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 9

Related Work

This dissertation contributes to the fields of Multi-Agent Systems (MAS), Machine Learning (ML), and a subfield of ML, Reinforcement Learning (RL). In each of these areas, there is an immense body of literature. The thesis also contributes to the growing body of research within the robotic soccer domain.

In this chapter, I review the prior work in these areas that is most related to my thesis research. In Section 9.1, I review the intersection of MAS and ML. In particular, I give an overview of MAS with emphasis on multi-agent learning approaches [Stone and Veloso 97]. In Section 9.2, I review research within the robotic soccer domain.

Note that Chapters 3–7 each contain discussions of the related work that most closely pertains to their individual topics. This chapter does not repeat those discussions.

9.1 MAS from an ML Perspective

There are many possible ways to divide MAS and the related field of Distributed Artificial Intelligence (DAI). Overviews and taxonomies are numerous [Decker 87, Bond and Gasser 88, Durfee *et al.* 89, Durfee 92, Lesser 95, Parunak 96, Stone and Veloso 97, Jennings *et al.* 98, Sycara 98], each with its own way to organize the field. This chapter is organized along two main dimensions: agent heterogeneity and amount of communication among agents. Agents are *homogeneous* if they are physically and behaviorally identical; they are *heterogeneous* if they differ in some way. *Communication* is direct interaction among agents in the world. Beginning with the simplest multi-agent scenario, homogeneous non-communicating agents, the full range of possible multi-agent systems, through highly heterogeneous communicating agents, is considered.

Because of the inherent complexity of MAS, there is much interest in using ML techniques to help deal with this complexity [Weiß and Sen 96, Sen 96, Weiß 97]. Many existing ML techniques can be directly applied in multi-agent scenarios by delimiting a part of the domain that only involves a single agent. However *multi-agent learning* is more concerned with learning issues that arise because of the multi-agent aspect of a given domain. As described by Weiß, multi-agent learning is "learning that is done by several agents and that becomes possible only because several agents are present" [Weiß 95].

In this section, I consider MAS from an ML perspective. Specifically, I consider the

primary research topics in MAS, giving examples of ML approaches to these topics when possible. Section 9.1.1 considers multi-agent systems with homogeneous non-communicating agents; Section 9.1.2 considers multi-agent systems with heterogeneous non-communicating agents; Section 9.1.3 considers multi-agent systems with homogeneous communicating agents; and Section 9.1.4 considers multi-agent systems with heterogeneous communicating agents. Many of the issues that arise in the earlier scenarios also apply in the later scenarios. Nevertheless, they are only mentioned again in the later scenarios to the degree that they differ or become more complex. The multi-agent scenarios along with the issues that arise therein are summarized in Table 9.1.

Homogeneous non-communicating	**Heterogeneous non-communicating**
• Reactive vs. deliberative agents	• Benevolence vs. competitiveness
• Local or global perspective	• Stable vs. evolving agents
• Modeling other agents' states	• Modeling others' goals, actions, knowledge
• How to affect others	• Resource management (interdependent actions)
	• Social conventions
	• Roles
Homogeneous communicating	**Heterogeneous communicating**
• Distributed sensing	• Understanding each other
	• Planning communicative acts
	• Benevolence vs. competitiveness
	• Negotiation
	• Resource management (schedule coordination)
	• Commitment/decommitment

Table 9.1: Issues arising in the various MAS scenarios as reflected in the literature.

In multi-agent systems, there are multiple agents which model each other's goals and/or actions. In the fully general multi-agent scenario, there may be direct interaction among agents via communication. Although this interaction could be viewed as environmental stimuli, we present inter-agent communication as being separate from the environment. From an individual agent's perspective, the environment's dynamics can be affected by other agents. In addition to the uncertainty that may be inherent in the domain, other agents intentionally affect the environment in unpredictable ways. Thus, all multi-agent systems can be viewed as having dynamic environments.

Figure 9.1 illustrates the view that each agent is both part of the environment and modeled as a separate entity. There may be any number of agents, with different degrees of heterogeneity and with or without the ability to communicate directly. From the fully general case depicted here, I begin by eliminating both the communication and the heterogeneity to present homogeneous, non-communicating MAS (Section 9.1.1). Then, in Sections 9.1.2 and 9.1.3, the possibilities of agent heterogeneity and inter-agent communication are considered one at a time. Finally, in Section 9.1.4, we arrive back at the fully general case by considering heterogeneous agents that can interact directly.

For each of these scenarios, I present the research issues that arise and the existing techniques to deal with them. The issues may appear across scenarios, but they are presented and discussed in the first scenario to which they apply. When possible, ML approaches are emphasized.

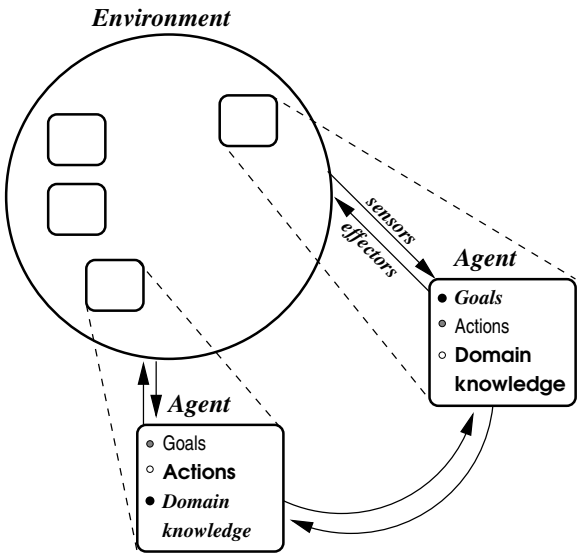

Figure 9.1: The general multi-agent scenario. Agents model each other's goals, actions, and domain knowledge, which may differ as indicated by the different fonts. They may also interact directly (communicate) as indicated by the arrows between the agents.

The simulated robotic soccer domain can be used to study all of the different multi-agent scenarios presented in this section. Throughout the section, I discuss how the different issues are reflected in this domain. Section 9.2 presents related robotic soccer research in detail.

9.1.1 Homogeneous, Non-Communicating MAS

In homogeneous, non-communicating multi-agent systems, all of the agents have the same internal structure including goals, domain knowledge, and *possible* actions. They also have the same procedure for selecting among their actions. The only differences among agents are their sensory inputs and the actual actions they take: they are situated differently in the world. Having different effector output is a necessary condition for MAS: if the agents all act as a unit, then they are essentially a single agent. In order to realize this difference in output, homogeneous agents must have different sensor input as well. Otherwise they will act identically. For this scenario, in which I consider non-communicating agents, assume that the agents cannot communicate directly. Figure 9.2 illustrates the homogeneous, non-communicating multi-agent scenario, indicating that the agents' goals, actions, and domain knowledge are the same by representing them with identical fonts.

Even in this most restrictive of multi-agent scenarios, there are several issues with which to deal. The techniques provided here are representative examples of ways to address the presented issues. The issues and techniques are summarized in Table 9.2.

Reactive vs. Deliberative agents. When designing any agent-based system, it is important to determine how sophisticated the agents' reasoning will be. Reactive agents simply retrieve pre-set behaviors similar to reflexes without maintaining any internal state. On the other hand, deliberative agents behave more like they are thinking, by searching through a space of behaviors, maintaining internal state, and predicting

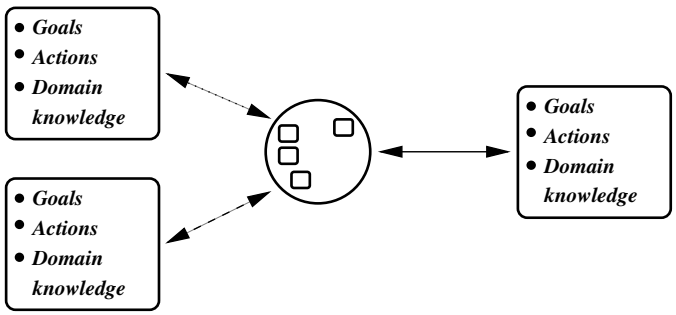

Figure 9.2: MAS with homogeneous, non-communicating agents. Only the sensor input and effector output of agents differ, as represented by the different arrow styles. The agents' goals, actions, and domain knowledge are all identical as indicated by the identical fonts.

Homogeneous, Non-Communicating MAS

Issues

- Reactive vs. deliberative agents
- Local or global perspective
- Modeling other agents' states
- How to affect others

Techniques

- Reactive behaviors for formation maintenance. [Balch and Arkin 95]
- Deliberative behaviors for pursuit. [Levy and Rosenschein 92]
- Mixed reactive and deliberative behaviors. [Sahota 94, Rao and Georgeff 95]
- Local knowledge sometimes better. [Roychowdhury *et al.* 96]
- (limited) Recursive Modeling Method (RMM). [Durfee 95]
- Don't model others–just pay attention to reward. [Schmidhuber 96]
- Stigmergy. [Goldman and Rosenschein 94, Holland 96]
- Q-learning for behaviors like foraging, homing, etc. [Mataric 94a]

Table 9.2: The issues and techniques for homogeneous, non-communicating multi-agent systems as reflected in the literature.

the effects of actions. Although the line between reactive and deliberative agents can be somewhat blurry, an agent with no internal state is certainly reactive, and one which bases its actions on the predicted actions of other agents is deliberative. Here we describe one system at each extreme as well as two others that mix reactive and deliberative reasoning.

Balch and Arkin [95] use homogeneous, reactive, non-communicating agents to study formation maintenance in autonomous robots. The robots' goal is to move together in a military formation such as a diamond, column, or wedge. They periodically come across obstacles which prevent one or more of the robots from moving in a straight line. After passing the obstacle, all robots must adjust in order to regain their formation. The agents reactively convert their sensory data (which includes the positions of the other robots) to motion vectors for avoiding obstacles, avoiding robots, moving to a goal location, and formation maintenance. The actual robot motion is a simple weighted sum of these vectors. Military formations differ from the formations introduced as part

of our flexible teamwork structure in Section 3.3. In our case, a "formation" refers to a decomposition of the team's overall task as opposed to a geometrical configuration of agents.

At the deliberative end of the spectrum is some research in the pursuit domain [Benda *et al.* 86], a multi-agent domain in which several independent predators try to catch a prey. Levy and Rosenschein [92] create agents that each act in service of its own goals. They use game theoretic techniques to find equilibrium points and thus to decide how to act. These agents are clearly deliberative, as they search for actions rather than simply retrieving them.

There are also several existing systems and techniques that mix reactive and deliberative behaviors. One example is the OASIS system which reasons about when to be reactive and when to follow goal-directed plans [Rao and Georgeff 95]. Another example is *reactive deliberation* [Sahota 94]. As the name implies, it mixes reactive and deliberative behavior: an agent reasons about which reactive behavior to follow under the constraint that it must choose actions at a rate of 60 Hz. Reactive deliberation was developed on the first robotic soccer platform [Barman *et al.* 93].

The agents developed in this thesis also mix reactive and deliberative behaviors. Internal behaviors in the team member agent architecture are exclusively deliberative as they rely upon the locker-room agreement and past internal states. In our soccer implementation, the top-level external behavior which chooses the agent's behavior mode is also deliberative, whereas some of the low-level external behaviors, such as turning to face the ball when it is visible, are purely reactive. When turning to face the ball, an agent refers only to the ball's current relative angle and turns in that direction.

Local or global perspective. Another issue to consider when building a multi-agent system is how much sensor information should be available to the agents. Even if it is feasible within the domain to give the agents a global perspectives of the world, it may be more effective to limit them to local views.

As presented in Chapter 2, agents in the simulated robotic soccer and network routing domains are restricted to local perspectives by the nature of their sensory inputs: they can only see a portion of the world. In the real robotic soccer domain, our agents have a global world view.

Roychowdhury et al. consider a case of multiple agents sharing a set of identical resources in which they have to learn (adapt) their resource usage policies [Roychowdhury *et al.* 96]. Since the agents are identical and do not communicate, if they all have a global view of the current resource usage, they will all move simultaneously to the most under-used resource. However, if they each see a partial picture of the world, then different agents gravitate towards different resources: a preferable effect. Better performance by agents with less knowledge is occasionally summarized by the cliche "Ignorance is Bliss."

Modeling other agents' states. Durfee [95] gives another example of "Blissful Ignorance," mentioning it explicitly in the title of his paper: "Blissful Ignorance: Knowing Just

Enough to Coordinate Well." Now rather than referring to resource usage, the saying applies to the limited recursive modeling method (RMM). When using RMM, agents explicitly model the belief states of other agents, including what they know about each others' beliefs. If agents have too much knowledge, RMM could recurse indefinitely. Even if further information can be obtained by reasoning about what agent A thinks agent B thinks agent A thinks ..., endless reasoning can lead to inaction. Durfee contends that for coordination to be possible, some potential knowledge must be ignored.

Although it may be useful to build models of other agents in the environment, agent modeling is not done universally. A form of multi-agent RL is defined in which agents do not model each other as agents [Schmidhuber 96]. Instead they consider each other as parts of the environment and affect each other's policies only as sensed objects. The agents pay attention to the reward they receive using a given policy and checkpoint their policies so they can return to successful ones. Schmidhuber shows that the agents can learn to cooperate without modeling each other.

Similarly, the agents developed in this thesis learn to act in an environment involving adversaries without explicitly modeling the adversaries' intentions. They do, on the other hand, model the goals and internal states of teammates.

How to affect others. When no communication is possible, agents cannot interact with each other directly. However, since they exist in the same environment, the agents can affect each other indirectly in several ways. They can be sensed by other agents, or they may be able to change the state of another agent by, for example, pushing it. Agents can also affect each other by one of two types of *stigmergy* [Holland 96]. First, *active stigmergy* occurs when an agent alters the environment so as to affect the sensory input of another agent. For example, a robotic agent might leave a marker behind it for other agents to observe. Goldman and Rosenschein [94] demonstrate an effective form of active stigmergy in which agents heuristically alter the environment in order to facilitate future unknown plans of other agents. Second, *passive stigmergy* involves altering the environment so that the effects of another agent's actions change. For example, if one agent turns off the main water valve to a building, the effect of another agent turning on the kitchen faucet is altered.

Holland [96] illustrates the concept of passive stigmergy with a robotic system designed to model the behavior of an ant colony confronted with many dead ants around its nest. An ant from such a colony tends to periodically pick up a dead ant, carry it for a short distance, and then drop it. Although the behavior appears to be random, after several hours, the dead ants are clustered in a small number of heaps. Over time, there are fewer and fewer large piles until all the dead ants end up in one pile. Although the ants behave homogeneously and, at least in this case, we have no evidence that they communicate explicitly, the ants manage to cooperate in achieving a task.

Holland [96] models this situation with a number of identical robots in a small area in which many pucks are scattered around. The robots are programmed reactively to move straight (turning at walls) until they are pushing three or more pucks. At that

point, the robots back up and turn away, leaving the three pucks in a cluster. Although the robots do not communicate at all, they are able to collect the pucks into a single pile over time. This effect occurs because when a robot approaches an existing pile directly, it adds the pucks it was already carrying to the pile and turns away. A robot approaching an existing pile obliquely might take a puck away from the pile, but over time the desired result is accomplished. Like the ants, the robots use passive stigmergy to affect each other's behavior.

A similar scenario with more deliberative robots is explored by Mataric [94a]. In this case, the robots use Q-learning to learn behaviors including foraging for pucks as well as homing and following. Mataric's robots actively affect each other through observation: a robot learning to follow another robot can base its action on the relative location of the other robot.

Our small-robots use active stigmergy. When a robot that does not have the ball is filling a forward role and it moves to a new location on the field, it changes the result of the pass evaluation function used by the robot that does have the ball. As presented in Section B.2.5 of Appendix B, the robot with the ball bases its action on the perceived locations of teammates and opponents.

Similarly, our simulated robotic soccer agents use active stigmergy when they change their locations in the environment. Particularly when using the SPAR variant of flexible positioning (see Section 3.5.2), teammate positions affect an agent's behavior.

9.1.2 Heterogeneous, Non-Communicating MAS

The multi-agent scenario with heterogeneous, non-communicating agents is depicted in Figure 9.3. As in the homogeneous case (Figure 9.2), the agents are situated differently in the environment which causes them to have different sensory inputs and necessitates their taking different actions. However in this scenario, the agents have much more significant differences. They may have different goals, actions, and/or domain knowledge, as indicated by the different fonts in Figure 9.3.

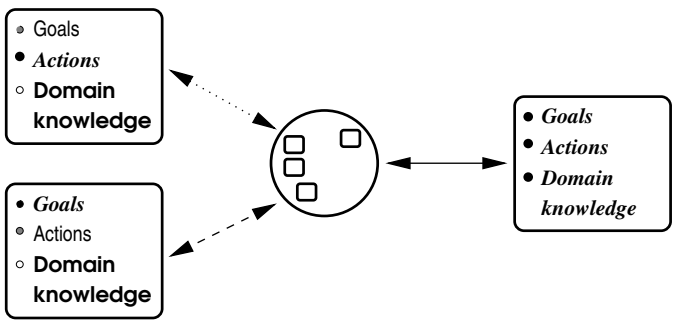

Figure 9.3: MAS with heterogeneous, non-communicating agents. Now agents' goals, actions, and/or domain knowledge may differ as indicated by the different fonts. The assumption of no direct interaction remains.

To this point, we have only considered agents that are homogeneous. Adding the possibility of heterogeneous agents in a multi-agent domain adds a great deal of potential power

at the price of added complexity. Agents might be heterogeneous in any of a number of ways, from having different goals to having different domain models and actions. An important sub-dimension of heterogeneous agent systems is whether agents are benevolent or competitive. Even if they have different goals, they may be friendly to each others' goals or they may actively try to inhibit each other. This aspect of heterogeneous systems, along with several others summarized in Table 9.3, is described below.

Heterogeneous, Non-Communicating MAS

Issues

- Benevolence vs. competitiveness
- Stable vs. evolving agents
- Modeling others' goals, actions, knowledge
- Resource management (interdependent actions)
- Social conventions
- Roles

Techniques

- Game theory, iterative play. [Mor and Rosenschein 95, Sandholm and Crites 96]
- Minimax-Q. [Littman 94]
- Competitive co-evolution. [Rosin and Belew 95, Haynes and Sen 96, Grefenstette and Daley 96]
- Deduce intentions, abilities through observation. [Huber and Durfee 95, Wang 96]
- Autoepistemic reasoning (ignorance). [Permpoontanalarp 95]
- Model as a team (individual $\rightarrow$ role). [Tambe 95, Tambe 96b]
- Social reasoning: depend on others for goal ($\neq$ game theory). [Sichman and Demazeau 95]
- GAs to deal with Braes' paradox (more resource $\rightarrow$ worse). [Glance and Hogg 95, Arora and Sen 96]
- Multi-Agent RL for adaptive load balancing. [Schaerf *et al.* 95]
- Focal points/emergent conventions. [Fenster *et al.* 95, Walker and Wooldridge 95]
- Agents filling different roles. [Prasad *et al.* 96, Tambe 97, Balch 98]

Table 9.3: The issues and techniques for heterogeneous, non-communicating multi-agent systems as reflected in the literature.

Benevolence vs. competitiveness. One of the most important issues to consider when designing a multi-agent system is whether the different agents will be benevolent or competitive. Even if they have different goals, the agents can be benevolent if they are willing to help each other achieve their respective goals [Goldman and Rosenschein 94]. On the other hand, the agents may be selfish and only consider their own goals when acting. In the extreme, the agents may be involved in a zero-sum situation so that they must actively oppose other agents' goals in order to achieve their own.

Mor and Rosenschein [95] illustrate the benevolent scenario in the context of the prisoner's dilemma. In the prisoner's dilemma, two agents try to act so as to maximize their own individual rewards. They are not actively out to thwart each other since it is not a zero-sum game, yet they place no inherent value on the other receiving reward. The prisoner's dilemma is constructed so that each agent is given two choices: defect or cooperate. No matter what the other agent does, a given agent receives a

higher reward if it defects. Yet if both agents cooperate, they are better off than if they both defect. In any given play, an agent is better off defecting. Nevertheless, Mor and Rosenschein show that if the same agents come up against each other repeatedly (the iterated prisoner's dilemma), cooperative behavior can emerge. In effect, an agent can serve its own self-interest by establishing a *reputation* for being cooperative. Then when coming up against another cooperative agent, the two can benefit from a sense of trust for each other: they both cooperate rather than both defecting. Only with repeated play can cooperation emerge among the selfish agents in the prisoner's dilemma.

In the prisoner's dilemma, the agents are selfish but not inherently competitive: in specific circumstances, they are willing to act benevolently. However, when the agents are actually competitive (such as in zero-sum games), cooperation is no longer sensible. For instance, Littman considers a zero-sum game in which two agents try to reach opposite ends of a small discrete world. The agents can block each other by trying to move to the same space. Minimax-Q is a variant of Q-learning which is designed to work on Markov games as opposed to Markov decision processes [Littman 94]. The competitive agents learn probabilistic policies since any deterministic policy can be completely counteracted by the opponent. Minimax-Q is applied to a simplified soccer simulation. Like in this application, the robotic soccer domains considered in this thesis are zero-sum games at the team level: the two teams are in direct competition with one another.

Stable vs. evolving agents. Another important characteristic to consider when designing multi-agent systems is whether the agents are stable or evolving. Evolving agents can be useful in dynamic environments. But particularly when using competitive agents, allowing them to evolve can lead to complications. Such systems that use competitive evolving agents are said to use a technique called *competitive co-evolution*. Systems that evolve benevolent agents are said to use *cooperative co-evolution*.

The evolution of both predator and prey agents in the pursuit domain qualifies as competitive co-evolution [Haynes and Sen 96]. When just the predators are evolved together, it is cooperative co-evolution [Haynes *et al.* 95]. Rather than evolving predator agents in a single evolutionary pool and then combining them into teams to test performance, this approach considers each individual in the population as a team of four agents already specifically assigned to different predators. Thus the predators can evolve to cooperate. This co-evolution of teammates is one possible way around the absence of communication in a domain. In place of communicating planned actions to each other, the predators can evolve to *know*, or at least act as if knowing, each other's future actions.

Grefenstette and Daley [96] conduct a preliminary study of competitive and cooperative co-evolution in a domain that is loosely related to the pursuit domain. Their domain has two robots that can move continuously and one morsel of (stationary) food that appears randomly in the world. In the cooperative task, both robots must be at the food in order to "capture" it.

One problem to contend with in competitive rather than cooperative co-evolution is the possibility of an escalating "arms race" with no end. Competing agents might continually adapt to each other in more and more specialized ways, never stabilizing at a good behavior. In a dynamic environment, it may not be feasible or even desirable to evolve a stable behavior. Applying RL to the iterated prisoner's dilemma, it has been demonstrated that a learning agent is able to perform optimally against a fixed opponent [Sandholm and Crites 96]. But when both agents are learning, there is no stable solution.

Another issue in competitive co-evolution is the credit-assignment problem. When performance of an agent improves, it is not necessarily clear whether the improvement is due to an improvement in that agent's behavior or a negative change in the opponent's behavior. Similarly, if an agent's performance gets worse, the blame or credit could belong to that agent or to the opponent.

One way to deal with the credit-assignment problem is to fix one agent while evolving the other and then switch. This method encourages the arms race more than ever. Nevertheless, Rosin and Belew [95] use this technique, along with an interesting method for maintaining diversity in genetic populations, to evolve agents that can play Tic-TacToe, Nim, and a simple version of Go. When it is a given agent's turn to evolve, it executes a standard Genetic algorithm (GA) generation. Individuals are tested against individuals from the competing population, but a technique called "competitive fitness sharing" is used to maintain diversity. When using this technique, individuals from agent X's population are given more credit for beating opponents (individuals from agent Y's population) that are not beaten by other individuals from agent X's population. More specifically, the reward to an individual for beating individual y is divided by the number of other individuals in agent X's population that also beat individual y. Competitive fitness sharing shows much promise for people building systems that use competitive co-evolution.

TPOT-RL (Chapter 7) implements cooperative co-evolution. The individual agents learn their policies simultaneously, eventually creating a compatible set of policies. Competitive co-evolution is not considered in this thesis.

Modeling others' goals, action, knowledge. In the case of homogeneous agents, it was useful for agents to model the internal states of other agents in order to predict their actions. With heterogeneous agents, the problem of modeling others is much more complex. Now the goals, actions, and domain knowledge of the other agents may also be unknown and thus need modeling.

Without communication, agents are forced to model each other strictly through observation. Huber and Durfee [95] consider a case of coordinated motion control among multiple mobile robots under the assumption that communication is prohibitively expensive. Thus the agents try to deduce each other's plans by observing their actions. In particular, each robot tries to figure out the destinations of the other robots by watching how they move. Plan recognition of this type is also useful in competitive domains, since knowing an opponent's goals or intentions can make it significantly

easier to defeat.

In addition to modeling agents' goals through observation, it is also possible to learn their actions. The OBSERVER system [Wang 96] allows an agent to incrementally learn the preconditions and effects of planning actions by observing domain experts. After observing for a time, the agent can then experimentally refine its model by practicing the actions itself.

When modeling other agents, it may be useful to reason not only about what is true and what is false, but also about what is not known. Such reasoning about ignorances is called *autoepistemic reasoning* [Permpoontanalarp 95].

Just as RMM is useful for modeling the states of homogeneous agents, it can be used in the heterogeneous scenario as well. Tambe [95] takes it one step further, studying how agents can *learn* models of teams of agents. In an air combat domain, agents can use RMM to try to deduce an opponents' plan based on its observable actions. For example, a fired missile may not be visible, but the observation of a preparatory maneuver commonly used before firing could indicate that a missile has been launched.

When teams of agents are involved, the situation becomes more complicated. In this case, an opponent's actions may not make sense except in the context of a team maneuver. Then the agent's *role* within the team must be modeled [Tambe 96b].

One reason that modeling other agents might be useful is that agents sometimes depend on each other for achieving their goals. Unlike in game theory where agents can cooperate or not depending on their utility estimation, there may be actions that *require* cooperation for successful execution. For example, two robots may be needed to successfully push a box, or, as in the pursuit domain, several agents may be needed to capture an opponent. Sichman and Demazeau [95] analyze how the case of conflicting mutual models of different co-dependent agents can arise and be dealt with.

Resource management. Heterogeneous agents may have interdependent actions due to limited resources needed by several of the agents. Example domains include network traffic problems, including the network routing domain used in this thesis, in which several different agents must send information through the same network; and load balancing in which several computer processes or users have a limited amount of computing power to share among them.

One interesting network traffic problem called Braess' paradox has been studied from a multi-agent perspective using GAs [Glance and Hogg 95]. Braess' paradox is the phenomenon of adding more resources to a network but getting worse performance. When using a particular GA representation to represent different parts of a sample network that has usage-dependent resource costs, agents that are sharing the network and reasoning separately about which path of the network to use cannot achieve global optimal performance [Glance and Hogg 95]. When the GA representation is improved, the system is able to find the globally optimal traffic flow [Arora and Sen 96].

Adaptive load balancing has been studied as a multi-agent problem by allowing different agents to decide which processor to use at a given time. Using RL, heterogeneous

agents can achieve reasonable load balance without any central control and without communication among agents [Schaerf *et al.* 95]. The agents keep track of how long a job takes when it is scheduled on a given resource, and they are given some incentive to explore untried processors or processors that did poorly in the past.

Social conventions. There has been some very interesting work done on how heterogeneous, non-communicating agents can reach "agreements," or make coinciding choices, if necessary. The locker-room agreement presented in Section 3.2 is an implementation of social conventions in a team of agents. As well as being used by the the communicating simulated robotic soccer agents, it is used by the non-communicating small-robot agents to achieve team coordination.

The focal point method is another example of social conventions [Fenster *et al.* 95]. The phenomenon of cultural (or programmed) preferences allows agents to "meet" without communicating. In this method, all else being equal, agents who need to meet should choose rare or extreme options.

Rather than coming from pre-analysis of the options as in the focal point method, conventions can emerge over time if agents are biased towards options that have been chosen, for example, most recently or most frequently in the past [Walker and Wooldridge 95].

Roles. When agents have similar goals, they can be organized into a team. Each agent then plays a separate *role* within the team. With such a benevolent team of agents, one must provide some method for assigning different agents to different roles. The flexible teamwork structure presented in Section 3.3 provides one such method.

Prasad et al. [96] study design agents that can either initiate or extend a design of a steam pump. In different situations, different agents are more effective at initiation and at extension. Thus a supervised learning technique is used to help agents learn what roles they should fill in different situations.

STEAM [Tambe 97] allows a team of agents to fill and switch roles dynamically. Particularly if a critical agent fails, another agent is able to replace it in its role so that the team can carry on with its mission.

If allowed to evolve independently, a group of agents might end up filling different roles in the domain or all end up with the same behavior. Balch [98] investigates methods of encouraging behavioral diversity in a team of agents.

9.1.3 Homogeneous, Communicating MAS

The multi-agent scenario with homogeneous, communicating agents is depicted in Figure 9.4. As in the homogeneous, non-communicating case (Figure 9.2), the agents are identical except that they are situated differently in the environment. However in this scenario, the agents can communicate directly as indicated by the arrows connecting the agents in Figure 9.4. From a practical point of view, the communication might be broadcast or posted on a "blackboard"

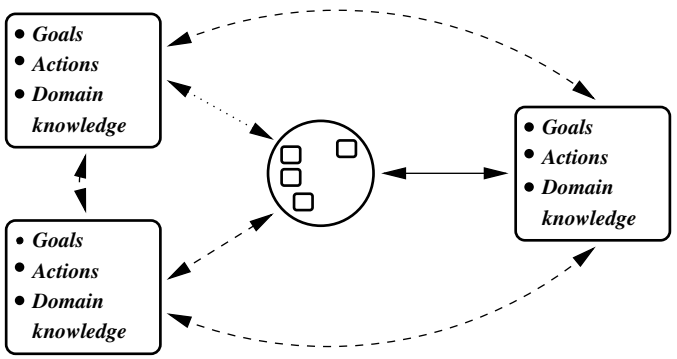

Figure 9.4: MAS with homogeneous, communicating agents. Only the sensor input and effector output of agents differ. Information can be transmitted directly among agents as indicated by the arrows between agents. Communication can either be broadcast or transmitted point-to-point.

for all to interpret, or it might be targeted point-to-point from an agent to another specific agent.

Communication raises several issues to be addressed in multi-agent systems. However, in most cases, the issues are addressed in the literature with heterogeneous, communicating agents. In this section, I consider distributed sensing, which is addressed with homogeneous, communicating agents as indicated in Table 9.4. Many more communication-related issues are addressed in Section 9.1.4.

Homogeneous, Communicating MAS
Issues
• Distributed sensing
Techniques
• Active sensing [Matsuyama 97]
• Query propagation for distributed traffic mapping [Moukas and Maes 97]

Table 9.4: The issues and techniques for homogeneous, communicating multi-agent systems as reflected in the literature.

Distributed Sensing The cooperative distributed vision project [Matsuyama 97] aims to construct and monitor a broad visual scene for dynamic three dimensional scene understanding by using multiple cameras, either stationary or on mobile robots. For example, consider the problem of tracking an individual car using cameras mounted at urban intersections. When the car leaves one camera's range and enters another's, there needs to be a way of identifying the two images as representing the same car, even though it probably looks different in the two cases (i.e. it is driving away from one camera and towards the other). The project combines active sensing—the ability to shift attention towards an area of higher uncertainty or interest—and communication among multiple sensing agents.

Another distributed sensing project is the trafficopter system [Moukas and Maes 97]. In trafficopter, cars themselves collect and propagate traffic information to help each other decide on the best route to a given location. For example, a car driving in one

direction might query an oncoming vehicle about traffic conditions up the road. By propagating such queries among vehicles, the original car can build a map of traffic conditions along different routes to its goal.

Our simulated robotic soccer agents use communication to achieve distributed sensing. In particular, the <selected-internal-state> in the communication paradigm presented in Section 3.4 can be used to transmit information that is visible to one agent and hidden to another. For example, in the soccer implementation, opponent and ball locations are communicated to agents that would otherwise not know their whereabouts.

9.1.4 Heterogeneous, Communicating MAS

The fully general multi-agent scenario with heterogeneous, communicating agents is illustrated in Figure 9.5.

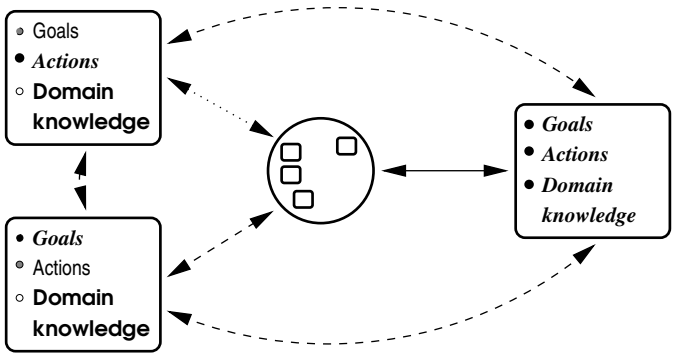

Figure 9.5: MAS with heterogeneous, communicating agents. Agents can be heterogeneous to any degree. Information can be transmitted directly among agents. Communication can either be broadcast or transmitted point-to-point.

Since heterogeneous communicating agents can choose not to communicate, and in some cases can also choose to be homogeneous or at least to minimize their heterogeneity, most of the issues discussed in the previous two scenarios apply in this one as well. Two of the most studied issues are communication protocols and theories of commitment. Already discussed in the context of the heterogeneous, non-communicating MAS scenario (Section 9.1.2), the issue of benevolence vs. competitiveness becomes more complicated in the current context. These issues and others along with some of the existing techniques to deal with them are described below and summarized in Table 9.5.

Understanding each other. In all communicating multi-agent systems, and particularly in domains that include agents built by different designers, there must be some set language and protocol for the agents to use when interacting. Independent aspects of protocols are information content, message format, and coordination conventions. Among many others, existing language protocols for these three levels are: KIF for content [Genesereth and Fikes 92], KQML for message format [Finin *et al.* 94], and COOL for coordination [Barbuceanu and Fox 95]. There has been a lot of research done on refining these and other communication protocols.

Heterogeneous, Communicating MAS

Issues

- Understanding each other
- Planning communicative acts
- Benevolence vs. competitiveness
- Negotiation
- Resource management (schedule coordination)
- Commitment/decommitment

Techniques

- Language protocols: KIF [Genesereth and Fikes 92], KQML [Finin *et al.* 94], COOL. [Barbuceanu and Fox 95]
- Legacy systems integration. [Jennings and Wittig 92]
- Language learning. [Grand and Cliff 98]
- Speech acts. [Cohen and Levesque 95, Lux and Steiner 95]
- Learning social behaviors. [Mataric 94b]
- Reasoning about truthfulness. [Rosenschein and Zlotkin 94, Sandholm and Lesser 96]
- Multi-agent Q-learning. [Tan 93, Weiß 95]
- Training other agents' Q-functions (track driving). [Clouse 96]
- Minimize the need for training. [Potter *et al.* 95]
- Cooperative co-evolution. [Bull *et al.* 95]
- Contract nets for electronic commerce. [Sandholm and Lesser 95b]
- Market-based systems. [Huberman and Clearwater 95]
- Bayesian learning in negotiation: model others. [Zeng and Sycara 96]
- Market-based methods for distributed constraints. [Parunak *et al.* 98]
- Generalized partial global planning (GPGP). [Decker and Lesser 95, Lesser 98]
- Learning to choose among coordination methods. [Sugawara and Lesser 95]
- Query response in information networks. [Sycara *et al.* 96]
- Division of independent tasks. [Parker 94]
- Internal, social, and collective (role) commitments. [Castelfranchi 95]
- Commitment states (potential, pre, and actual) as planning states. [Haddadi 95]
- Belief/desire/intention (BDI) model: OASIS. [Rao and Georgeff 95]
- BDI commitments only over intentions. [Rao and Georgeff 95]
- Coalitions. [Zlotkin and Rosenschein 94, Shehory and Kraus 95, Sandholm and Lesser 95a]

Table 9.5: The issues and techniques for heterogeneous, communicating multi-agent systems as reflected in the literature.

One of the first industrial multi-agent systems, ARCHON [Jennings and Wittig 92] successfully integrated several legacy systems. Applied in five different industrial settings, ARCHON successfully allows independently developed, heterogeneous computer systems to communicate in order to create collaborative, process control systems.

Creatures [Grand and Cliff 98] is a multi-agent computer game based on sophisticated biological models. Agents have the ability to grow and learn, including a simple verb-object language, based on interactions with a human user or other agents in the environment.

Within the team member agent architecture presented in Section 3.2, agents under-

stand each other based on a communication protocol defined within the locker-room agreement.

Planning communicative acts. When an agent transmits information to another agent, it has an effect just like any other action would have. Thus within a planning framework, one can define preconditions and effects for communicative acts. When combined with a model of other agents, the effect of a communication act might be to alter an agent's belief about the state of another agent or agents. The theory of communication as action is called *speech acts* [Cohen and Levesque 95, Lux and Steiner 95].

Mataric adds a learning dimension to the idea of speech acts. Starting with the foraging behavior mentioned above [Mataric 94a], the agents can then learn to choose from among a set of social behaviors that includes broadcasting and listening [Mataric 94b]. Q-learning is extended so that reinforcement can be received for direct rewards or for rewards to other agents.

When using communication as a planning action, the possibility arises of communicating misinformation in order to satisfy a particular goal. For instance, an agent may want another agent to believe that something is true. Rather than actually making it true, the agent might just *say* that it is true. For example, Sandholm and Lesser [96] analyze a framework in which agents are allowed to "decommit" from agreements with other agents by paying a penalty to these other agents. They consider the case in which an agent might not be truthful in its decommitment, hoping that the other agent will decommit first. In such situations, agents must also consider what communications to believe [Rosenschein and Zlotkin 94].

The communication paradigm for single-channel, low-bandwidth unreliable communication environments presented in Section 3.4 safeguards against false communications with the <encoded-time-stamp> field. In the simulated robotic soccer implementation (Section 3.5), it is assumed that messages from teammates are truthful: only opponent messages could be deceptive.

Benevolence vs. competitiveness. Several studies involving competitive agents were described in the heterogeneous, non-communicating scenario in Section 9.1.2. In the current scenario, there are many more examples of competitive agents.

In the pursuit domain, multi-agent Q-learning has been investigated [Tan 93]. Agents help each other by exchanging reinforcement episodes and/or control policies. Similarly, competing Q-learners can bid against each other to earn the right to control a single system [Weiß 95]. The highest bidder pays a certain amount to be allowed to act, then receives any reward that results from the action.

Another Q-learning approach, this time with benevolent agents, has been to explore the interesting idea of having one agent teach another agent through communication [Clouse 96]. Starting with a *trainer* that has moderate expertise in a task, a learner can be rewarded for mimicking the trainer. Furthermore, the trainer can recommend to the learner what action to take in a given situation so as to direct the learner towards a reward state. Eventually, the learner is able to perform the task without any guidance.

While training is a useful concept, some research is driven by the goal of reducing the role of the human trainer. As opposed to the process of *shaping*, in which the system designer develops simple behaviors and slowly builds them into more complex ones, populations appropriately seeded for competitive co-evolution can reduce the amount of designer effort. Potter and Grefenstette [95] illustrate this effect in their domain described above in which two robots compete for a stationary pellet of food. Subpopulations of rules used by GAs are seeded to be more effective in different situations. Thus specialized subpopulations of rules corresponding to shaped behaviors tend to emerge.

GAs have also been used to evolve separate communicating agents to control different legs of a quadrapedal robot using cooperative co-evolution [Bull *et al.* 95].

TPOT-RL as presented in Chapter 7 is another example of cooperative co-evolution. Independent agents learn policies that coordinate to create an effective team behavior.

Negotiation Drawing inspiration from competition in human societies, several researchers have designed negotiating multi-agent systems based on the law of supply and demand. In the *contract nets* framework [Smith 80], agents all have their own goals, are self-interested, and have limited reasoning resources. They bid to accept tasks from other agents and then can either perform the tasks (if they have the proper resources) or subcontract them to other agents. Agents must pay to contract their tasks out and thus shop around for the lowest bidder. Many multi-agent issues arise when using contract nets [Sandholm and Lesser 95b].

In a similar spirit is an implemented multi-agent system that controls air temperature in different rooms of a building [Huberman and Clearwater 95]. A person can set one's thermostat to any temperature. Then depending on the actual air temperature, the agent for that room tries to "buy" either hot or cold air from another room that has an excess. At the same time, the agent can sell the excess air at the current temperature to other rooms. Modeling the loss of heat in the transfer from one room to another, the agents try to buy and sell at the best possible prices. The market regulates itself to provide equitable usage of a shared resource.

Zeng and Sycara [96] study a competitive negotiation scenario in which agents use Bayesian Learning techniques to update models of each other based on bids and counter bids in a negotiation process.

The MarCon system [Parunak *et al.* 98] uses market-based methods for distributed constraint problems. Designers at different points along a supply chain negotiate the characteristics of the overall design by buying and selling characteristics and propagating the resulting constraints.

Resource management. MarCon is an example of multi-agent resource management: the design characteristics desired by one agent may consume the resources of another.

Similarly, generalized partial global planning (GPGP) allows several heterogeneous agents to post constraints, or commitments to do a task by some time, to each other's local schedulers and thus coordinate without the aid of any centralized agent [Decker

and Lesser 95]. A proposed general multi-agent architecture based on GPGP contains five components: "local agent scheduling, multi-agent coordination, organizational design, detection, and diagnosis [Lesser 98]."

In a heterogeneous, communicating multi-agent system applied to diagnosis of a local area network, agents learn to choose among different coordination strategies based on the current situation [Sugawara and Lesser 93, Sugawara and Lesser 95]. Less sophisticated coordination methods require fewer network and time resources, but may lead to tasks failing to be executed or to redundant actions by multiple agents.

RETSINA [Sycara *et al.* 96] uses three classes of heterogeneous, communicating agents to deliver information in response to specific user queries in information networks. RETSINA is able to satisfy the information requests of multiple users by searching multiple information sources, while considering network constraints and resource limitations of information agents. RETSINA has been used to implement several distributed network applications including a financial portfolio manager, a personal information manager and meeting scheduler, and a satellite visibility forecaster.

ALLIANCE and its learning variant L-ALLIANCE [Parker 94] use communication among heterogeneous robots to help divide independent tasks among the robots. With an emphasis on fault tolerance, agents only broadcast the task that they are currently working on. If the communication fails, multiple robots might temporarily try to do the same task, but they will eventually realize the conflict by observation and one will move on to a different task. In L-ALLIANCE, robots learn to evaluate each other's abilities with respect to specific tasks in order to more efficiently divide their tasks among the team.

Commitment/decommitment. When agents communicate, they may decide to cooperate on a given task or for a given amount of time. In so doing, they make *commitments* to each other. Committing to another agent involves agreeing to pursue a given goal, possibly in a given manner, regardless of how much it serves one's own interests. Commitments can make systems run much more smoothly by providing a way for agents to "trust" each other, yet it is not obvious how to get self-interested agents to commit to others in a reasonable way. The theory of commitment and decommitment (when the commitment terminates) has consequently drawn considerable attention.

Castelfranchi [95] defines three types of commitment: internal commitment—an agent binds itself to do something; social commitment—an agent commits to another agent; and collective commitment—an agent agrees to fill a certain role. Setting an alarm clock is an example of internal commitment to wake up at a certain time.

Commitment states have been used as planning states: potential cooperation, precommitment, and commitment [Haddadi 95]. Agents can then use means-ends analysis to plan for goals in terms of commitment opportunities. This work is conducted within a model called belief/desire/intention, or BDI.

BDI is a popular technique for modeling other agents. Other agents' domain knowledge (beliefs) and goals (desires) are modeled as well as their "intentions," or goals they are currently trying to achieve and the methods by which they are trying to achieve

them. The BDI model is used to build a system for air-traffic control, OASIS [Rao and Georgeff 95], which has been implemented for testing (in parallel with human operators who retain full control) at the airport in Sydney, Australia. Each aircraft is represented by a controlling agent which deals with a global sequencing agent. OASIS mixes reactive and deliberative actions in the agents: they can break out of planned sequences when coming across situations that demand immediate reaction. Since agents cannot control their beliefs or desires, they can only make commitments to each other regarding their intentions.

Finally, groups of agents may decide to commit to each other. Rather than the more usual two-agent or all-agent commitment scenarios, there are certain situations in which agents may want to form coalitions [Zlotkin and Rosenschein 94]. Since this work is conducted in a game theory framework, agents consider the utility of joining a coalition in which they are bound to try to advance the utility of other members in exchange for reciprocal consideration. Shehory and Kraus [95] present a distributed algorithm for task allocation when coalitions are either needed to perform tasks or more efficient that single agents. Sandholm and Lesser [95a] use a vehicle routing domain to illustrate a method by which agents can form valuable coalitions when it is intractable to discover the optimal coalitions.

Within the context of this thesis, when using set-plays as presented within the flexible teamwork structure (Section 3.3), agents are committed to each other for the duration of the execution of the multi-agent plan. If any agent fails to fill its set-play role, the plan is less likely to succeed.

9.1.5 Summary

This section has given an overview of the prior research within the field of MAS, with emphasis on multi-agent learning approaches. Based on this detailed review of the literature, this thesis makes several contributions to the state of the art.

Several people have studied collaboration among agents (commitment, co-evolution, etc.), and several people have studied adversarial multi-agent situations (game theory, Markov games, etc.). Yet there has been little effort towards studying situations in which agents reason about collaborating with other benevolent agents while at the same time trying to outwit one or more opponents. Our agents within the simulated robotic soccer domain are the first to operate in a real-time, noisy, collaborative and adversarial environment.

TPOT-RL (Section 7.2) is a multi-agent RL algorithm for domains in which agents do not know the effects of their actions in terms of state transitions. While there are several multi-agent RL approaches mentioned in this section [Tan 93, Littman 94, Mataric 94a, Clouse 96, Schaerf *et al.* 95, Weiß 95], none is situated in an opaque-transition environment.

There have been some prior efforts towards having different agents change roles dynamically [Prasad *et al.* 96, Tambe 97]. But there is no previous system in which a team of agents operating in a real-time environment can dynamically change both its set of roles (formation) and role assignments within the formation. As illustrated in Figure 3.3 and implemented in the simulated robotic soccer implementation (Section 3.5.2) the flexible teamwork structure

defined within the team member agent architecture (Section 3.3) contributes a method for agents to smoothly switch both formations and roles within a formation.

In Sections 9.1.3 and 9.1.4, several communicating multi-agent systems were presented. However, none addresses communication in single-channel, low-bandwidth, unreliable communication environments. The communication paradigm presented in Section 3.4 contributes a method for teams to communicate in such environments by taking advantage of the locker-room agreement within the team member agent architecture.

Finally, layered learning is a new approach to learning in complex systems, including those with multiple agents. While several multi-agent learning systems are described in this section, none layers multiple machine learning techniques within a single application. As presented in Chapters 4–7, layered learning allowed us to connect learned individual skills, multi-agent behaviors, and team behaviors to create a coherent team of agents in a complex, multi-agent domain.

9.2 Robotic Soccer

Robotic soccer is a particularly good domain for studying multi-agent systems. The main goal of any test bed is to facilitate the trial and evaluation of ideas that have promise in the real world [Decker 96a]. A wide variety of real MAS issues can be studied in robotic soccer. In fact, all of the seventeen MAS issues listed in Table 9.1 can be feasibly studied in the soccer simulator.

Homogeneous, non-communicating MAS can be studied in robotic soccer by fixing the behavior of the opposing team and populating the team being studied with identical, mute players. To keep within the homogeneous agent scenario, the opponents must not be modeled as agents.

- In this context, the players can be *reactive or deliberative* to any degree. The extremely reactive agent might simply look for the ball and move straight at it, shooting whenever possible. At this extreme, the players may or may not have any knowledge that they are part of a team.
- On the other hand, players might *model each other*, thus enabling deliberative reasoning about whether to approach the ball or whether to move to a different part of the field in order to defend or to receive a pass.
- With players modeling each other, they may also reason about how to *affect each other's behaviors* in this inherently dynamic environment.
- It is possible to study the relative merits of *local and global perspectives* on the world. Robots can be given global views with the help of an overhead camera, and the soccer server comes equipped with an omniscient mode that permits global views. Simulated robotic soccer is usually approached as a problem requiring local sensing.

Heterogeneous, non-communicating MAS can also be studied in robotic soccer.

- Since each player has several teammates with the same global goal and several opponents with the opposite goal, each player is both *benevolent and competitive* at the same time. This possibility for combination of collaborative and adversarial reasoning is a major feature of the domain.
- If the teams are learning during the course of a single game or over several games, all the issues of *evolving agents*, including the "arms race" and the credit-assignment problem, arise.
- In the soccer server, stamina is a resource assigned to each individual agent. At the team level, stamina is important for *resource management*: if too many agents are tired, the team as a whole will be ineffective. Therefore, it is to the team's advantage to distribute the running among the different agents.
- When trying to collaborate, players' actions are usually interdependent: to execute a successful pass, both the passer and the receiver must execute the appropriate actions. Thus *modeling each other* for the purpose of coordination is helpful. In addition, if opponents' actions can be predicted, then proactive measures might be taken to render them ineffective.
- *Social conventions*, such as programmed notions of when a given agent will pass or which agents should play defense, can also help coordination. The locker-room agreement is an example of social conventions within a team.
- Since communication is still not allowed, the players must have a reliable method for filling the different team *roles* needed on a soccer team (e.g. defender, forward, goaltender). The flexible teamwork structure presented in Section 3.5.2 is one such method.

Homogeneous, communicating MAS can be studied by again fixing the behavior of the opposing team and allowing teammates to communicate.

- *Distributed sensing* can be studied in this context due to the large amount of hidden state inherent in the soccer server (see Section 2.2). At any given moment, a particular agent sees only a small portion of the world. By communicating with teammates, it can get a more complete picture of the world. In our communication paradigm presented in Section 3.4, the <selected-internal-state> field is designed for this purpose.

Heterogeneous, communicating MAS is perhaps the most appropriate scenario to study within the context of robotic soccer. Since the agents indeed are heterogeneous and can communicate, the full potential of the domain is realized in this scenario.

- With players sending messages to each other, they must have a language in order to *understand each other*. Our agents define such a language within the locker-room agreement.
- Especially in the single-channel, low-bandwidth communication environment modeled by the soccer server, agents must *plan their communicative acts*. If the opponents can understand the same language, a planned utterance can affect the

knowledge of both teammates and opponents. The utility of communication must be carefully considered and the possibility of lying in order to fool the opponent arises. In addition, the low-bandwidth creates the condition that sending a message may prevent other messages from getting through.

- Like in the heterogeneous, non-communicating scenario, since agents have both teammates and adversaries, they must reason about being both *benevolent and competitive*.
- *Negotiation* protocols may be useful in the robotic soccer domain if different agents, based on their different sensory perspectives, have different opinions about what course of action would be best for the team.
- In a real-time environment, timing is very important for any team play, including a simple pass. Thus, *resource management* in terms of timing, or action coordination, is crucial.
- Protocols are also needed for *commitment* to team plays: the passer and receiver in a pass play must both agree to execute the pass. For more complex team plays, such as our set-plays, several players may need to commit to participate. But then the issue arises of how single-mindedly they must adhere to the committed play: when may they react to more pressing situations and ignore the commitment?

As demonstrated above, all of the MAS issues summarized in Table 9.1 can be studied in robotic soccer. I now review the research that has been conducted in this domain. In Section 9.2.1, I describe research conducted in the "early years", before organized robotic soccer workshops, that served as the foundations for the recent popularity of the domain. Then, in Section 9.2.2, I review some of the research presented at dedicated robotic soccer workshops held in conjunction with the international competitions described in Chapter 8, as well other contemporary robotic soccer-related research.

9.2.1 Foundations

Producing natural language commentary from real-time input, the SOCCER system [Andre *et al.* 88] was the first AI research related to soccer. SOCCER analyzed *human* soccer games. By looking for triggers and terminations of events such as a player running or the ball being passed, SOCCER aims to announce important events without redundancy.

Robotic soccer was introduced as an interesting and promising domain for AI research at the Vision Interface conference in June, 1992 [Mackworth 93]. Dynamite, the first working robotic soccer system [Barman *et al.* 93, Sahota *et al.* 95] was also described at that time. A ground-breaking system for robotic soccer, and the one that served as the inspiration and basis for my work, the Dynamite test bed was designed to be capable of supporting several robots per team, but most work has been done in a 1 vs. 1 scenario. It uses an overhead camera and color-based detection to provide global sensory information to the robots. Dynamite was used to introduce a decision making strategy called reactive deliberation which was used to choose from among seven hard-wired behaviors [Sahota 94]. Subsequently, an RL approach based on high-level sensory predicates was used to choose from among the same hard-wired behaviors [Ford *et al.* 94].

Asada et al. [94a] developed the first robots equipped with on-board sensing capabilities. As mentioned in Section 5.5, these robots use learning from easy missions, an RL training technique, to learn to hit a stationary ball into the goal. One contribution of this work is the construction of state and action spaces that reduce the complexity of the learning task [Asada *et al.* 96]. As opposed to the action-dependent features used by TPOT-RL which create an abstract feature space prior to learning, states are clustered during learning based on the best action to take from each state. Another contribution is the combination of low-level behaviors, such as shooting and avoiding an opponent, that are learned using RL [Asada *et al.* 94b, Uchibe *et al.* 96]. Rather than building the learned behaviors at different behavior levels as in layered learning, two previously learned control strategies are used to produce a new one, which then replaces the original two.

Minimax-Q learning for Markov games was first applied in an abstract simulated soccer game [Littman 94]. This version of the domain is much simpler than the soccer server, having 800 states, 5 actions, and no hidden information. One player on each team moves in a grid world and the ball is always possessed by one of the players. Using minimax-Q, players learn optimal probabilistic policies for maneuvering past each other with the ball.

In research prior to that reported in this dissertation, we conducted machine learning experiments in a simulator based closely on the Dynasim simulator [Sahota 96] which simulates the Dynamite robots mentioned above. First, we used memory-based learning to allow a player to learn when to shoot and when to pass the ball [Stone and Veloso 96a]. We then used neural networks to teach a player to shoot a moving ball into particular parts of the goal [Stone and Veloso 96b]. Based on training in a small region of the field, our agent was able to learn to successfully time its approach to a moving ball such that it could score from all areas of the field. These experiments served as the basis for our initial learning experiments in the soccer server [Stone and Veloso 96c].

In another early learning experiment in the soccer server, a player learned when to shoot and when to pass [Matsubara *et al.* 96]. As described in Section 5.5, the agent bases its decision on the positions of the ball, the goaltender, and one teammate.

9.2.2 The Competition Years

The research reported in Section 9.2.1 confirmed the potential of robotic soccer as an AI research domain and justified the value of having large-scale competitions from a research perspective. Starting with the first competitions held in 1996 (Pre-RoboCup-96 and MiroSot-96) and continuing since then, there has been a great deal of robotic soccer-related research. It has been presented both at dedicated robotic soccer workshops held in conjunction with the competitions and in other scientific forums. In this section I review some of this recent robotic soccer research.

Robot Hardware

Much of the research inspired by competitions has been devoted to building robot hardware that is suitable for this challenging environment, e.g. [Achim *et al.* 96, Hong *et al.* 96, Hsia and Soderstrand 96, Kim *et al.* 96, Shim *et al.* 96]. The emphasis in hardware approaches

varies greatly. Some research focuses on fast and robust visual perception of the environment [Sargent *et al.* 97, Cheng and Zelinsky 98, Han and Veloso 98]. In response to the need for vision systems that work under various lighting conditions (conditions at competitions are never the same as in the lab), some research focuses on automatic calibration of vision parameters [Shen *et al.* 98, Veloso and Uther 99]. Instead of vision, one alternative approach is to use a laser range-finder for localization in the environment [Gutmann *et al.* 98].

Other research focuses on robot path planning in crowded, dynamic, environments [Han *et al.* 96, Kim and Chung 96]. Path planning is particularly challenging with non-holonomic robots because they can only move straight in the direction that they are facing or in curved paths starting from their current location and direction. Our approach to non-holonomic robotic path planning is described in Appendix B, specifically in Section B.2.1. Omnidirectional robots can simplify path planning considerably: they do not have to consider the direction they are facing as a constraint [Price *et al.* 98, Yokota *et al.* 98].

In addition to robots developed specifically for the competitions, there have been robots created to exhibit special soccer-related skills. Shoobot [Mizuno *et al.* 96, Mizuno *et al.* 98] is a nomad-based robot that can dribble and shoot a soccer ball as it moves smoothly through an open space. The Sony legged robots [Fujita and Kageyama 97] walk on four legs. They have been used as the basis of an exclusively legged-robot soccer competition [Veloso *et al.* 98c]. And the Honda humanoid robots [Hirai 97] have been demonstrated kicking a real soccer ball and performing a penalty shot with a shooting and a goaltending robot. This demonstration indicates the feasibility of RoboCup's long-term goal of having a humanoid robot soccer competition on a real soccer field [Kitano *et al.* 98].

Soccer Server Accessories

This dissertation uses the soccer server described in Section 2.2 as a test bed for research in MAS and ML. In order to test the teamwork structure and communication paradigm presented in Chapter 3, our agents first needed to be equipped with the low-level individual skills described in Sections 3.5.1 and B.1 (Appendix B). Other approaches to building individual skills in the soccer server are also presented in the literature, e.g. [Burkhard *et al.* 98, Pagello *et al.* 98].

In addition to soccer-playing agent development, the soccer server has been used as a substrate for 3-dimensional visualization, real-time natural language commentary, and education research.

Figure 2.1 shows the 2-dimensional visualization tool that is included in the soccer server software. SPACE [Shinjoh 98] converts the 2-dimensional image into a 3-dimensional image, changing camera angle and rendering images in real time.

Another research challenge being addressed within the soccer server is producing natural language commentary of games as they proceed. Researchers aim to provide both low-level descriptions of the action, for example announcing which team is in possession of the ball, and high-level analysis of the play, for example commenting on the team strategies being used by the different teams. Commentator systems for the soccer server include ROCCO [Andre *et al.* 98b], MIKE [Matsubara *et al.* 99], and Byrne [Binsted 99].

Robotic soccer has also been used as the basis for education research. A survey of

RoboCup-97 participants indicates that the majority of participants were students motivated principally by the research opportunities provided by the domain [Verner 99]. There has also been an undergraduate AI programming course based on teaching students to create robotic soccer-playing agents in the soccer server [Coradeschi and Malec 99].

Multi-Agent Control and Robotic Soccer Strategy

The robotic soccer domain has inspired many different approaches to building and organizing teams of agents, including the team member agent architecture and layered learning as presented in this dissertation.

Some research is based on applying existing programming methodologies to the robotic soccer domain. Team GAMMA [Noda 98] is built using Gaea [Nakashima *et al.* 95], a logic programming language that is essentially a multi-threaded, multi-environment version of prolog. Gaea implements a dynamic subsumption architecture, allowing agents to override behaviors in different ways based on the current environment, or behavior context. Team ROGI [de la Rosa *et al.* 97] is built using another programming methodology, namely agent-oriented programming [Shoham 90].

Other research, like our own, introduces new multi-agent control methodologies and applies them to robotic soccer. For example, the MICROB robotic soccer team is an implementation of the Cassiopeia programming method [Drogoul and Collinot 98]. Cassiopeia focuses on the organizational issues of multi-agent tasks, analyzing the interdependencies of low-level skills and facilitating the formation of groups based on these inter-dependencies. Temporary organizations are formed based on the contract net framework [Smith 80]. For example, the player with the ball might contract with another player to place itself in a particular location to receive a pass. This approach differs from our own in that our agents position themselves autonomously, for instance using SPAR, and the agent with the ball decides autonomously where to pass: no negotiation is involved, enabling the players to act as quickly as possible.

Another multi-layered approach to robotic soccer is presented by Scerri [98]. However, unlike our own hierarchical approach, it does not involve the learning of any behaviors. In this approach, the different abstraction layers deal with different granularities of sensory input. For example, a low-level move-to-ball behavior is given the ball's precise location, while a high-level defend behavior—which might call go-to-ball—knows only that the ball is in the defensive half of the field. The Samba control architecture [Riekki and Roening 98] uses two behavior layers: the reactive layer which defines action maps from sensory input to actuator output; and the task layer which selects from among the action maps.

ISIS [Tambe *et al.* 98] is a role-based approach based on STEAM [Tambe 97]. STEAM defines team behaviors that can be invoked dynamically. There has also been another formation-based approach to positioning agents on the soccer field [Matsumoto and Nagai 98]. However, unlike in our dynamic formations with flexible positions, the player positions are static and the team formation cannot change dynamically. Several other researchers also recognize the importance of decomposing the soccer task into different roles, e.g. [Coradeschi and Karlsson 98, Ch'ng and Padgham 98].

One approach with dynamically changing roles is developed in a soccer simulator other

than the soccer server [Balch 98]. Balch uses a behavioral diversity measure to encourage role learning in an RL framework, finding that providing a uniform reinforcement to the entire team is more effective than providing local reinforcements to individual players.

Often, definitions of robotic soccer positions involve fixed locations at which an agent should locate itself by default, e.g. [Gutmann *et al.* 98,Matsumoto and Nagai 98]. In contrast, as described in Section 3.5.2, our flexible positions allow players to adjust their locations within their roles. The ranges of flexibility are defined a priori as a part of the locker-room agreement. Observational reinforcement learning [Andou 98] allows agents to *learn* their positions dynamically based on the distribution of past ball locations in a game. A similar approach is also described in [Inoue and Wilkin 97].

In another learning approach, teammate and opponent capabilities are learned through repeated trials of specific actions [Nadella and Sen 97]. This research is conducted in a soccer simulator in which the ball is always in possession of a player, eliminating the necessity for fine ball control. Each player has an assigned efficiency in the range $[0, 1]$ for the execution of actions such as passing, tackling, and dribbling corresponding to the probability that the action will succeed. Agents do not know the abilities of themselves, their teammates, or the opponents. Instead, they learn to estimate them based on repeated trials. The agents can then base action decisions on the learned parameters.

All of the learning approaches described above, as well as the approach described in this dissertation, are used to learn portions of an agent's behavior. Other aspects are created manually. In contrast, a few entirely learned soccer behaviors have been created.

Hexcer [Uther and Veloso 97] is an extension of the grid world soccer game described above [Littman 94]. Rather than square grid locations, the world is defined as a lattice of hexagons. Thus the action space is increased and the geometric constraints are altered. The added complexity necessitates the development of generalized U-trees to allow agents to learn successful policies [Uther and Veloso 97]. In hexcer, it is possible to learn straight from sensors to actuators because, like Littman's simulation, it has a much smaller state space than the soccer server and the agents have no hidden state.

As mentioned in Section 4.4.1, the RoboCup-97 and RoboCup-98 competitions each included one team created using genetic programming [Koza 92]. In both cases, the goal was to learn entirely from agent sensors to actuators in the soccer server. The first attempt [Luke *et al.* 98] was eventually scaled down, although a successful team was created based on some manually created low-level skills. The following year, Darwin United [Andre and Teller 99] entered an entirely learned team.

9.2.3 Summary

Within this large and growing body of research in the robotic soccer domain, this dissertation fills several gaps:

- While there are several role-based approaches, and at least one examination of possible team formations ([Matsumoto and Nagai 98]), no other robotic soccer system enables its agents to dynamically switch both roles and formations.

- No other implemented system includes pre-defined, multi-agent plans (set-plays) for frequently occurring situations.
- Our communication paradigm for single-channel, low-bandwidth, unreliable communication environments, such as that of the soccer server, is the only such paradigm presented.
- While there have been many learning approaches to robotic soccer, including several within the soccer server, no other system layers multiple learning modules as we do using layered learning.

Nonetheless, even with all of the above contributions by us and by others, there is still a lot of room for new AI research within the robotic soccer domain. The next and concluding chapter outlines some of these potential future directions as well as summarizing the contributions of this dissertation.