systems[2].

Prior to conducting the research reported in this chapter, we used neural networks to teach an agent to shoot a moving ball into a goal in a simulator based closely upon the Dynasim system [Sahota 96]. The simulator models the non-holonomic wheeled Dynamite robots [Sahota *et al.* 95]. In this scenario, we were able to use one neural network to time the agent's approach towards the ball, and another to aim the ball. The learned behavior successfully enabled the agent to redirect a moving ball with varying speeds and trajectories into specific parts of the goal. By carefully choosing the input representation to the neural networks so that they would generalize as much as possible, the agent was able to use the learned behavior in all quadrants of the field even though it was trained in a single quadrant [Stone and Veloso 98d].

Learning was applied in both simulated and real robotic, to hit a stationary ball into the goal using the concept of learning from easy missions [Asada *et al.* 94a]. By first placing the ball directly in front of the goal and then gradually moving it farther away, thus making the task harder, the learning time was reduced from exponential to linear in the size of the state space within an RL framework. The robots achieved 70% success in simulation and 40% success in the real world.

Another early example of learning an individual skill in the RoboCup soccer server involved using a neural network to enable an agent to learn when to shoot and when to pass [Matsubara *et al.* 96]. An agent was given the ball near the opponent's goal with a goaltender blocking the way and a teammate nearby. Based on the positions of the ball, the goaltender, and the teammate, the agent with the ball learned when it was best to shoot directly at the goal and when it was best to pass.

In a similar setup, using the same Dynasim-based simulator mentioned above, we used memory-based learning to allow an agent to learn when to shoot and when to pass the ball [Stone and Veloso 96a]. This learning was of an individual skill since the goaltender and the teammate used deterministic, fixed behaviors.

[2] A full discussion of related robotic soccer systems, including more recent learning approaches, appears in Chapter 9 (Section 9.2).

Chapter 6

Learning a Multi-Agent Behavior

In this chapter I present a learned multi-agent behavior, the second layer in our layered learning implementation of simulated robotic soccer agents. Pass evaluation—the estimation of whether or not a pass to a particular teammate will succeed—is a necessary but difficult capability with which to equip our agents. However, by collecting data and using it to train the agents, a successful evaluation function can be constructed [Stone and Veloso 98c]. As prescribed by the principles of layered learning, this training process incorporates the learned ball-interception skill (see Chapter 5).

Chapter 6 is organized as follows. Section 6.1 describes the motivation for, training of, and testing of the learned multi-agent behavior. Section 6.2 verifies that the off-line training can be successfully used in a real-time game situation. In Section 6.3 I describe our use of the learned pass-evaluation capability in a full team behavior. Sections 6.4 and 6.5 are devoted to discussion and related work.

6.1 Decision Tree Learning for Pass Evaluation

6.1.1 Setup and Training

Once young soccer players have learned how to control a ball, they are ready to use their skill to start learning how to make decisions on the field and playing as part of a team. Similarly, our simulated robotic soccer agents can use their learned ball-interception skill to exhibit a more complex behavior: passing. Passing requires action by two different agents. A *passer* must kick the ball towards the *receiver*, who must collect the ball. Since the receiver's task is identical to that of the defender in Chapter 5, the agents can (and do) use the same trained neural network.

Although the execution of a pass in the open field is not difficult given the receiver's ball-interception skill, it becomes more complicated in the presence of opponents that try to intercept the pass. In this section, we assume that the opponents are equipped with the same learned ball-interception skill as the receivers. The passer is faced with the task of assessing the likelihood that a pass to a particular teammate will succeed. For example, in Figure 6.1 teammate 2 may be able to receive a pass from the passer, while teammates 3 and 4 are much less likely to be able to do so. The function that our agents learn evaluates whether or not a pass to a given teammate is likely to succeed.

When deciding whether or not to make a pass, there are many world state features that may be relevant. When many features are available for a machine learning task, it can be very difficult to determine which ones are predictive of the target concept. Therefore, we use a learning method that is capable of determining for itself the features on which to focus. In particular, we use the C4.5 decision tree (DT) training algorithm [Quinlan 93].

In order to gather training data, like for the ball-interception behavior, we again define a constrained training scenario and use an omniscient, omnipotent agent to monitor the trials. The training process is illustrated in detail in Figures 6.1–6.5 including their accompanying captions. Table 6.1 specifies the training procedure.

1. The players are placed randomly within a region (Figure 6.1).
2. The passer announces its intention to pass (Figure 6.1).
3. The teammates reply with their views of the field when ready to receive (Figure 6.2).
4. The passer chooses a receiver randomly during training, or with a DT during testing (Figure 6.3).
5. The passer collects a large number of features of the training instance (see below).
6. The passer announces to whom it is passing (Figure 6.4).
7. The receiver and four opponents attempt to get the ball **using the learned ball-interception skill** (Figure 6.5).
8. The training example is classified as a *success* if the receiver manages to advance the ball towards the opponent's goal; a *failure* if one of the opponents clears the ball in the opposite direction; or a *miss* if the receiver and the opponents all fail to intercept the ball (Figure 6.5).

Table 6.1: The training procedure for learning pass evaluation.

Rather than restricting the number of features in the input representation used for learning, we capitalized on the DT's ability to filter out the irrelevant ones. Thus, we gather a total of 174 features for each training example, half from the passer's perspective and half from the receiver's. The features from the receiver's perspective are communicated to the passer before the passer has to decide to which player to pass.

The complete list of features—**all continuous or ordinal**—available to the DT are defined in Table 6.2. All of the features starting with "passer" are from the passer's perspective; features starting "receiver" are from the receiver's perspective. For example, `receiver-players-dist8-ang12` is the number of players that the receiver sees within a distance of 8 and angle of 12 from the passer. The features are defined in the following terms. If x and y are players, then:

- dist(x,y) = the distance between players x and y;

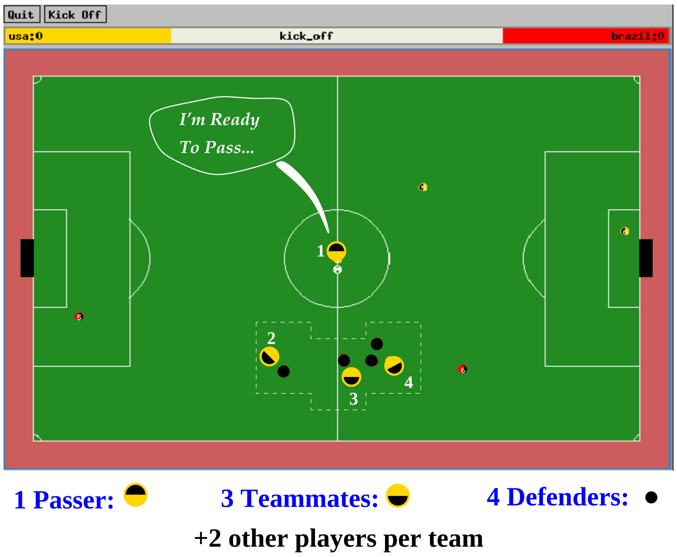

Figure 6.1: The pass evaluation training process, part 1. At the beginning of a trial, the passer is placed behind the ball. 3 teammates and 4 opponents are placed randomly within the region indicated by the dashed line, while 2 other players from each team are placed randomly on the field. In this and the following figures, the players involved in the play are enlarged for presentation purposes. When the passer sees that it has the ball, it announces its intention to pass. Its goal is to assess the likelihood of a pass to a given teammate succeeding.

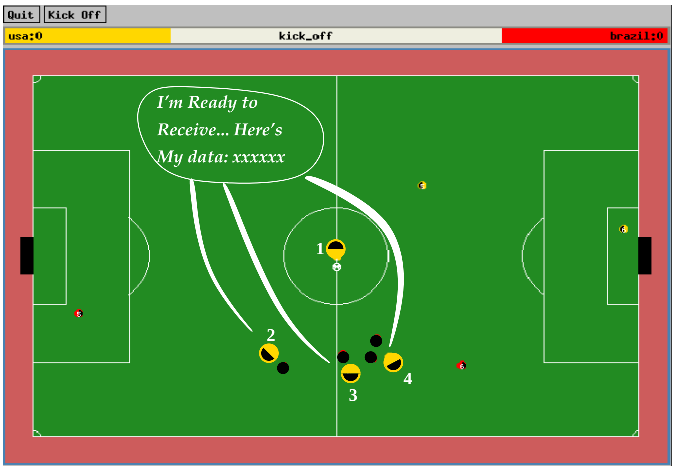

Figure 6.2: The pass evaluation training process, part 2. When the teammates are facing the ball, they tell the passer what the world looks like to them. The passer can use the transmitted data to help it assess the likelihood that each teammate would successfully receive a pass. The data includes distances and angles to the other players as well as some statistics regarding the numbers of players within given distances and angles.

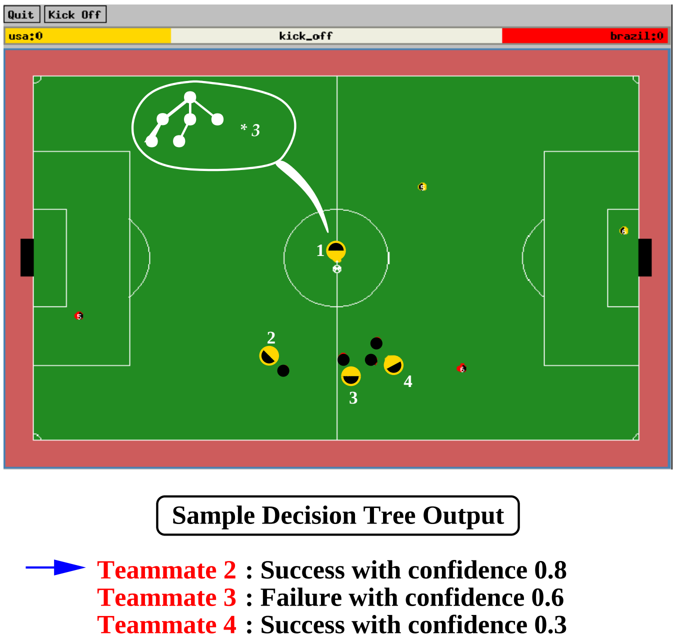

Figure 6.3: The pass evaluation training process, part 3. During training, the passer chooses its receiver randomly. During testing, it uses a DT to evaluate the likelihood that a pass to each of the teammates would succeed. It passes to the one most likely to successfully receive the pass (Teammate 2 in this case).

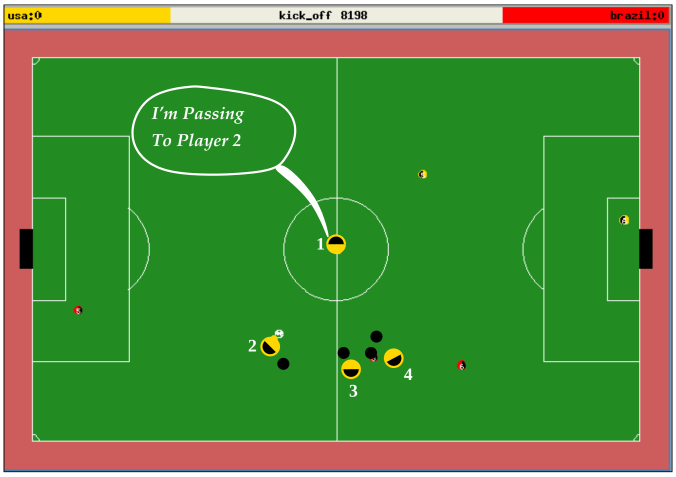

Figure 6.4: The pass evaluation training process, part 4. After choosing its receiver, the passer announces its decision so that the receiver knows to expect the ball and the other teammates can move on to other behaviors. In our experiments, the non-receivers remain stationary.

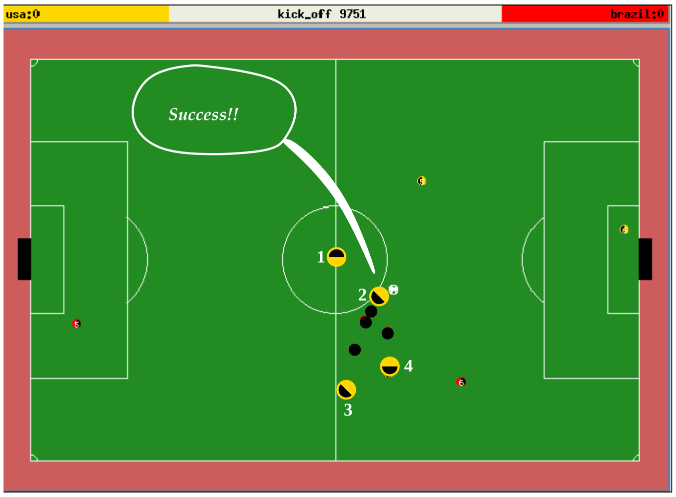

Figure 6.5: The pass evaluation training process, part 5. Finally, the receiver announces the result of the pass.

- ang(x,y) = the angle to player y from player x's perspective;
- rel-ang(x,y,z) = |ang(x,y) − ang(x,z)|;
- rel-dist(x,y,z) = |dist(x,y) − dist(x,z)|.

Even though the training examples do not include full teams of players, the features are defined such that they could be used in a game situation. The players whose positions are unknown create missing feature values.

Along with the ability to ignore irrelevant features, another strength of decision trees is their ability to handle missing features. Whenever fewer than the maximum number of players are visible, some features are missing. In addition, if the potential receiver is unable to successfully communicate the data from its perspective, all of the features from the receiver's perspective are missing.

The goal of learning is to use the feature values to predict whether a pass to the given teammate will lead to a success, a failure, or a miss. For training, we use standard off-the-shelf C4.5 code with all of the default parameters [Quinlan 93]. We gathered a total of 5000 training examples, 51% of which were successes, 42% of which were failures, and 7% of which were misses.

Training on this data produces a pruned tree with 87 nodes giving a 26% error rate on the training set. The resulting tree is shown in Figure 6.6. The first node in the tree tests for the number of opponents within 6 degrees of the receiver from the passer's perspective. If there are any such opponents, the tree predicts that the pass will fail. Otherwise, the tree moves on to the second node which tests the angle of the first opponent. Since the passer sorts the opponents by angle, the first opponent is the *closest* opponent to the receiver in terms of angle from the passer's perspective. If there is no opponent within 13 degrees of the receiver, the tree predicts success. Otherwise it goes on to deeper nodes in the tree.

Each leaf in the tree shown in Figure 6.6 includes a classification ("S" for success or "F" for failure—no misses are shown) followed by two numbers in parentheses: (N/E). As

- Distance and Angle to the receiver (2 features):
  - `passer-receiver-distance` = dist(passer,receiver).
  - `passer-receiver-angle` = ang(passer,receiver).
- Distance and Angle to other teammates sorted by angle from the receiver (18 features):
  - Sort the 9 other teammates such that ∀i,j, if i<j, then rel-ang(passer,teammate-i,receiver) < rel-ang(passer,teammate-j,receiver).
  - For i=1-9, `passer-teammate(i)-distance` = dist(passer,teammate-i).
  - For i=1-9, `passer-teammate(i)-angle` = ang(passer,teammate-i).
- Distance and Angle to opponents sorted by angle from the receiver (22 features):
  - Sort the 11 opponents such that ∀i,j, if i<j, then rel-ang(passer,opponent-i,receiver) < rel-ang(passer,opponent-j,receiver).
  - For i=1-11, `passer-opponent(i)-distance` = dist(passer,opponent-i).
  - For i=1-11, `passer-opponent(i)-angle` = ang(passer,opponent-i).
- Distance and Angle from the receiver to teammates sorted by distance (20 features):
  - Sort the 10 teammates (including the passer) such that ∀i,j, if i<j, then dist(receiver,teammate-i) < dist(receiver,teammate-j).
  - For i=1-10, `receiver-teammate(i)-distance` = dist(receiver,teammate-i).
  - For i=1-10, `receiver-teammate(i)-angle` = ang(receiver,teammate-i).
- Distance and Angle from the receiver to opponents sorted by distance (22 features):
  - Sort the 11 opponents such that ∀i,j, if i<j, then dist(receiver,opponent-i) < dist(receiver,opponent-j).
  - For i=1-11, `receiver-opponent(i)-distance` = dist(receiver,opponent-i).
  - For i=1-11, `receiver-opponent(i)-angle` = ang(receiver,opponent-i).
- Player distribution statistics from the passer's (45 features) and receiver's (45 features) perspectives:
  - For i=1-6, `passer-teammates-ang(i)` = |{k∈teammates | rel-ang(passer,k,receiver) ≤ i ∧ dist(passer,k) ≤ dist(passer,receiver)}|.
  - For i=1-6, `passer-opponents-ang(i)` = |{k∈opponents |rel-ang(passer,k,receiver) ≤ i ∧ dist(passer,k) ≤ dist(passer,receiver)}|.
  - For i=1-6, `passer-players-ang(i)` = `passer-teammates-ang(i)` + `passer-opponents-ang(i)`.
  - For i=1-3, j=1-3, `passer-teammates-dist(4i)-ang(4j)`= |{k∈teammates | rel-ang(passer,k,receiver) ≤ 4i ∧ rel-dist(passer,k,receiver) ≤ 4j}|.
  - For i=1-3, j=1-3, `passer-opponents-dist(4i)-ang(4j)`= |{k∈opponents | rel-ang(passer,k,receiver) ≤ 4i ∧ rel-dist(passer,k,receiver) ≤ 4j}|.
  - For i=1-3, j=1-3, `passer-players-dist(4i)-ang(4j)`= `passer-teammates-dist(4i)-ang(4j)` + `passer-opponents-dist(4i)-ang(4j)`.
  - For i=1-6, `receiver-teammates-ang(i)` = |{k∈teammates | rel-ang(receiver,k,passer) ≤ i ∧ dist(receiver,k) ≤ dist(passer,receiver)}|.
  - For i=1-6, `receiver-opponents-ang(i)` = |{k∈opponents |rel-ang(receiver,k,passer) ≤ i ∧ dist(receiver,k) ≤ dist(passer,receiver)}|.
  - For i=1-6, `receiver-players-ang(i)` = `receiver-teammates-ang(i)` + `receiver-opponents-ang(i)`.
  - For i=1-3, j=1-3, `receiver-teammates-dist(4i)-ang(4j)`= |{k∈teammates | rel-ang(receiver,k,passer) ≤ 4i ∧ rel-dist(receiver,k,passer) ≤ 4j}|.
  - For i=1-3, j=1-3, `receiver-opponents-dist(4i)-ang(4j)`= |{k∈opponents | rel-ang(receiver,k,passer) ≤ 4i ∧ rel-dist(receiver,k,passer) ≤ 4j}|.
  - For i=1-3, j=1-3, `receiver-players-dist(4i)-ang(4j)`= `receiver-teammates-dist(4i)-ang(4j)` + `receiver-opponents-dist(4i)-ang(4j)`.

Table 6.2: The complete list of 174 continuous and ordinal features available to the decision tree.

```
C4.5 [release 8] decision tree interpreter
-----------------------------------------
Decision Tree:

passer-opponents-ang6 > 0 : F (1266.0/251.3)
passer-opponents-ang6 <= 0 :
|   passer-opponent1-angle > 13 : S (1054.6/290.5)
|   passer-opponent1-angle <= 13 :
|   |   passer-teammates-ang6 <= 0 :
|   |   |   passer-receiver-distance <= 22 :
|   |   |   |   passer-opponent1-distance <= 20.9 :
|   |   |   |   |   passer-players-dist8-ang12 <= 3 : S (162.0/72.9)
|   |   |   |   |   passer-players-dist8-ang12 > 3 : F (15.0/5.8)
|   |   |   |   passer-opponent1-distance > 20.9 :
|   |   |   |   |   passer-opponent2-distance <= 21.1 :
|   |   |   |   |   |   passer-opponents-dist12-ang8 <= 1 :
|   |   |   |   |   |   |   passer-teammates-dist8-ang8 <= 1 :
|   |   |   |   |   |   |   |   passer-opponents-dist12-ang4 <= 0 :
|   |   |   |   |   |   |   |   |   passer-receiver-distance <= 20.3 : S (52.2/8.6)
|   |   |   |   |   |   |   |   passer-opponents-dist12-ang4 > 0 :
|   |   |   |   |   |   |   |   |   passer-opponents-dist4-ang12 <= 1 : S (60.5/20.0)
|   |   |   |   |   passer-opponent2-distance > 21.1 :
|   |   |   |   |   |   receiver-teammates-dist8-ang12 <= 1 : S (704.3/139.6)
|   |   |   passer-receiver-distance > 22 :
|   |   |   |   passer-opponent1-distance <= 23.1 :
|   |   |   |   |   passer-opponents-dist12-ang12 <= 0 :
|   |   |   |   |   |   receiver-players-dist8-ang12 <= 1 : S (87.5/44.7)
|   |   |   |   |   passer-opponents-dist12-ang12 > 0 :
|   |   |   |   |   |   passer-opponents-dist12-ang8 > 0 : F (191.0/63.0)
|   |   |   |   |   |   passer-opponents-dist12-ang8 <= 0 :
|   |   |   |   |   |   |   passer-opponents-dist4-ang12 > 1 : S (14.0/6.8)
|   |   |   |   |   |   |   passer-opponents-dist4-ang12 <= 1 :
|   |   |   |   |   |   |   |   passer-teammate1-distance <= 19.5 : S (15.0/6.8)
|   |   |   |   |   |   |   |   passer-teammate1-distance > 19.5 : F (234.0/91.7)
|   |   |   |   passer-opponent1-distance > 23.1 :
|   |   |   |   |   passer-opponents-dist12-ang12 <= 1 : S (665.9/259.2)
|   |   |   |   |   passer-opponents-dist12-ang12 > 1 :
|   |   |   |   |   |   passer-players-dist4-ang12 <= 1 : F (11.0/5.6)
|   |   |   |   |   |   passer-players-dist4-ang12 > 1 :
|   |   |   |   |   |   |   passer-opponents-dist4-ang8 <= 0 : S (49.0/19.9)
|   |   |   |   |   |   |   passer-opponents-dist4-ang8 > 0 :
|   |   |   |   |   |   |   |   passer-opponent2-distance <= 23.1 : F (85.0/26.5)
|   |   |   |   |   |   |   |   passer-opponent2-distance > 23.1 :
|   |   |   |   |   |   |   |   |   passer-opponents-dist12-ang12 <= 2 :
|   |   |   |   |   |   |   |   |   |   receiver-opponent3-angle <= 48 : F (9.6/2.2)
|   |   |   |   |   |   |   |   |   |   receiver-opponent3-angle > 48 : S (108.4/43.0)
|   |   passer-teammates-ang6 > 0 :
|   |   |   passer-teammates-ang5 > 0 : F (21.0/7.0)
```

Figure 6.6: The trained decision tree. Some subtrees with fewer cases covered have been removed for purposes of presentation. The features are described in Table 6.2. Predictions are indicated as “S” for success and “F” for failure. The numbers in parentheses indicate the predicted error rates for unseen cases at each leaf.

explained in [Quinlan 93, page 42], "N is the number of training examples covered by the leaf. E is just the number of predicted errors if a set of N unseen cases were classified by the tree." Fractional values of N arise when activations are split among branches of the tree due to missing attributes. C4.5 uses N and E to associate a confidence factor $\in [0,1]$ with each predicted classification [Quinlan 93].

We use C4.5's confidence factors to define a function Φ(passer,receiver) $\mapsto [-1,1]$. Assume that for a pass from passer to receiver the DT predicts class κ with confidence $\gamma \in [0,1]$. Then

$$\Phi(\text{passer}, \text{receiver}) = \begin{cases} \gamma & \text{if } \kappa = \text{S (success)} \\ 0 & \text{if } \kappa = \text{M (miss)} \\ -\gamma & \text{if } \kappa = \text{F (failure)} \end{cases}$$

6.1.2 Results

In order to test the DT's performance, we ran 5000 trials with the passer using the DT to choose the receiver in Step 4 of Table 6.1. All other steps including the placement of players, and consequently the example distribution, are the same during testing as during training. Since the DT returns a confidence estimate in its classification, the passer can choose the best receiver candidate even if more than one is classified as likely to be successful: it passes to the teammate with maximum $\Phi(passer, teammate)$.

We compiled results sorted by the DT's confidence in the success of the pass to the chosen receiver (see Table 6.3). The largest number of passes were classified as successes with confidence between .7 and .8, with another large portion classified as successes with confidence between .8 and .9. Overall, the success rate of 65% is much better than the 51% success rate obtained when a receiver is chosen randomly. However, this result is obtained under a condition of forced passing: the passer is *required* to pass the ball during all trials. In a game situation the passer would be given the option to dribble or shoot instead. Notice that if the passer wants to be fairly sure of success, it could pass only when the DT predicted success with confidence greater than .8. The resulting 79% success rate approaches the limit imposed by the success rate of the ball-interception skill. When the testing is repeated with no opponents to intercept the ball, the success rate is 86%.

		Φ(passer,receiver)					
		(success predictions)			(failure predictions)		
	Total	.8–.9	.7–.8	.6–.7	−.6–(−.7)	−.7–(−.8)	−.8–(−.9)
(Number)	(5000)	(1050)	(3485)	(185)	(34)	(3)	(39)
Success (%)	65	**79**	63	58	44	33	13
Failure (%)	26	15	29	31	53	67	79
Miss (%)	8	5	8	10	3	0	8

Table 6.3: The results of 5000 trials during which the passer uses the DT to choose the receiver. Overall results are given as well as a breakdown by the passer's confidence prior to the pass. 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. Results are given in percentages of the number of cases falling within each confidence interval (shown in parentheses).

With all the different features describing each training example, it is not obvious how to construct an analytic heuristic for the passer to use when choosing a receiver. For comparison purposes, we defined a heuristic by which the passer always passes to the closest of the three teammates.

Over 5000 trials, the closest teammate heuristic produces a success rate of 64%. Although this number compares favorably with the overall DT success rate, it is significantly lower than the 79% success rate the passer can achieve with the DT when given the option of not passing. Furthermore, the closest teammate heuristic gives no way of estimating the likelihood that a pass will succeed. It simply postulates that given a choice, the passer should pass to the closest teammate. Since the likelihood estimation is the true goal of our learning in this section, there is a clear advantage to using the DT method. As shown in Section 6.3, when deciding whether to pass, dribble, or shoot, the knowledge of whether or not a given pass is likely to succeed is extremely useful.

6.1.3 Summary

In this section, I demonstrated that a higher-level decision can be built upon the low-level skill learned in Chapter 5. Using a DT, our agents learn to judge the likelihood that a pass to a given receiver will be successfully received. This judgment represents the second layer in our layered learning implementation.

Before moving on to the third layer in Chapter 7, I first present some experiments that demonstrate the effectiveness of the learned ball-interception and pass-evaluation capabilities in game situations. Since they are both trained and tested off-line in limited types of situations, their applicability to more realistic scenarios has not yet been established. Section 6.2 presents the initial verification that encourages us to proceed. Section 6.3 presents extensive results demonstrating the utility of the learned behaviors in full game situations.

6.2 Using the Learned Behaviors

Once able to judge the likelihood that a pass will succeed, a human or simulated soccer player is ready to start making decisions in game-like situations. When considering what to do with the ball, the player can pass to a strategically positioned teammate, dribble, or shoot. To verify that the second level of our layered learning implementation can be incorporated into game-like situations, we implemented a play sequence that uses the passing decision described in Section 6.1.

In Figure 6.7 which illustrates the play sequence, the teammates executing the play sequence are shown as white circles and labeled A-G; opponents are shown as black circles. The play sequence is as follows:

- Player A starts with the ball in front of it and dribbles towards the opponent's goal.

- When it approaches within 15 of an opponent, it stops dribbling and prepares to pass to one of the two closest teammates: players B and C.

- In accordance with the protocol laid out in Table 6.1, it announces its intention to pass and gets responses from the two nearest players. It then uses the DT to decide which teammate is more likely to successfully receive the pass. In Figure 6.7, player A passes to teammate B (Φ(A,B) > Φ(A,C)).

- Player B and the adjacent opponent then both try to intercept the ball using the trained ball-interception skill. If the opponent gets the ball, it kicks it back towards the left goal and the play starts over. However, if player B gets the ball, it immediately kicks the ball to player D (if player A had passed to player C, the ball would have been sent to player E and the play would have continued symmetrically).

- Since player D is not covered, it can easily collect the ball and begin dribbling towards the goal. Using the same behavior as player A, player D stops dribbling when it approaches within 15 of a defender and chooses between passing to teammates F and G. Again, it uses the trained DT.

- If player F or G is able to get to the ball before the opponents, it immediately shoots towards the goal.

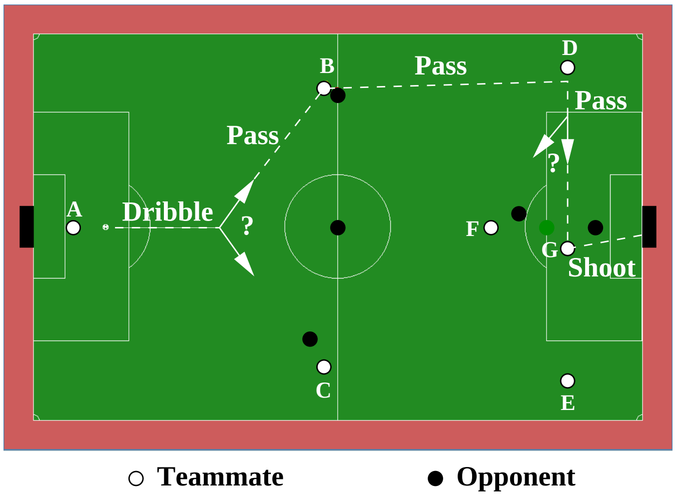

Figure 6.7: An illustration of the implemented play sequence. Players are emphasized for improved visibility. Teammates are labeled A–G. Every player uses at least one of the two learned behaviors: ball interception and pass evaluation.

We ran this play sequence several times in order to verify that the learned behaviors are both robust and reliable. Since the opponents are all equipped with the same ball-interception skill as the receivers, the opponents are sometimes able to break up the play. However, the fact that the attacking team can sometimes successfully string together three passes and a shot on goal when using the learned behaviors demonstrates that these behaviors are appropriate for game-like situations.

Indeed, in the following section, the learned behaviors are incorporated into a full team of soccer-playing agents. The power of the learned pass evaluation is tested by comparing teams that use it against teams that do not.

6.3 Scaling up to Full Games

As presented in Chapter 5 and above, the ball-interception and pass-evaluation behaviors are both trained and tested in limited, artificial scenarios which do not reflect the full range of game situations. In this section I extend these basic learned behaviors into a full multi-agent behavior that is capable of controlling agents throughout an entire game [Stone and Veloso 98e]. This multi-agent behavior is designed for the purposes of testing: it is not the same as the CMUnited-98 behavior specification laid out in Section 3.5.

Since the two learned behaviors defined thus far only apply when a player is in the vicinity of the ball, the player needs to have some other mechanism for acting when it does not have the ball. In addition, when it *does* have the ball, it must decide *when* to pass it: in some cases it may have enough time to execute the ideal pass; however, in other cases it may have to release the ball immediately in order to avoid losing it to an opponent player.

This section is organized as follows. Section 6.3.1 defines the mechanism by which the player with the ball chooses its action, either with the aid of the learned DT or without. In Section 6.3.2, I explain how our agents reason about the time available to execute a pass. Section 6.3.3 presents the full multi-agent behavior including behavior specifications for occasions when the player does not have the ball. Section 6.3.4 demonstrates the effectiveness of the learned pass-evaluation function in the context of full game situations via extensive empirical testing.

6.3.1 Receiver Choice Functions

Recall that the DT estimates the likelihood that a pass to a specific player will succeed. Thus, for an agent to use the DT in a game, the DT must be incorporated into a full *receiver choice function* (RCF). We define an RCF to be a function that determines what the agent should do when it has possession of the ball—when the ball is within `kickable_area` (see Section 2.2)—and it has the opportunity to choose a receiver to which to pass. The input and output of a receiver choice function are as follows.

Input. The input of an RCF is the agent's perception of the current state of the world. This perceived state includes both the agent's latest sensory perception and remembered past positions of currently unseen objects (see Section 3.2).

Output. The output of an RCF is an action from among the options *dribble, kick,* or *pass*, and a direction, either in terms of a player (i.e. towards teammate number 4) or in terms of a part of the field (i.e. towards the goal).

First, the RCF identifies a set of candidate receivers given the current world state. Then, the RCF selects a receiver or else indicates that the agent should dribble or kick the ball.

The set of candidate receivers is determined in part by the player's *position*, or role. Each player is assigned a particular position on the field, and the team remains in a constant formation (see Chapter 3). Throughout this section, the team being tested uses the 4-3-3 formation illustrated by the black team in Figure 6.8. The positions in this formation are a goaltender (G), a sweeper (S), three defenders—left (LD), center (CD), and right(RD)—three midfielders (LM, CM, and RM), and three forwards (LF, CF, and RF). The arrows emanating from the players indicate the positions to which each player considers passing when using the RCFs. The options are also apparent in Table 6.5. The position on the left of the field (LD, LM, and LF) consider symmetrical options to their counterparts on the right of the field. The goaltender (G) has the same options as the sweeper (S).

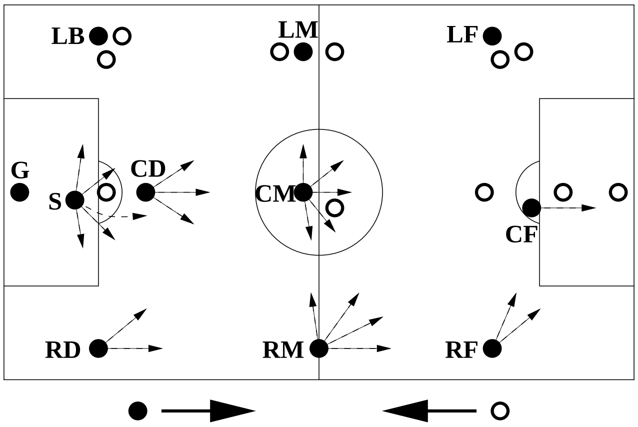

Figure 6.8: Player positions in the 4-3-3 formation used by the behaviors in this section. The black team, moving from left to right, has a goaltender (G), a sweeper (S), three defenders—left (LD), center (CD) right(RD)—three midfielders (LM, CM, and RM), and three forwards (LF, CF, and RF). The arrows emanating from the players indicate the positions to which each player considers passing when using the RCFs. The players on the left of the field (top of the diagram) consider symmetrical options to their counterparts on the right of the field. The goaltender has the same options as the sweeper. The white team has the same positions as the black, except that it has no players on its left side of the field, but rather two in each position on its right.

When a player is near its default position, it periodically announces its position to teammates; when a player leaves its position to chase the ball, it announces this fact and is no longer considered "in position." The agents determine which players are in which positions by listening to their teammates' announcements.

Notice that Figure 6.8 is the same as Figure 3.7, except that an opposing team is also depicted. The white team has the same positions as the black, except that it has no players on its left side of the field. Instead, it has two players in each position on the right side of the field.

Table 6.4 defines three RCFs, one of which uses the DT; the others are defined for the purposes of comparison.

- The PRW—*prefer right wing*—RCF uses a fixed ordering on the candidate receivers for each of the positions on the field. In general, defenders prefer to pass to the wings rather than forward; midfielders prefer to pass forward rather than sideways; and forwards tend to shoot. All players in the center of the field, as indicated by the name, prefer passing to the right rather than passing to the left. The ordered preference lists D_r of the different positions r are specified in Table 6.5. The RCF simply returns the most preferable candidate receiver according to this fixed ordering. Again, if no receivers are eligible, the RCF returns "dribble" or "kick." This RCF was our initial hand-coded behavior for use in games (at Pre-RoboCup-96).

Let agent $\alpha \in A$ have the ball at time t and be using the RCF. Assume agent α is in position (role) $\rho \in R$ (recall from Chapter 3 that A is the set of agents and R is the set of roles in the team's current formation). First, the RCF determines the set of candidate receivers $C_{\alpha,t} \subseteq A$. Each position $r \in R$ has a set of candidate receiver roles $C_r \subseteq R$, as indicated in Figure 6.8. For example, $C_{\mathrm{CM}} = \{\mathrm{LM,RM,LF,CF,RF}\}$. At time t, α's mapping from agents $A \mapsto R$ specifies which agent is playing which position. Let $a_{r,t}$ be the agent playing position r at time t.

1. Start by setting $C_{\alpha,t} = \{a_{r,t} | r \in C_\rho\}$.

2. Any potential receiver that is too close (closer than d_{min}) or too far away (farther than d_{max}) according to α's world state at time t is eliminated from consideration: set $C_{\alpha,t} = \{a \in C_{\alpha,t} | d_{min} \leq \mathrm{dist}(\alpha, a) \leq d_{max}\}$. We use $d_{min} = 10$ and $d_{max} = 40$.

3. Any player that is out of position (because it was chasing the ball) is eliminated from consideration. Let $A' \subseteq A$ be the set of agents that are currently away from their home positions as determined by the periodic announcements from all teammates. For instance, these agents might be chasing the ball. Set $C_{\alpha,t} = C_{\alpha,t} - A'$.

4. **IF** $\exists i \in$ opponents such that $\mathrm{dist}(\alpha, i) \leq d_{min}$ **THEN** let $C' \subseteq C_{\alpha,t}$ be the set of agents to which α cannot kick the ball directly without the ball hitting α: $C' = \{a \in C_{\alpha,t} | (|\mathrm{ang}(\alpha, a) - \mathrm{ang}(\alpha, \texttt{ball})|) > c_{ang}\}$. We use $c_{ang} = 130$. Set $C_{\alpha,t} = C_{\alpha,t} - C'$.

After having determined the set of candidate receivers $C_{\alpha,t}$, the RCF specifies which candidate receiver, if any, should receive the pass.

5. **IF** $C_{\alpha,t} = \emptyset$ **THEN**
   - **IF** $\exists i \in$ opponents such that $\mathrm{dist}(\alpha, i) \leq d_{min}$, **THEN** return *kick* to the opponent's goal.
   - **ELSE** return *dribble* to the opponent's goal.

6. **ELSE** ($C_{\alpha,t} \neq \emptyset$) **THEN** pass according to which RCF is being used:

PRW (*Prefer Right Wing*): Use a fixed ordering on the receivers $D_\rho \subseteq R = (d_1, d_2, \ldots)$ as specified in Table 6.5. Return *pass* to $c \in C_{\alpha,t}$ such that $c = a_{d_i,t} \wedge \forall a_{d_j,t} \in C_{\alpha,t}, j \geq i$.

RAND (*Random*): Choose randomly among the receivers. Return *pass* to some $c \in C_{\alpha,t}$.

DT (*Decision Tree*): Set $C_{\alpha,t} = \{a \in C_{\alpha,t} | \Phi(\alpha, a) > 0\}$.
IF $C_{\alpha,t} = \emptyset$ **THEN** return *kick* or *dribble* as in Step 5.
ELSE return *pass* to $c \in C_{\alpha,t}$ such that $\forall b \in C_{\alpha,t}, \Phi(\alpha, c) > \Phi(\alpha, b)$.

Table 6.4: Specification of three RCFs: PRW, RAND, and DT.

- The RAND—*random*—RCF is the same as the PRW RCF except that it chooses randomly from among all candidate receivers.

- As suggested by its name, the DT—*decision tree*—RCF uses the DT described in Section 6.1 to choose from among the candidate receivers. In particular, as long as the DT predicts a pass to at least one candidate would be successful, the DT RCF indicates that the passer should pass to the teammate with the highest success confidence, breaking ties randomly. If the DT does not predict that any pass would succeed, the RCF specifies that the agent with the ball should dribble or kick the ball forwards (towards the opponent goal or towards one of the forward corners).

r	D_r
G	(RD,RM,LD,LM,CM)
S	(RD,RM,LD,LM,CM)
LD	(LM,LF,CM)
CD	(RM,LM,CM)
RD	(RM,RF,CM)
LM	(LF,CF,RF)
CM	(RM,RF,CF,LM,LF)
RM	(RF,CF,LF)
LF	(CF)
CF	—
RF	(CF)

Table 6.5: The ordered preference lists of the positions in the 4-3-3 formation when using the PRW RCF. D_r is the preference list for position r. The positions are indicated as labeled in Figure 6.8. The CF position has no passing options: it always dribbles or shoots.

6.3.2 Reasoning about Action Execution Time

In the experiments reported in this chapter, the agents are not equipped with the turnball behavior described in Appendix B (Section B.1.1). Thus, when a player is between the ball and the teammate to which it wants to pass, it must move out of the ball's path before passing, which empirically takes between 5 and 15 simulator cycles. If there is an opponent in the vicinity, it may be able to steal the ball in that time.

Therefore, the RCF definition (Table 6.4) includes reasoning about the available time to execute an action. In particular, if there is an opponent within d_{min}, there is a danger of losing the ball before being able to pass or shoot it. In this situation, it is to the passer's advantage to get rid of the ball as quickly as possible.

This priority is manifested in the RCFs in two ways:

- In Step 4 of Table 6.4, when there is an opponent within d_{min}, the RCFs only consider passing to players to whom the agent can pass immediately. As mentioned above, this concept is not purely reactive: the positions of opponents that are outside an agent's field of view are remembered.

- In Step 5, when an opponent is within d_{min}, the agent kicks the ball away (or shoots) rather than dribbling. The point of dribbling the ball (kicking the ball a small amount in a certain direction and staying with it) is to keep the ball for a little longer until a good pass becomes available or until the player is in a good position to shoot. However, if there is an opponent nearby, dribbling often allows the opponent time to get to the ball. In this situation, as indicated in Step 5 of Table 6.4, the RCF causes the player to kick the ball forward (or shoot) rather than dribbling.

Thus, the RCF considers whether there is enough time to execute an action without an opponent stealing the ball. The ability to reason about how much time is available for action is an important component of the RCFs and contributes significantly to their success in game situations (see Section 6.3.4).

6.3.3 Incorporating the RCF in a Behavior

In Section 6.3.1, the method of using a DT as a part of an RCF is described in detail. However, the RCF is itself not a complete agent behavior: it only applies when the ball is within `kickable_area` of a the player. This section situates the RCF within a complete behavior that can then be used throughout the course of a game. Again, this behavior is not the same as the CMUnited-98 implementation presented in Section 3.5: it is a preliminary version of the full-fledged CMUnited-98 implementation. The complete behavior is laid out in Table 6.6.

Let α be the agent using this behavior, and ball-dist = dist(α,`ball`).

1. **IF** confidence in the ball's location (see Section 3.5.1) < .5 **THEN** face the ball.
2. **IF** ball-dist $\leq d_{chase}$ or $\forall a \in$ teammates, ball-dist $\leq$ dist(a,`ball`) **THEN**:
   - **IF** ball-dist > `kickable_area` **THEN** move to the ball, using the trained NN when appropriate;
   - **ELSE** pass, dribble, or kick the ball as indicated by the RCF.
3. **ELSE**: (ball-dist > d_{chase} and $\exists a \in$ teammates, ball-dist > dist(a,`ball`))
   - Move within α's home position using ball-dependent flexible positioning (see Section 3.5.2).

Table 6.6: The complete behavior used by the players in game situations to test the different RCFs.

When using this behavior, the player's first priority (Step 1) is always to find the ball's location (only objects in front of the player are seen). If it does not know where the ball is, it turns until the ball is in view. When turning away from the ball, it remembers the ball's location for a short amount of time; however after about three seconds without seeing the ball, its confidence in the ball's location decays enough that it assumes that it no longer knows where the ball is (see Section 3.5.1).

Once the ball has been located, the agent can carry on with its behavior. As indicated in Step 2 of Table 6.6, the agent chases the ball when either of two conditions is met:

- When the ball is within d_{chase}, the agent always goes towards the ball. We set $d_{chase} = 10$.
- The agent chases the ball whenever it thinks that it is the closest team-member to the ball.

In the second case, notice that the agent may not *actually* be the closest player to the ball if some of its teammates are too far away to see and if they have not announced their positions recently. However, if a player mistakenly thinks that it is the closest player, it will get part of the way to the ball, notice that another teammate is closer, and then turn back to its position.

As required for use of the DT, every player is equipped with the trained neural network (see Chapter 5) which can be used to help intercept the ball. Whenever the ball is within `kickable_area`, the agent uses its RCF to decide whether to dribble, kick, or pass, and to where. Every team member uses the same RCF.

Finally, as described in Section 6.3.1, each player is assigned a particular position on the field. Unless chasing the ball, the agent goes to its position, moving around within the position's home range using ball-dependent flexible positioning (Step 3).

6.3.4 Results

In this section I present the results of empirically testing how the behaviors specified in Section 6.3.3 perform. Since the behaviors differ only in their RCFs, I refer below to, for example, "the complete behavior using the DT RCF" simply as "the DT RCF." Also presented are empirical results verifying the advantage of reasoning about action-execution time.

In order to test the different RCFs, we created a team formation that emphasizes the advantage of passing to some teammates over others. When both teams use the standard 4-3-3 formation (that of the black team in Figure 6.8), every player is covered by one opponent. However, this situation is an artificial artifact of using the ball-dependent player-positioning algorithm. In reality, the players—using SPAR (Section 3.5.2)—have the ability to move to open positions on the field. For the purposes of these experiments we use a simpler behavior than the full CMUnited-98 implementation: the players only use ball-dependent positioning. In order to reflect the fact that some players are typically more open than others, we test the RCFs against the OPR—*only play right*—formation which is illustrated by the white team in Figure 6.8. We also use the symmetrical OPL—*only play left*—formation for testing. These behaviors are specified in Table 6.7.

During testing, each run consists of 34 five-minute games between a pair of teams. We tabulate the cumulative score both in total goals and in games won (ties are not broken) as shown in Table 6.8. Graphs record the *differences* in cumulative goals scored (Figure 6.9) and games won (Figure 6.10) as the runs progress.

In order to test the effectiveness of the DT RCF, we compared its performance against the performance of the PRW and RAND RCFs when facing the same opponent: OPR. While the DT and RAND RCFs are symmetrical in their decision making, the PRW RCF gives preference to one side of the field and therefore has an advantage against the OPR strategy.

- The opponent behaviors are exactly the same as the RAND behavior except that the players are assigned to different positions:

  OPR (*only play right*): As illustrated by the white team in Figure 6.8, two players are at each position on the right side of the field, with no players on the left side of the field.

  OPL (*only play left*): Same as above, except all the players are on the left side of the field.

Table 6.7: OPR and OPL behavior specifications.

RCF (vs. OPR)	Games (W – L)	Overall Score
DT	19 – 9	135 – 97
PRW	11 – 14	104 – 105
PRW (vs. OPL)	8 – 16	114 – 128
RAND	14 – 12	115 – 111

Table 6.8: The results of using different RCFs. Results are cumulative over 34 five-minute games: ties are not broken. Unless otherwise indicated, the opponent—whose score always appears second—uses the OPR formation.

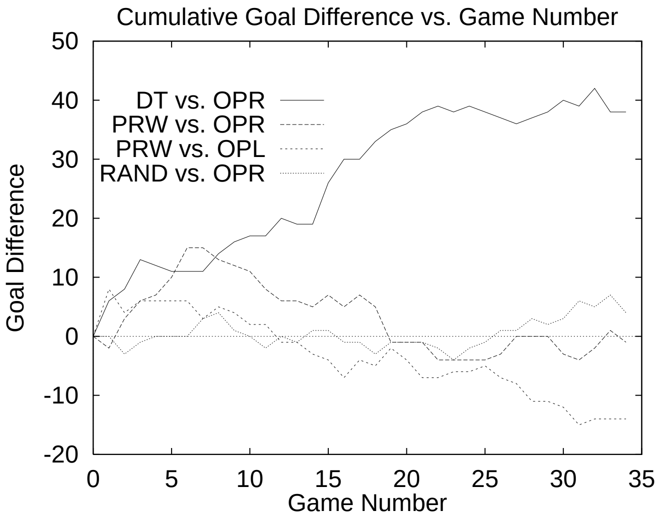

Figure 6.9: The differences in cumulative goals as the runs progress.

Thus we also include the results of the PRW RCF when it faces the symmetrical opponent: OPL. From Table 6.8 and Figures 6.9 and 6.10 it is apparent that the DT RCF is an effective method of decision making in this domain.

In order to test the effectiveness of the reasoning about action-execution time, we compare the performance of the standard DT RCF against that of the same RCF with the assumption that there is *never* an opponent within d_{min} ("No-rush DT"): even if there is, the RCF ignores it. This assumption affects Steps 4 and 5 of the RCF specification in Table 6.4 as described in Section 6.3.2. Both RCFs are played against the OPR behavior. As apparent from Table 6.9,

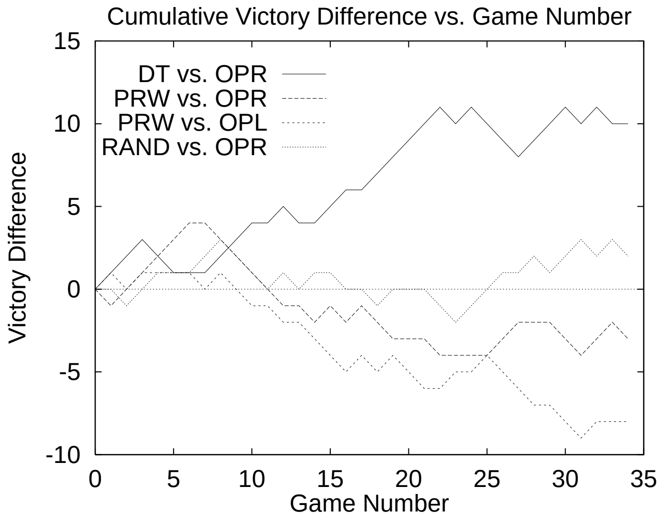

Figure 6.10: The differences in cumulative games won as the runs progress.

the reasoning about action-execution time makes a significant difference.

RCF (vs. OPR)	Games (W – L)	Overall Score
Standard DT	19 – 9	135 – 97
No-rush DT	13 – 16	91 – 108

Table 6.9: The effect of reasoning about action-execution time within an RCF. No-rush DT is the same RCF as the standard DT RCF except that there is no reasoning about action-execution time. The Standard DT RCF performs significantly better.

6.3.5 Summary

This section demonstrated that, even though trained in a limited, artificial scenario, the learned pass-evaluation capability trained in Section 6.1.1 generalizes successfully to full games. The players can successfully use it to choose which pass to make from among several options. Combined with some basic reasoning about the action-execution times of different options—necessitated by the real-time nature of this domain—the DT-based control function outperforms both random and hand-coded alternatives. The success of this multi-agent behavior encouraged us to incorporate it into a *learned* full team behavior as presented in Chapter 7.

6.4 Discussion

In this section I discuss two aspects of the learned pass-evaluation capability presented in this chapter. First, I discuss how it fits within the layered learning context. Then I discuss the use of the decision tree confidence factors for agent control.

6.4.1 Pass Evaluation within Layered Learning

The pass-evaluation capability described in this chapter is the second layer of our robotic soccer layered learning implementation. Its choice and implementation follow the principles of layered learning as laid out in Section 4.1:

- It is easier to gather data and to exploit it for training a pass-evaluation function than it is to code such a function by hand. Especially since the question of whether or not a pass will succeed depends on the abilities of players to receive the ball, there is no obvious heuristic that could be used.

- We choose the learning method—the C4.5 decision tree training algorithm—based on the task characteristics. We identify 174 features with potential predictive power for pass evaluation, many of which are frequently unknown. Therefore, the abilities of C4.5 to identify the most relevant features and to handle missing feature values are very important. In addition, the fact that C4.5 provides confidence factors along with its predictions turns out to be very useful.

- The previous learned layer—ball-interception—is used as part of the training behavior: both the intended receiver and all of the opponents use the trained neural network when trying to intercept the moving ball.

Just as this multi-agent behavior builds upon the interception skill, higher-level learned behaviors can be built upon the knowledge of when a pass will succeed. Such knowledge can contribute to the decision of to which player to pass or of whether to pass, dribble, or shoot.

The behavior defined in Section 6.3 uses the DT as a part of a hand-coded high-level multi-agent behavior. However, the decision function is highly constrained by the limited number of passing options allowed to each player. A behavior that *learns* completely how to map the classifications and confidence factors of the DT to passing/dribbling/shooting decisions may perform better.

Indeed, in Chapter 7, such a learned behavior is created. It uses the learned pass-evaluation function as the input representation to a reinforcement learning algorithm. The result is a completely learned RCF with no limitations on the passing options: players can pass backwards as well as forwards.

6.4.2 Confidence Factors for Agent Control

Although Decision Trees are widely used for classification tasks, they are typically not used for agent control. Nevertheless, the DT RCF uses the confidence factors associated with classifications to differentiate between pass options at a fine-grained level. Rather than just classifying each option as a likely success or likely failure, the RCFs choose the option with the highest confidence of being a success.

The experiments reported Section 6.3.4 indicate that the confidence factors provided by standard DT software can be used for effective agent control. To my knowledge, this is the first successful use of DT classification confidence factors for agent control. As DT confidence factors are effective tools in this domain, they are potentially useful for agent control in general.

6.5 Related Work

The DT RCF is not the only instance of using tree-like structures for agent control. Specifically, in a multi-agent planning context, operator success probabilities have been stored in COBWEB trees [Garland and Alterman 96]. This multi-agent, case-based planning system derives operator success probabilities from the results of past operator executions and uses the success probabilities to guide an agent's future planning choices. If an operator fails to achieve its intended effect due to the non-cooperation of other agents, an agent can adapt by instead using an operator (or operators) that allows it to achieve its goal on its own.

While the COBWEB tree is built with the explicit aim of estimating the success probability of an operator, the DT trained in this chapter was originally intended simply to classify the results of passes into three discrete classes. This classification task is the type of task that DT training algorithms such as C4.5 are typically used for. The confidence factors embedded in the C4.5 algorithm have not been used for control tasks before.