Layered Learning and Flexible Teamwork in RoboCup Simulation Agents

Peter Stone[1] and Manuela Veloso[2]

[1]AT&T Labs — Research
180 Park Ave., room A273
Florham Park, NJ 07932
pstone@research.att.com
http://www.research.att.com/~pstone

[2]Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
veloso@cs.cmu.edu
http://www.cs.cmu.edu/~mmv

Abstract. RoboCup was introduced as a challenge area at IJCAI-97. We have been actively pursuing research in this area and have participated in the RoboCup competitions, winning the RoboCup-98 and RoboCup-99 simulator competitions. In this paper, we report on the main technical issues that we encountered and addressed in direct response to the learning and teamwork challenges stated in the IJCAI-97 challenge paper. We describe "layered learning" in which off-line and on-line, individual and collaborative, learned robotic soccer behaviors are combined hierarchically. We achieve effective teamwork through a team member agent architecture that encompasses a "flexible teamwork structure." Agents are capable of decomposing the task space into flexible roles and can switch roles while acting. We report detailed empirical results verifying the effectiveness of the learned behaviors and the components of the team member agent architecture.

1 Introduction

The RoboCup Synthetic Agents Challenge 97 [4] specifies three challenges within the simulated robotic soccer server [5]: (i) learning of individual agents and teams; (ii) multi-agent team planning and plan-execution; and (iii) opponent modeling. Many researchers are working towards these and other challenges in the robotic soccer domain [3, 1], some specifically on learning, e.g. [2], and some specifically on teamwork structures, e.g. [11]. In conjunction with our own work, these pursuits have contributed to the success of the RoboCup challenge. In particular, we have addressed and successfully met the first two of the three specific challenges. Most of our specific contributions have been described in detail elsewhere. This paper serves to summarize our broad course of research within the context of the specific IJCAI challenge.

In the context of learning (challenge i), we have created three different learned behaviors and combined them hierarchically following the layered learning paradigm [8]. Given a hierarchical task decomposition, layered learning allows for learning at each level of the hierarchy, with learning at each level directly affecting learning at the next higher level.

As called for in the challenge, the three learned behaviors represent off-line skill learning by individual agents; off-line collaborative learning by teams of agents; and on-line collaborative and adversarial learning. First, individual

M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI 1856, pp. 495–508, 2000.
Springer-Verlag Berlin Heidelberg 2000

agents learn to intercept a moving ball by training a neural network off-line. Second, they use the learned ball-interception behavior as part of the training procedure for learning to evaluate whether a pass to a particular teammate will succeed or fail. This collaborative skill is trained off-line using the C4.5 decision tree training algorithm [6]. Third, the team of agents collectively learns an effective passing and shooting policy against a particular opponent using the on-line TPOT-RL multi-agent reinforcement learning method [7].

Within the context of the teamwork challenge (challenge ii), we characterize simulated robotic soccer as an example of a periodic team synchronization (PTS) domain [9]. PTS domains are domains in which periods of limited communication and time-critical action are interleaved with periods of safe, full communication. During the limited communication periods, agents need to act autonomously, while still working towards a common team goal. Time-critical environments such as robotic soccer require real-time response and therefore eliminate the possibility of heavy communication among team agents. However, in PTS domains, agents can periodically synchronize in a safe, full-communication setting.

We implement and test a team agent architecture suitable for PTS domains. Our team member agent architecture includes a flexible teamwork structure which allows agents to decompose the task space into flexible roles and allows them to smoothly switch roles while acting. Team organization is achieved by the introduction of a *locker-room agreement* as a collection of conventions followed by all team members. It defines agent roles, team formations, and pre-compiled multi-agent plans.

The remainder of this paper is organized as follows. Section 2 presents our learning experiments within the simulated robotic soccer domain. Section 3 provides details of our solution to the teamwork challenge. Section 4 describes our complete robotic soccer team which incorporates both learning and teamwork and concludes.

2 Learning Challenge

To address the learning challenge, we have created three different learned behaviors and combined them hierarchically following the layered learning paradigm. Section 2.1 describes layered learning and Section 2.3 presents our implementation within robotic soccer.

2.1 Layered Learning

Table 1 summarizes the principles of our layered learning paradigm.

Principle 1 Motivated by robotic soccer, layered learning is designed for domains that are too complex for learning a mapping directly from an agent's sensory inputs to its actuator outputs. Instead, the layered learning approach consists of breaking a problem down into several behavioral layers. At each layer, a concept needs to be acquired. A machine learning (ML) algorithm abstracts and solves the local concept-learning task.

1. A mapping directly from inputs to outputs is not tractably learnable.
2. A bottom-up, hierarchical task decomposition is given.
3. Machine learning exploits data to train and/or adapt. Learning occurs separately at each level.
4. The output of learning in one layer feeds into the next layer.

Table 1: The key principles of layered learning.

Principle 2 Layered learning uses a bottom-up incremental approach to hierarchical task decomposition. Starting with low-level behaviors, the process of creating new ML subtasks continues until reaching high-level strategic behaviors that deal with the full domain complexity. The appropriate behavior granularity and the aspects of the behaviors to be learned are determined as a function of the specific domain. The task decomposition in layered learning is not automated. Instead, the layers are defined by the ML opportunities in the domain.

Principle 3 Machine learning is used as a central part of layered learning to exploit data in order to *train* and/or *adapt* the overall system. ML is useful for training behaviors that are difficult to fine-tune manually. It is useful for adaptation when the task details are not completely known in advance or when they may change dynamically. In the former case, learning can be done off-line and frozen during actual task execution. In the latter, on-line learning is necessary: since the agent needs to adapt to unexpected situations, it must be able to alter its behavior even while executing its task. Like the task decomposition itself, the choice of machine learning method depends on the subtask.

Principle 4 The key defining characteristic of layered learning is that each learned layer directly affects the learning at the next layer. A learned subtask can affect the subsequent layer either by:

- pruning the set of training examples;
- providing the features used for learning; and/or
- determining the actions available;

All three possibilities are illustrated in our simulated robotic soccer implementation described below.

2.2 Formalism

Consider the learning task of identifying a hypothesis h from among a class of hypotheses H which map a set of state feature variables S to a set of outputs O such that, based on a set of training examples, h is most likely (of the hypotheses in H) to represent unseen examples.

When using the layered learning paradigm, the complete learning task is decomposed into hierarchical subtask layers $\{L_1, L_2, \ldots, L_n\}$ with each layer defined as

$$L_i = (\boldsymbol{F_i}, O_i, T_i, M_i, h_i)$$

where:

$\boldsymbol{F_i}$ is the input vector of state features relevant for learning subtask L_i. $\boldsymbol{F_i} = <F_i^1, F_i^2, \ldots>$. $\forall j$, $F_1^j \in S$.

$\boldsymbol{O_i}$ is the set of outputs among which to choose for subtask L_i. $O_n = O$.

$\boldsymbol{T_i}$ is the set of training examples used for learning subtask L_i. Each element of T_i consists of a correspondence between an input feature vector $\boldsymbol{f} \in \boldsymbol{F_i}$ and $o \in O_i$.

$\boldsymbol{M_i}$ is the ML algorithm used at layer L_i to select a hypothesis mapping $\boldsymbol{F_i} \mapsto O_i$ based on T_i.

$\boldsymbol{h_i}$ is the result of running M_i on T_i. h_i is a function from $\boldsymbol{F_i}$ to O_i.

As set out in Principle 2 of layered learning, the definitions of the layers L_i are given a priori. Principle 4 is addressed via the following stipulation. $\forall i < n$, h_i directly affects L_{i+1} in at least one of three ways:

- h_i is used to construct one or more features F_{i+1}^k.
- h_i is used to construct elements of T_{i+1}; and/or
- h_i is used to prune the output set O_{i+1}.

It is noted above in the definition of $\boldsymbol{F_i}$ that $\forall j$, $F_1^j \in S$. Since F_{i+1} can consist of new features constructed using h_i, the more general version of the above special case is that $\forall i, j$, $F_i^j \in S \cup_{k=1}^{i-1} O_k$.

Again, in layered learning, the task decomposition is assumed to be given a priori. Layered learning can, however, be combined with any algorithm for learning abstraction levels. In particular, let A be an algorithm for learning task decompositions within a domain. Suppose that A does not have an objective metric for comparing different decompositions. Applying layered learning on the task decomposition and quantifying the resulting performance can be used as a measure of the utility of A's output.

2.3 Layered Learning in Robotic Soccer

Table 2 illustrates our set of learned behavior levels within the simulated robotic soccer domain. We identify a useful low-level skill that must be learned before moving on to higher-level strategies. Then we build upon it to create higher-level multi-agent and team behaviors.

Layer	Behavior type	Learned behavior	Learning method	Training type
1	individual	ball interception	neural network	off-line
2	multi-agent	pass evaluation	decision tree	off-line
3	team	pass selection	TPOT-RL	on-line

Table 2: Examples of different behavior levels in robotic soccer and the learning methods used for the implemented layers in the simulated robotic soccer layered learning implementation.

L_1: Ball Interception — an individual skill. First, the agents learn a low-level individual skill that allows them to control the ball effectively. While executed individually, the ability to intercept a moving ball is required due to the presence of other agents: it is needed to block or intercept opponent shots or passes as well as to receive passes from teammates. As such, it is a prerequisite for most ball-manipulation behaviors. We chose to have our agents learn this behavior because it was easier to collect training data than to fine-tune the behavior by hand[1].

L_1 is defined as follows.

$F_1 = \{BallDist_t, BallAng_t, BallDist_{t-1}\}$: The agent learns what action to take based on the ball's current distance and angle from the defender, and the ball's distance a fixed time (250 msec.) in the past.

$O_1 = \{TurnAng\}$: The agent chooses an angle to turn such that it will be likely to intercept the ball.

T_1: The training procedure for ball interception involves a stationary forward repeatedly shooting the ball towards a defender in front of a goal. The defender collects training examples by acting randomly and noticing when it successfully stops the ball. Test examples are classified as saves (successful interceptions), goals (unsuccessful attempts), and misses (shots that went wide of the goal).

M_1 = a neural network: Ball interception is trained with a fully-connected neural network with 4 sigmoid hidden units and a learning rate of 10^{-6}. The weights connecting the input and hidden layers use a linearly decreasing weight decay starting at .1%. We use a linear output unit with no weight decay. The neural network was trained for 3000 epochs.

h_1 = a trained interception behavior: Table 3 shows the effect of the number of training examples on learned save percentage. With about 750 training examples, the defender is able to stop 91% of shots on goal (saves + goals: misses are omitted), a comparable save rate to that achieved when using an analytic ball interception behavior [8].

L_2: Pass Evaluation — a multi-agent behavior. Second, the agents use their learned ball-interception skill as part of the behavior for training a multi-agent behavior. When an agent has the ball and has the option to pass to a particular teammate, it is useful to have an idea of whether or not the pass will actually succeed if executed: will the teammate successfully receive the ball? Such an evaluation depends on not only the teammate's and opponents' positions, but also their abilities to receive or intercept the pass. Consequently, when creating training examples for the pass-evaluation function, we equip the intended pass recipient as well as all opponents with the previously learned ball-interception behavior, h_1. Again, we chose to have our agents learn the pass-evaluation capability because it is easier to collect training data than to construct it by hand.

L_2 is defined as follows.

[1] The learning was done in an early implementation of the soccer server (Version 2) in which agents did not receive any velocity information when seeing the ball.

Training Examples	Saves(%)	Goals(%)	$\frac{\text{Saves}}{\text{Goals+Saves}}$(%)
100	57	33	63
200	73	18	80
300	81	13	86
400	81	13	86
500	84	10	89
750	**86**	9	**91**
1000	83	10	89
4773	84	9	90

Table 3: The defender's performance when using neural networks trained with different numbers of training examples.

$\boldsymbol{F_2}$ **= a set of 174 continuous and ordinal features:** There are many features that could possibly affect pass evaluation. We encode a large set of attributes representing the relative positions of teammates and opponents on the field as well as statistical counts reflecting their relative positioning [8].

$\boldsymbol{O_2 = [-1, 1]}$ **:** A potential pass to a particular receiver is classified as a success with a confidence factor $\in (0, 1]$, a failure with a confidence factor $\in [-1, 0)$, or a miss $(= 0)$.

$\boldsymbol{T_2}$**:** The training procedure for pass evaluation involves a passer executing passes to randomly-placed teammates interspersed with randomly-placed opponents. The training scenario is illustrated within a screen shot of the soccer server in Figure 1. The dashed line indicates the region in which the teammates and opponents are randomly placed. *The intended pass recipient and the opponents all use the learned ball-interception behavior,* h_1. Trials are classified as successes (a teammate intercepts the ball), failures (an opponent intercepts the ball), and misses (no player intercepts the ball). When passing to a random teammate, 51% of passes are successful.

$\boldsymbol{M_2}$ **= C4.5:** To learn pass evaluation, we use the C4.5 decision tree training algorithm [6] with all of the default parameters. Decision trees are chosen over neural networks because of their ability to ignore irrelevant attributes.

$\boldsymbol{h_2}$ **= a trained pass-evaluating decision tree:** During testing, the trained decision tree returns a predicted classification as well as a confidence factor, resulting in a value between -1 and 1. Table 4 tabulates our results indicating that the trained decision tree enables the passer to choose successfully from among its potential receivers. Overall results are given as well as a breakdown by the passer's confidence prior to the pass. In this experiment, the passer is forced to pass even if it predicts failures for all 3 teammates. In that case, it passes to the teammate with the lowest likelihood of failure. 65% of all passes and 79% of passes predicted to succeed with high confidence are successful.

$\boldsymbol{L_3}$**: Pass Selection — a collaborative and adversarial team behavior.** Third, the agents use their learned pass-evaluation capability h_2 to create the input space and output set for learning pass selection. When an agent has the

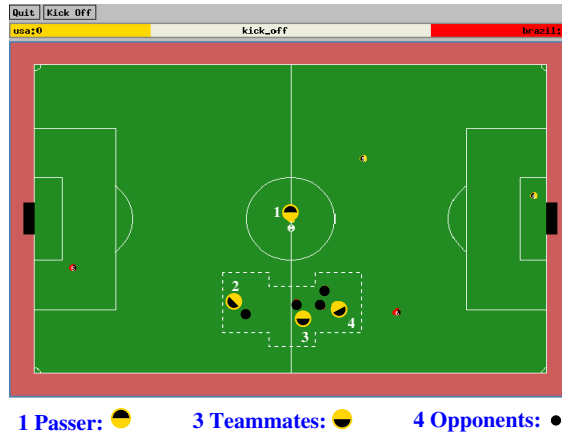

Fig. 1: The training scenario for pass evaluation. The dashed line indicates the region in which the teammates and opponents are randomly placed prior to each trial.

Result	Overall	Success Confidence: .8–.9	.7–.8	.6–.7
(Number)	(5000)	(1050)	(3485)	(185)
SUCCESS (%)	65	**79**	63	58
FAILURE (%)	26	15	29	31
MISS (%)	8	5	8	10

Table 4: The results of 5000 trials during which the passer uses the DT to choose the receiver. Results are given in percentages of the number of cases falling within each confidence interval (shown in parentheses).

ball, it must decide to which teammate it should pass the ball[2]. Such a decision depends on a huge amount of information including the agent's current location on the field, the current locations of all the teammates and opponents, the teammates' abilities to receive a pass, the opponents' abilities to intercept passes, teammates' subsequent decision-making capabilities, and the opponents' strategies. The merit of a particular decision can only be measured by the long-term performance of the team as a whole. Therefore, we drastically reduce the input space with the help of the previously learned decision tree, h_2: rather than considering the positions of all of the players on the field, only the pass evaluations for the possible passes to each teammate are considered.

L_3 is defined as follows.

$\boldsymbol{F_3 = \{PlayerPosition, O_2, O_2, O_2, \ldots\}}$**:** The input representation consists of one coarse geographical component and one action-dependent feature [10] for each possible pass. *The action-dependent features are precisely the result of h_2 executed for each possible receiver.*

[2] It could also choose to shoot. For the purposes of this behavior, the agents are not given the option to dribble.

$\boldsymbol{O_3 = \{shoot\} \cup \{Teammates\}}$**:** The result of a pass selection decision is either a shot on goal or a pass to a particular teammate.

$\boldsymbol{T_3}$**:** Training examples are gathered on-line by individual team members during real games. Each individual agent learns in a separate partition of F_3 according to its position on the field. Agents learn based on the observed long-term effects of their actions [7]. *For each particular action decision, the eligible members of* O_3 *are pruned based on* h_2*: only passes predicted to succeed are considered.*

$\boldsymbol{M_3}$ **= TPOT-RL:** For training pass selection, we use TPOT-RL [10], an on-line, multi-agent, reinforcement learning method motivated by Q-learning that is applicable in team-partitioned, opaque-transition domains such as simulated robotic soccer. We use the default parameters as reported in [10].

$\boldsymbol{h_3}$ **= a distributed pass-selection policy:** We test the pass-selection learning by directly comparing two teams with identical behaviors other than their pass-selection policies. Agents on both teams begin by passing randomly, but agents on one team adjust their behavior based on experience using TPOT-RL. The other agents continue passing randomly. Figure 2 demonstrates the effectiveness of the learned passing policies.

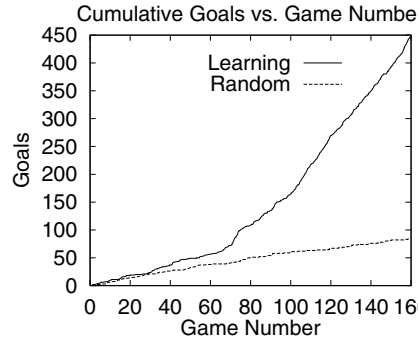

Fig. 2: Total goals scored by a learning team playing against a randomly passing team. The independent variable is the number of 10-minute games that have elapsed.

The learning methods used for each of the above behaviors are summarized in Table 2.

The three learned layers described above illustrate the principles of the layered learning paradigm as laid out in Section 2.1:

- The decomposition of the task into smaller subtasks enables the learning of a more complex behavior than would be possible if learning straight from the agents' sensory inputs.
- The hierarchical task decomposition is constructed in a bottom-up, domain-dependent fashion.
- Machine learning methods are chosen or created to suit the subtask in question. They exploit available data to train difficult behaviors (ball interception

and pass evaluation) or to adapt to changing/unforeseen circumstances (pass selection).
- Learning in one layer feeds into the next layer either by providing a portion of the behavior used for training (ball interception – pass evaluation) or by creating the input representation and pruning the action space (pass evaluation – pass selection).

3 Teamwork Challenge

To address the teamwork challenge, we characterize simulated robotic soccer as an example of a periodic team synchronization (PTS) domain and we create a general team member agent architecture suitable for PTS domains. Section 3.1 defines PTS domains and Section 3.2 lays out the agent architecture. Section 3.3 summarizes our implementation of a locker-room agreement in the robotic soccer domain. For the domain-independent formulation, see [9].

3.1 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.
- 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 a random 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.

3.2 Team Member Agent Architecture

At the core of our agents' coordination mechanism is the locker-room agreement [9]. 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. agreement can be hard-wired or it can be the result of deliberative automatic planning during the off-line phase of PTS domains. In our work so far, the locker-room agreement is hard-wired: we focus instead on the on-line phase.

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 3 shows the functional input/output model of the architecture.

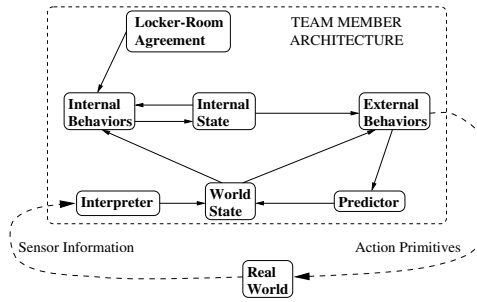

Fig. 3: A functional input/output model of the team member agent architecture for PTS domains.

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 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.

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.

3.3 Flexible Teamwork Structure

Within the team member agent architecture, our agents are equipped with a flexible teamwork structure that allows agents to decompose the task space into flexible roles and allows them to smoothly switch roles while acting. The teamwork structure consists of flexible positions (roles), dynamically changeable formations, and pre-defined, multi-agent set-plays. The structure is "flexible" in that agents can dynamically change their collective strategy and individual behaviors within this strategy in response to changing conditions. This section summarizes the robotic soccer implementation of our general flexible teamwork structure [9].

Flexible Positions In our multi-agent approach, the player positions itself flexibly in *anticipation* of where it will be useful to the team, either offensively or defensively. 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.

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.

Dynamically Changeable 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. The captain of a unit may have privileged decision-making responsibilities, for example to assign players to "mark" (stay close to) specific opponents.

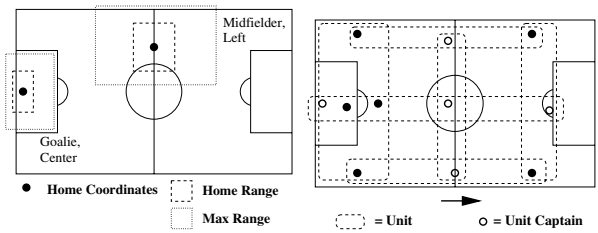

Fig. 4: (a) Different positions with home coordinates and home and max ranges. (b) Positions can belong to more than one unit.

We implemented several different formations, ranging from very defensive (8-2-0) to very offensive (2-4-4).[3] 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.

In the RoboCup simulator competitions, our agents 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.

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

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. The players maintain consistent beliefs as to the team's state via communication and via conventions encoded in the locker-room agreement [9].

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 multi-agent 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. 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.

3.4 Results

To test our flexible teamwork structure, we played a team using flexible positions and set-plays against one using rigid positions and no set-plays. Otherwise, the agents behaviors on the two teams were identical. Table 5 shows the results which clearly indicate the effectiveness of our teamwork structure.

(Game = 10 min.)	Flexible and Set-Plays	Default
Games won	34	1
Total goals	223	82
Avg. goals	5.87	2.16
Ball in own half	43.8%	56.2%

Table 5: Results when a flexible team plays against a rigid team. The flexible team won 34 out of 38 games with 3 ties.

To compare the different formations, we played each formation against every other several times. We used the results to help construct the formation-switching strategy used by our agents in competitions [7]. The cues for switching formations are functions of globally accessible variables (time remaining and score of the game) as defined in the locker-room agreement.

4 Conclusion

Layered learning and the team member agent architecture address the learning and teamwork components of the RoboCup Synthetic Agents Challenge 97. We leave the opponent modeling portion of the challenge as future work.

As well as being tested individually as reported above, our learning and teamwork techniques have been combined into a complete team of robotic soccer agents[4]. This team became the champion of the RoboCup-98 simulator competition, winning from among a field of 34 teams. Our team out-scored its opponents by a total of 66–0. Our subsequent team entered in RoboCup-99 also won that competition, which included 38 teams, outscoring its opponents by a total of 110–0 [7].

References

1. Minoru Asada and Hiroaki Kitano, editors. *RoboCup-98: Robot Soccer World Cup II.* Lecture Notes in Artificial Intelligence 1604. Springer Verlag, Berlin, 1999.
2. Minoru Asada, Shoichi Noda, Sukoya Tawaratumida, and Koh Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.
3. Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I.* Springer Verlag, Berlin, 1998.
4. Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The RoboCup synthetic agent challenge 97. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 24–29, San Francisco, CA, 1997. Morgan Kaufmann.
5. Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
6. J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.
7. Peter Stone. *Layered Learning in Multi-Agent Systems.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1998. Available as technical report CMU-CS-98-187.
8. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.
9. Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.
10. Peter Stone and Manuela Veloso. Team partitioned, opaque transition reinforcement learning. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 206–212. ACM Press, May 1999.
11. Milind Tambe, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem andGal A. Kaminka, Stacy C. Marsela, Ion Muslea, and Marcelo Tallis. Using an explicit model of teamwork in RoboCup-97. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 123–131. Springer Verlag, Berlin, 1998.
12. Christopher J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK, 1989.

[4] Robust individual agent skills also were essential for the team's performance. Thanks to Patrick Riley for his contributions to the team, developing effective individual low-level skills.