0.36, thought vel was (1.73, -0.70)
-------Position based velocity estimating:
       gpos (-32.1 -23.4), prev seen pos (-33.5 -23.1)
---------Sight 838.0: B team:   opp:  9
-Mode: AM With Ball
--Handling the ball
----OurBreakaway() == 0
-----CanDribbleTo (-22.05, -20.52): TRUE No players in cone
---handle ball: dribbling to goal (2) -- have room


Action log for CMUnited99 2, level 20, at time 845
-Mode: AM With Ball
--Handling the ball
---handle ball: need to clear
---clear ball: target ang == 24.0
----starting kick to angle 24.0, translated to point (-16.5, -34.0)
----kick ball: starting kick to angle 24.0



Action log for CMUnited99 5, level 20, at time 850
-Mode: AM With Ball
--Handling the ball
---handle ball: dribbling to goal (2) -- have room
```

Figure 4. Layered Disclosure Information for the Passing Example (Boldface Has Been Added for Emphasis).

that comes with the SOCCER SERVER. Our layered disclosure module, implemented as an extension to this log player, makes it possible to inspect the details of an individual player's decision-making process at any point. Figure 2 shows our robotic soccer layered disclosure interface.

Layered disclosure has two main uses: (1) as a debugging tool for agent development in complex environments and (2) as a vehicle for interactive agent control. In the remainder of this section, we provide an example illustrating the usefulness of layered disclosure.

Discovering Agent Beliefs

When observing an agent team performing, it is tempting, especially for a person familiar with the agents' architectures, to infer high-level beliefs and intentions from the observed actions. Sometimes, such inferences can be helpful for predicting future events in the world, but misinterpretation is a significant danger.

Consider the example in figure 3. Here, two defenders seem to pass the ball back and forth while quite close to their own goal. In general, this sort of passing back and forth in a short time span is undesirable, and it is exceptionally dangerous near the agents' own goal. Using the layered disclosure tool, we get the information displayed in figure 4. Note that each dash represents five levels.

First, we see that both times that player 2 was in control of the ball, it was trying to clear it (just kick it away from the goal), not pass to player 5. Given the proximity of the goal and opponents, clearing is a reasonable behavior here. If a teammate happens to intercept a clear, then our team is still in control of the ball. Therefore, we conclude that this agent's behavior matches what we want and expect.

Next, we can see that both times that player 5 was trying to dribble toward the opponent's goal, it controlled the ball. There are no opponents immediately around it, and the path on the way to the goal is clear. This agent's intention is certainly reasonable.

However, player number 5 does not perform as it intended. Rather than dribbling forward with the ball, it kicked the ball backwards, pointing to some problem with the dribbling behavior. As we go down in the layers, we see that the agent invalidated the ball's velocity; that is, it thought the ball's observed position was so far off its predicted position that the agent's estimate for the ball's velocity could not possibly be right. The agent then computed a new estimate for the ball's velocity based on its past and current positions.

Given this estimation of the ball's velocity (which is crucial for accurate ball handling), we need to look further into how this velocity is estimated. Also, we can compare the estimate of the velocity to the recorded world state. In the end, we find that the ball collided with the player. Therefore, it was invalid to estimate the ball's velocity based on position. In fact, this realization led us to more careful application of this velocity estimation technique.

In this case, inferring the intentions of the players was extremely challenging given their behaviors. Without layered disclosure, the natural place to look to correct this undesirable behavior would have been in the passing decisions of the players. It would have been difficult or impossible to determine that the problem was with the estimation of the ball's velocity.

We envision that layered disclosure will continue to be useful in the RoboCup simulator and other agent development projects, particularly those with complex agents acting in complex, dynamic environments. We also plan to begin using layered disclosure in interactive semiautonomous agent-control scenarios.

Opponent	Affiliation	Score (CMU Opp.)
ULM SPARROWS	University of Ulm, Germany	29–0
ZENG99	Fukui University, Japan	11–0
HEADLESS CHICKENS III	Link University, Sweden	17–0
OULU99	University of Oulu, Finland	25–0
11MONKEYS	Keio University, Japan	8–0
MAINZ ROLLING BRAINS	University of Mainz, Germany	9–0
MAGMA FREIBURG	Freiburg University, Germany	7–0
MAGMA FREIBURG	Freiburg University, Germany	4–0
Total		110–0

Table 2. The Scores of CMUNITED-99's Games in the Simulator League of RoboCup-99.
CMUNITED-99 won all 8 games, finishing in first place out of 37 teams.

Results and Conclusions

The third international RoboCup championship, RoboCup-99, was held on 8 July to 4 August 1999 in Stockholm, Sweden, in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence (Veloso, Pagello, and Kitano 2000). As the defending champion, the CMUNITED-98 simulator team was entered in the competition. Its code was left unaltered from that used at RoboCup-98 except for minor changes necessary to update to version 5 of the SOCCER SERVER. Server parameter changes that reduced player size, speed, and kickable area required adjustments in the CMUNITED-98 code. However CMUNITED-98 did not take advantage of additions to the players' capabilities, such as the ability to look in a direction other than straight ahead (simulation of a neck).

The CMUNITED-98 team became publicly available soon after RoboCup-98 so that other people could build on our research. Thus, we expected there to be several teams at RoboCup-99 that could beat CMUNITED-98, and indeed there were. Nonetheless, CMUNITED-98 performed respectably, winning 3 games, losing 2, tying 1, and outscoring its opponents by a combined score of 29–3.

Meanwhile, the CMUNITED-99 team was even more successful at the RoboCup-99 competition than was its predecessor at RoboCup-98. It won all 8 of its games by a combined score of 110–0, finishing first in a field of 37 teams. Table 2 shows CMUNITED-99's game results.

Qualitatively, there were other significant differences between CMUNITED-98's and CMUNITED-99's performances. At RoboCup-98, several matches were quite close, with many offensive and defensive sequences for both teams. The CMUNITED-98 goalie performed quite well, stopping many shots. At RoboCup-99, CMUNITED-99's goalie only had to touch the ball 3 times over all 8 games. Only two teams (zeng99 and mainz rolling brains) were able to create enough of an offense to get shots on our goal. Improvements in individual skills, and a myriad of small improvements made possible by layered disclosure, greatly improved CMUNITED-99's midfield play over that of CMUNITED-98.

Another qualitative accomplishment of CMUNITED-99 was how closely its actions matched our ideas of what should be done. When watching games progress, we would often just be starting to say "Pass the ball!" or "Shoot it!" when the agents would do exactly that. This observation indicates that our development techniques, featuring the new layered disclosure, were such that we were able to refine behaviors in a complex domain to match our high-level expectations.

Looking forward, there are many improvements still to be made. Further, adapting models to opponents during play, as well as changing team strategy, is a promising future direction. We have done some experimentation with approaches to quick adaptation in complex domains such as robotic soccer (Riley and Veloso 2000). Other researchers associated with RoboCup are also looking in this direction, especially with the newly introduced coach agent.

Various software from the team is available, including binaries for the player and coach agents; full source code for the coach agent, the trainer agent, and the layered disclosure tool; and skeleton source code for the player agents, including the low-level skills.[2]

Notes

1. The CMUnited small-robot team is also a two-time RoboCup champion.

2. The CMUnited-99 source code is available at www.cs.cmu.edu/~pstone/RoboCup/CMUnited99-sim.html).

References

Asada, M., and Kitano, H., editors. 1999. *RoboCup-98: Robot Soccer World Cup II.* Lecture Notes in Artificial Intelligence 1604. Berlin: Springer Verlag.

Kitano, H., editor. *RoboCup-97: Robot Soccer World Cup I.* Berlin: Springer Verlag.

Kitano, H.; Tambe, M.; Stone, P.; Veloso, M.; Coradeschi, S.; Osawa, E.; Matsubara, H.; Noda, I.; and Asada, M. 1997. The RoboCup Synthetic Agent Challenge 97. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 24–29. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

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.

Riley, P., and Veloso, M. 2000. On Behavior Classification in Adversarial Environments. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000). Menlo Park, Calif.: American Association for Artificial Intelligence. Forthcoming.

Riley, P.; Stone, P.; and Veloso, M. 2000. Layered Disclosure: Revealing Agents' Internals. Paper presented at the Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL-2000), 7–9 July, Boston, Massachusetts.

Stone, P. 2000. Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. In *Intelligent Robotics and Autonomous Agents*. Cambridge, Mass.: MIT Press.

Stone, P.; Riley, P.; and Veloso, M. 2000. Defining and Using Ideal Teammate and Opponent Agent Models. In Proceedings of the Twelfth Annual Conference on Innovative Applications of Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence. Forthcoming.

Stone, P.; Veloso, M.; and Riley, P. 2000. CMUNITED-98 Simulator Team. *AI Magazine* 21(1): 20–28.

Veloso, M.; Pagello, E.; and Kitano, H., editors. 2000. RoboCup-99: Robot Soccer World Cup III. Berlin: Springer Verlag. Forthcoming.

Peter Stone is a senior technical staff member in the Artificial Intelligence Principles Research Department at AT&T Labs Research. He received his Ph.D. in 1998 and his M.S. in 1995 from Carnegie Mellon University, both in computer science. He received his B.S. in mathematics from the University of Chicago in 1993. Stone's research interests include planning and machine learning, particularly in multiagent systems. His doctoral thesis research contributed a flexible multiagent team structure and multiagent machine-learning techniques for a team operating in real-time noisy environments in the presence of both teammates and adversaries. He is currently continuing his investigation of multiagent learning at AT&T Labs. His e-mail address is pstone@research.att.com.

Patrick Riley is a Ph.D. student at Carnegie Mellon University. His research interests are mostly in dealing with adversaries, from theoretical and applied perspectives, including opponent modeling, model selection, machine learning, and game theory. He has been involved in the simulated robotic soccer domain for more than two years. His e-mail address is pfr@cs.cmu.edu.

Manuela M. Veloso is associate professor of computer science at Carnegie Mellon University (CMU). She received her Ph.D. in computer science from CMU in 1992. She received a B.S. in electrical engineering in 1980 and an M.Sc. in electrical and computer engineering in 1984 from the Instituto Superior Tecnico in Lisbon. Veloso's long-term research goal is the effective construction of teams of intelligent agents, where cognition, perception, and action are combined to address planning, execution, and learning tasks, in particular in uncertain, dynamic, and adversarial environments. Veloso has developed a team of robotic soccer agents in three different leagues that are RoboCup world champions: (1) simulation (1988), (2) CMU-built small-wheeled robots (1997 and 1998), and (3) Sony four-legged dog robots (1998). Veloso was awarded a National Science Foundation Career Award in 1995 and the Allen Newell Medal for Excellence in Research in 1997. Her e-mail address is mmv@cs.cmu.edu.