Interaction and Intelligent Behavior

by

Maja J Matarić

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis addresses situated, embodied agents interacting in complex domains. It focuses on two problems: 1) synthesis and analysis of intelligent group behavior, and 2) learning in complex group environments.

Basic behaviors, control laws that cluster constraints to achieve particular goals and have the appropriate compositional properties, are proposed as effective primitives for control and learning. The thesis describes the process of selecting such basic behaviors, formally specifying them, algorithmically implementing them, and empirically evaluating them. All of the proposed ideas are validated with a group of up to 20 mobile robots using a basic behavior set consisting of: *safe–wandering*, *following*, *aggregation*, *dispersion*, and *homing*. The set of basic behaviors acts as a substrate for achieving more complex high–level goals and tasks. Two behavior combination operators are introduced, and verified by combining subsets of the above basic behavior set to implement collective *flocking*, *foraging*, and *docking*.

A methodology is introduced for automatically constructing higher–level behaviors by learning to select among the basic behavior set. A novel formulation of reinforcement learning is proposed that makes behavior selection learnable in noisy, uncertain multi–agent environments with stochastic dynamics. It consists of using **conditions** and **behaviors** for more robust control and minimized state–spaces, and a reinforcement shaping methodology that enables principled embedding of domain knowledge with two types of shaping functions: **heterogeneous reward functions** and **progress estimators**. The methodology is validated on a collection of robots learning to forage. The generality of the approach makes it compatible with the existing reinforcement learning algorithms, allowing it to accelerate learning in a variety of domains and applications.

The presented methodologies and results are aimed at extending our understanding of synthesis, analysis, and learning of group behavior.

Thesis Supervisor: Rodney A. Brooks
Title: Professor of Computer Science and Engineering

Chapter 1

Overview of the Thesis

One of the main goals of Artificial Intelligence (AI) is to gain insight into natural intelligence through a synthetic approach, by generating and analyzing artificial intelligent behavior. In order to glean an understanding of a phenomenon as complex as natural intelligence, we need to study complex behavior in complex environments.

Traditionally, AI has concerned itself with complex agents in relatively simple environments, simple in the sense that they could be precisely modeled and involved little or no noise and uncertainty. In contrast to traditional systems, reactive and behavior–based systems have placed agents with low levels of cognitive complexity into complex, noisy and uncertain environments. This thesis describes work that attempts to simultaneously scale up along both dimensions. The environmental complexity is scaled up by introducing other agents, and cognitive complexity is scaled up by introducing learning capabilities into each of the agents (Figure 1-1).

This thesis addresses two problems:

1. synthesis and analysis of intelligent group behavior

2. learning in complex group environments

Our ideas are based on the notion of *basic behaviors*, a means for combining constraints from the agent, such as its mechanical and sensory characteristics, and the constraints for the environment, such as the types of interactions and sensory information the agent can obtain, in order to construct an appropriate abstraction for structuring primitives for control.

We will present a methodology that uses basic behaviors to generate various robust group behaviors, including following, homing, and flocking (Figure 1-2). We will also introduce a formulation of reinforcement learning based on behaviors as the unit of representation that allows a group of agents to learn complex tasks such as foraging

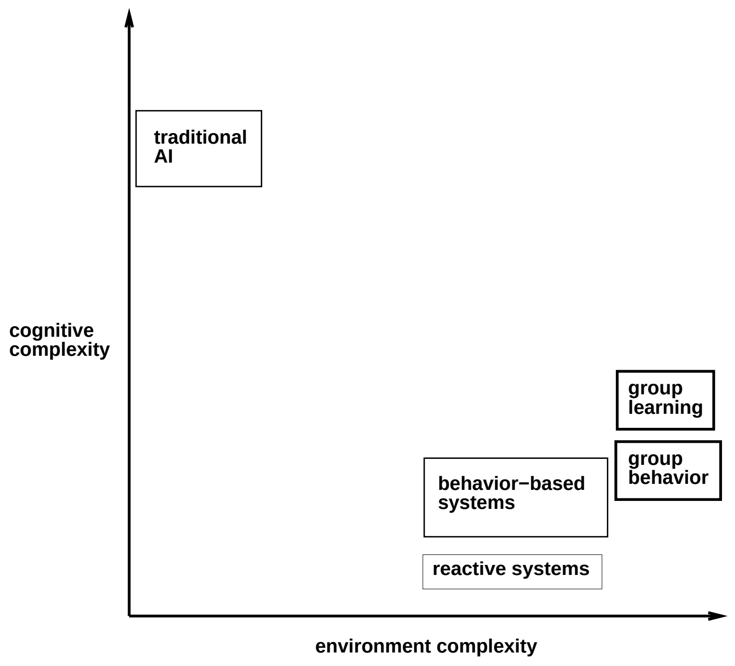


Figure 1-1: Traditional AI has addressed complex agents in simple environments while reactive and behavior–based approaches have dealt with simple agents in noisy and uncertain worlds. This work attempts to scale up along both dimensions simultaneously, by addressing synthesis and learning of complex group behavior.

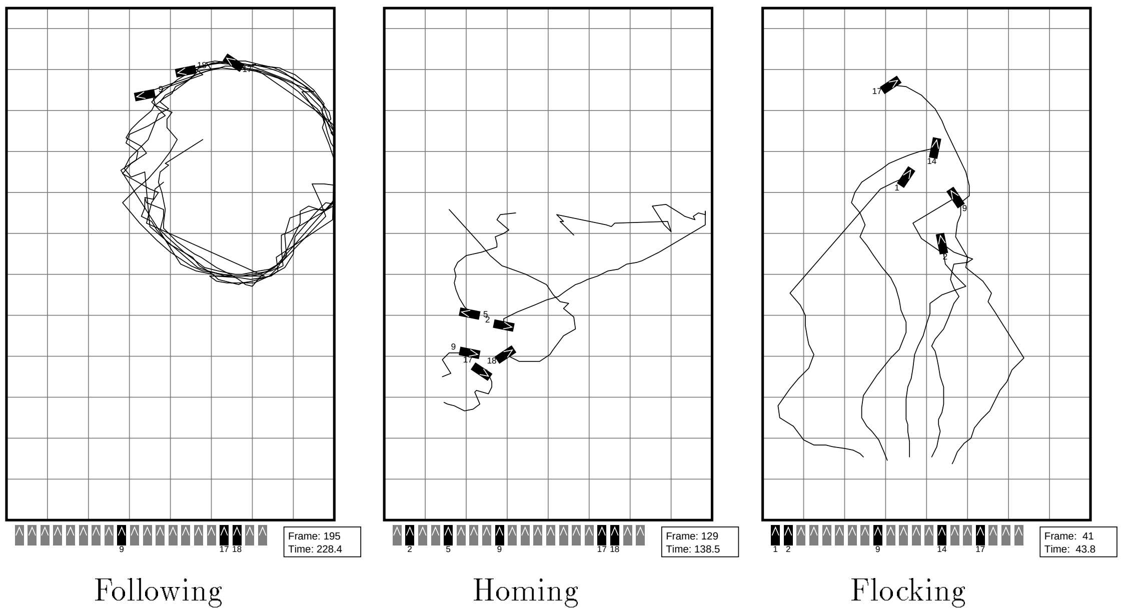


Figure 1-2: This figure shows examples of real robot data for three different group behaviors: following, homing, and flocking. The robots, physically 12 inches long, are scaled down and plotted as black rectangles, with white arrows indicating their heading. The dark robots in the row of rectangles at the bottom shows the robots that were used in the experiment. Boxes on the lower right indicate frame numbers and the elapsed time in seconds for each of the runs.

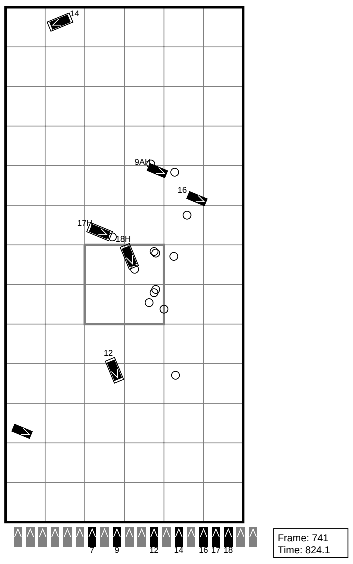


Figure 1-3: An example of the foraging behavior of 7 robots, shown after 13.7 minutes of running. About eight pucks have been delivered to the home region, marked with a grey box. The two robots near home are following each other on the way to the drop-off. Other robots are wandering in search of additional pucks.

(Figure 1-3). Finally, we will validate the proposed approaches with experiments on homogeneous groups of mobile robots.

This chapter gives a brief summary of the novel approaches, of the experimental data, and of the implications of the thesis. The organization of the thesis is outlined at the end of the chapter.

1.1 Synthesis and Analysis of Group Behavior

This thesis is based on the belief that intelligent collective behavior in a decentralized system results from *local interactions* based on simple rules. **Basic behaviors** are proposed as a methodology for structuring those rules through a principled process of synthesis and evaluation. A *behavior* is a control law that clusters a set of constraints in order to achieve and maintain a goal. For example, *safe–wandering* is a behavior that maintains the goal of avoiding collisions while the agent is moving.

We postulate that, for each domain, a set of behaviors can be found that are **basic** in that they are required for generating other behaviors, as well as being a minimal set the agent needs to reach its goal repertoire. The process of choosing the set of basic behaviors for a domain is dually constrained. From the bottom–up, the process is constrained by the dynamics of the agent and the environment. From the top–down, the process is constrained by the agent's goals as specified by the task. The combination of the two types of constraints helps to prune the agent's behavior space.

We will use the example of group interactions between situated, embodied agents to illustrate the process of selecting a basic behavior set. The agents are mobile robots, embodied and endowed with specific mechanical, sensory, and effector constraints. We define the high–level goals of the system as consisting of collectively moving objects (pucks) in the environment in an efficient fashion. In this work, efficiency is defined in terms of minimizing energy by minimizing the amount of time required to complete a task or the number of moves required for each of the agents.

An effective set of basic behaviors in the spatial domain should enable the agents to employ a variety of flexible strategies for puck manipulation, collection, and distribution. The effectiveness of such strategies depends on maximizing synergy between agents: achieving the necessary goals while minimizing inter–agent interference.

We propose the following set of basic behaviors:

- *safe–wandering* – minimizes collisions between agents and environment
- *following* – minimizes interference by structuring movement of any two agents
- *aggregation* – gathers the agents
- *dispersion* – dissipates the agents
- *homing* – enables the agent to proceed to a particular location

According to our definition, the above behavior set is *minimal* or *basic* in that its members are not further reducible to each other. Additionally, we will show that they are sufficient for achieving the set of pre–specified goals. The described basic behaviors are defined with respect to the group. Other utility behaviors, such as *grasping* and *dropping*, can also be a part of an agent's repertoire.

The basic behavior set is evaluated by giving a formal specification of each of the behaviors, and comparing the collection of those specifications to a formal specification of the set of global tasks required for the group.

Once a basic behavior set is established, it can be implemented with a variety of algorithms. The first step in the verification of basic behavior algorithms is a comparison between the formal behavior specification and the formal correctness of the algorithm. We will argue that it is difficult to prove properties of the exact behavior of individual agents within a group, but it is possible to evaluate and predict the behavior of the ensemble as a whole. Using the notion of ensemble behavior, we

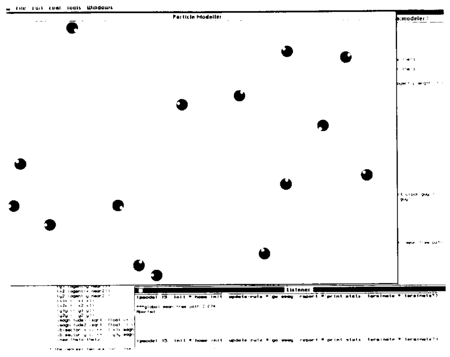


Figure 1-4: The simulator environment called the Interaction Monitor was used to validate the methodologies for synthesizing and analyzing group behavior described in the thesis. The agents are shown as black circles, with white markers indicating their heading. The large rectangle represents the agents' workspace.

will propose group behavior algorithms that utilize a centroid operator that averages the inputs from multiple agents. This operator has statistical properties that allow analyzing and making predictions about the behavior of the group.

This thesis provides detailed specifications and algorithms for each of the basic behaviors. Instead of analytical proofs, it provides empirical evaluations of the performance of each of the algorithms, based on the following criteria:

- repeatability: how consistent is the behavior over different trials?
- stability: does the behavior oscillate under any conditions?
- robustness: how robust is the behavior in the presence of sensor and effector error and noise?
- scalability: how is the behavior effected by increased and decreased group sizes?

The above criteria were applied to the data obtained by performing at least 50 trials of each basic behavior. The experiments were performed on two different multi–agent environments, in order to minimize domain biases. The first environment was a multi–agent simulator (the Interaction Monitor) featuring up to 50 agents with local sensing and distributed, local control (Figure1-4).

The second environment was a collection of 20 physical mobile robots equipped with local sensors and local control (Figure 1-5). Each of the robots is equipped with a suite of infra–red sensors for collision avoidance, puck detection, and stacking, and with micro switches and bump sensors for contact detection. In addition to the

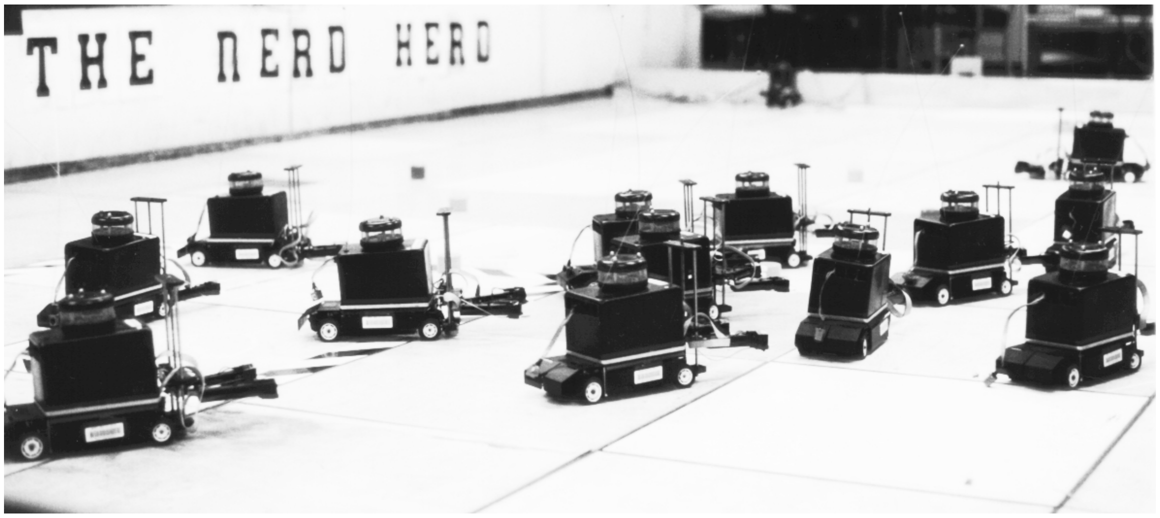


Figure 1-5: Some of the 20 mobile robots used to validate the group behavior methodologies described in the thesis. These robots demonstrated group safe–wandering, following, aggregation, dispersion, flocking, and foraging.

local sensors, the robots are equipped with radios and sonars for triangulating their position relative to two stationary beacons, and for broadcasting that position within a limited radius. The radios are used to detect other robots and gather data for local centroid computations.

The basic behaviors, each consisting of one rule or a small set of simple rules, generated robust group behaviors that met the prespecified evaluation criteria. A small subset of the data is shown here, using the Real Time Viewer[1], a software package for displaying and replaying each of the robots runs, plotting their positions over time, and displaying each frame and the elapsed time for each experiment. The figures show following (Figure 1-6), dispersion (Figure 1-7), and homing (Figure 1-8). More of the data, the algorithms, the specifications, and a detailed evaluation can be found in Chapter 4.

Basic behaviors are intended as building blocks for achieving higher–level goals. The behaviors are embedded in an architecture that allows two types of combination: direct, by summation, and temporal, by switching (see figure 1-9). Both types of combination operators were tested empirically. A simple and robust *flocking* behavior was generated by summing the outputs of *safe–wandering*, *aggregation*, and *homing* (Figure 1-10). A more complex *foraging* behavior that involves finding and collecting pucks, was implemented by switching between *safe–wandering*, *dispersion*, *following*,

[1]Written by Matthew Marjanović.

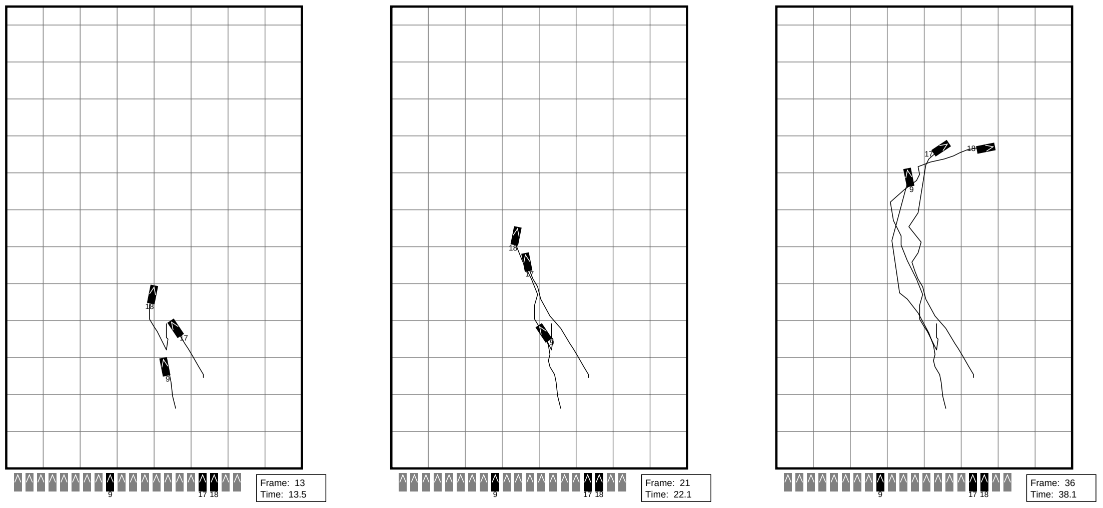


Figure 1-6: Continuous following behavior of 3 robots. The entire time history of the robots' positions is plotted.

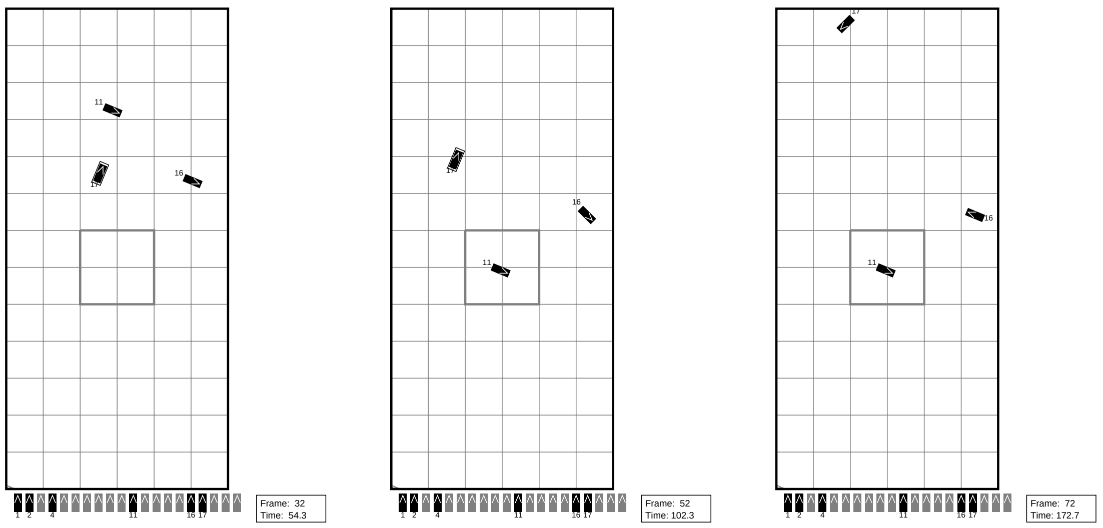


Figure 1-7: Dispersion behaviors of 3 robots.

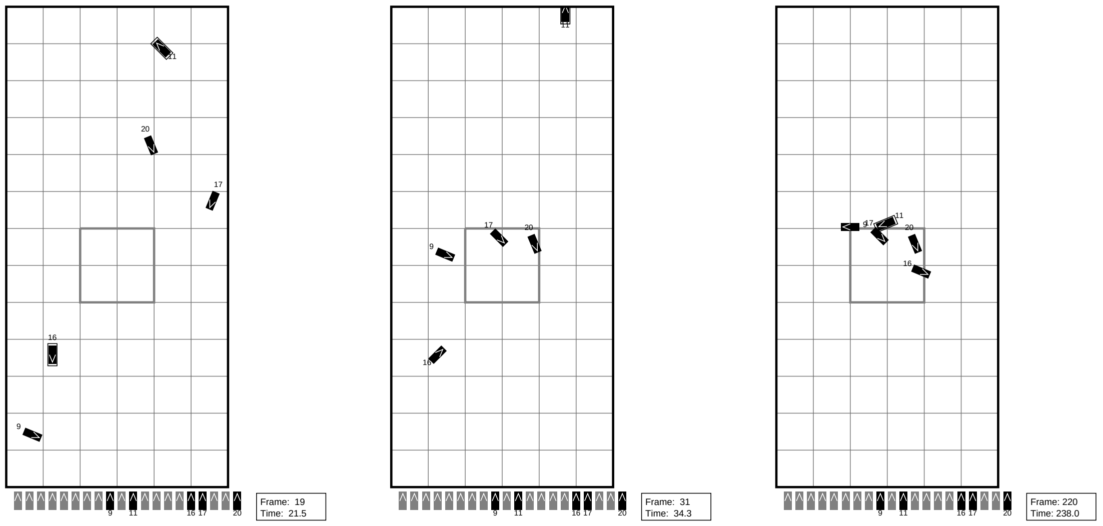


Figure 1-8: Homing behaviors of 5 robots. Four of the five robots reach home quickly and the fifth joints them about 60 second later.

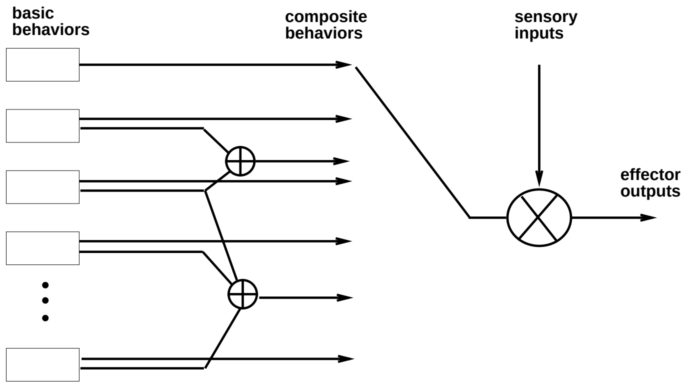


Figure 1-9: The control architecture for generating group behaviors consists of direct and temporal combinations (i.e. sums and switches) of subsets from a fixed basic behavior set. Direct combinations are marked with $\oplus$, temporal combinations with $\otimes$.

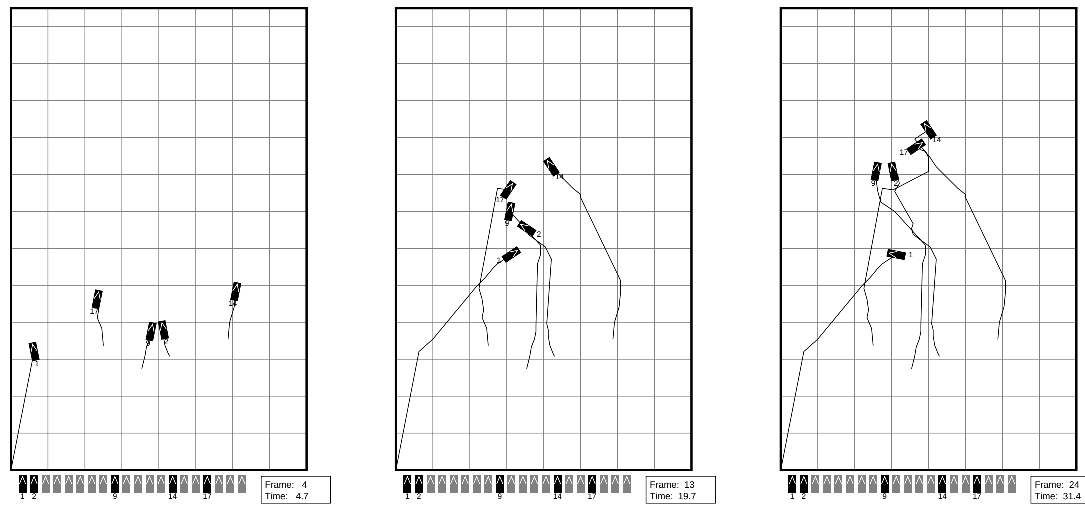


Figure 1-10: Flocking behavior of 5 robots. The robots are started out in a nearly linear dispersed state. They quickly establish a flock and maintain it as the positions of the individual robots within the flock fluctuate over time.

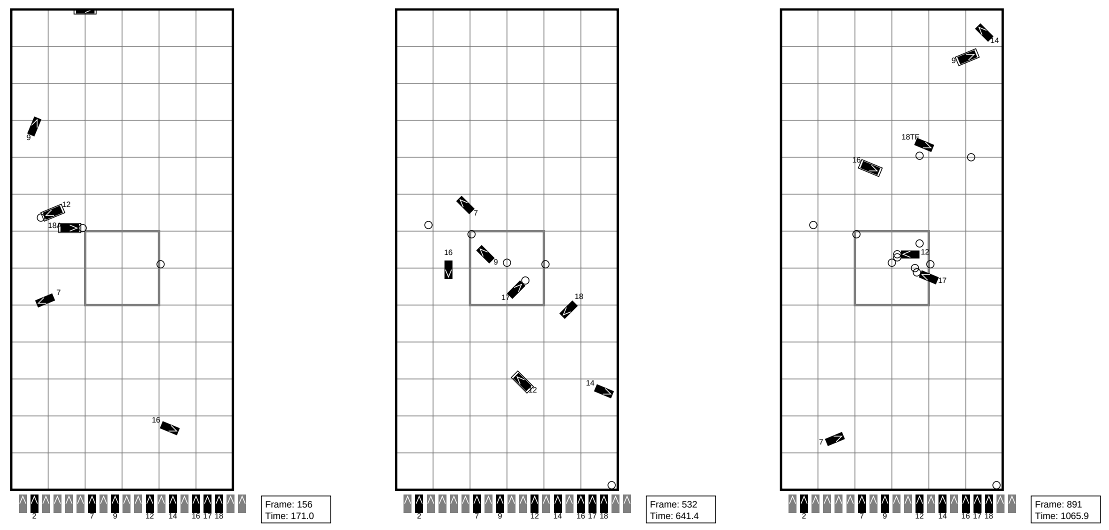


Figure 1-11: An example of the foraging behavior of 6 robots. About eight pucks have been delivered to the home region, marked with a grey box. Two of the robots are dropping off pucks while the others are wandering in search of additional pucks to pick up and deliver home.

and *homing* (Figure 1-11).

In addition to empirical testing of the behaviors and their combinations, the proposed methodology for generating decentralized group behavior was compared to a centralized, "total knowledge" approach. The experimental results showed that the simple, fully distributed strategies, applied to *dispersion* and *aggregation* tasks, converged only a constant factor slower than the centralized approach.

1.2 Learning in Complex Group Environments

The first part of the thesis introduces basic behaviors as a methodology for structuring simple rules into flexible and effective repertoires of group behavior. It also presents combination operators that allow for constructing and achieving higher–level goals. The second part of the thesis, starting with Chapter 6, describes a methodology for automatically combining basic behaviors into higher–level ones, though unsupervised **reinforcement learning** based on the agents' interactions with the environment.

In reinforcement learning (RL) approaches the agent learns from external scalar reward and punishment. RL has been successfully applied to a variety of domains that have largely been modeled as Markovian, where the agent–environment interaction can be described as a Markov Decision Process (MDP). However, the MDP assumption does not directly apply to the noisy and uncertain multi–agent environments addressed in this work. Nonetheless, since external and internal feedback are the most natural sources of information for learning in situated agents, methods for applying RL to such complex domains are needed.

The traditional formulation of RL problems in terms of states, actions, and reinforcement required a reformulation in order to be applied to our domain. The notion of state as a monolithic descriptor of the agent and the environment did not scale up to the multi–agent domain used here, given the continuous and discrete aspects describing the agent (e.g., velocity, IR sensors, radio data), and the existence of many other agents in the environment. Furthermore, the most commonly used notion of actions was inappropriate since atomic actions were too low level and had effect too unpredictable and noisy to be useful to a learning algorithm. Finally, delayed reinforcement and reward discounting were insufficient for learning in our domain.

To make learning possible we propose a reformulation that elevates the level of system description from states and actions to conditions and behaviors. *Behaviors* are control laws that achieve goals but hide low–level control details. Using the notion of *basic behaviors*, a small basis set can be defined as used as a substrate for learning. When actions are replaced with behaviors, states can be replaced with *conditions*, the necessary and sufficient subsets of state required for triggering the behavior set.

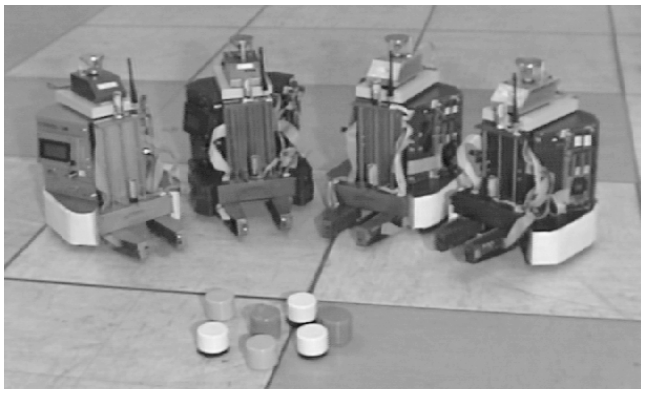


Figure 1-12: The mobile robots used to validate the group behavior and learning methodologies described in this thesis. These robots demonstrated learning to forage by using group safe–wandering, following, and resting behaviors.

Conditions are many fewer than states, thus greatly diminishing the agent's learning space and speeding up any RL algorithm.

In addition to the use of behaviors and conditions, we propose two ways of shaping the reinforcement function in order to aid the learner in a nondeterministic, noisy, and dynamic environment. We introduced *heterogeneous reward functions* that partition the task into subgoals, thus providing more immediate reinforcement. Within a single behavior (i.e., a single goal), we introduced *progress estimators*, functions associated with particular conditions that provided some metric of the learner's performance. Progress estimators, or internal critics, decrease the learner's sensitivity to noise, minimize thrashing, and minimize the effect of fortuitous rewards by correlating some domain knowledge about progress with appropriate behaviors the agent has taken in the past. The details of the reformulation are given in Chapter 7.

The proposed formulation was validated on the task of learning to associate the conditions and behaviors for group foraging with a collection of robots. The behaviors included the foraging subset of basic behaviors: *safe–wandering*, *dispersion*, and *homing*, augmented with *grasping* and *dropping*, as well as with *resting*, a new behavior triggered by an internal "day–time night–time" clock. By clustering, the condition set was reduced to the power set of the following predicates: *have-puck?*, *at-home?*, *night-time?*, and *near-intruder?*.

A smaller group of robots with more reliable hardware was used for the learning experiments. In terms of sensors and effectors, the robots were functionally identical to the first set (Figure 1-12), and the implemented basic behaviors and combinations were directly portable.

Three learning algorithms were implemented and tested on the foraging task. The

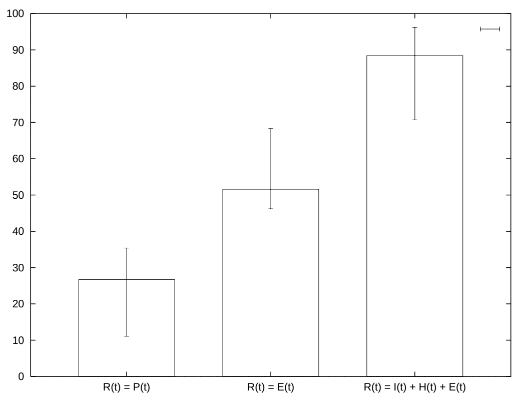


Figure 1-13: The performance of the three reinforcement strategies on learning to forage. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the agents learned, averaged over twenty trials.

first was standard Q-learning, while the other two simply summed the reinforcement received over time.

Q-learning received a reward whenever a robot dropped a puck in the home region. The second algorithm was based on the reinforcement received from heterogeneous reward functions based on reaching subgoals including grasping and dropping pucks, and reaching home. The third algorithm used reinforcement both from the heterogeneous reward functions and from two progress estimators: one monitoring progress in getting away from an intruder, and the other monitoring progress toward home. The two progress estimators were found to be sufficient for making the given learning task possible and for consistent and complete learning performance. The absence of either one disabled the robots from learning the complete policy.

The performance of each of the three algorithms was averaged over 20 trials (Figure 1-13). The analysis of the learning performance showed that the parts that were not learned by the first two algorithms relied on the progress estimators and were successfully learned in the third case. Detailed analysis of the results is given in Chapter 8.

1.3 Thesis Outline

The preceding sections briefly summarized the contributions of the thesis. This section outlines the structure of the thesis and summarizes each of the chapters.

Chapters 2 through 5 deal with synthesizing and analyzing group behavior. Chapters 6 through 8 address learning in multi–agent domains. Readers interested in moving directly to the details of the basic behavior approach should skip to Chapter 4. Those interested in going directly to the learning part of the thesis should skip to Chapter 6. All newly introduced, ambiguous, or frequently used terms are defined in Appendix B. The following are summaries of the chapter contents.

Chapter 2 describes the biological, sociological, and pragmatic motivation behind this work. It describes the key issues in individual and multi agent control, and introduces and defines the main concepts of the thesis.

Chapter 3 presents an overview of related work in Robotics, Simulation, Artificial Life, Distributed AI, and analysis of behavior.

Chapter 4 introduces the basic behavior approach, describes the methodology for selecting basic behaviors, and illustrates the process by defining the basic behaviors for a collection of mobile agents interacting in the plane. The chapter describes the experimental environments, basic behavior specifications and algorithms, and the empirical data and the criteria for evaluating the performance of each of the behaviors as well as their efficacy relative to centralized alternatives.

Chapter 5 describes two methodologies for combining basic behaviors into more complex, higher–level behaviors. The methodologies are demonstrated by combining the basic behaviors described in Chapter 4 to generate three different kinds of higher–level behaviors and evaluate their performance. This chapter also discusses methods for minimizing interference between behaviors within an agent.

Chapter 6 motivates learning in situated agents and reviews the existing learning work based on the type of information being acquired by the agent. It then defines the group learning problem discussed in the thesis as an instance of reinforcement learning (RL) and overviews existing RL models and algorithms as applied to the situated agent domain.

Chapter 7 describes a formulation of RL that enables and facilitates learning in our complex situated multi–agent domain. It introduces the use of behaviors and their conditions in place of actions and states, and describes a method for shaping the learning process through the use of heterogeneous reward functions and progress estimators.

Chapter 8 presents the experimental robot environment and the learning task used to validate the methodologies proposed in Chapter 7. It describes the experimental design, the three learning algorithms that were implemented and compared, and discusses the results. In conclusion, the chapter addresses extensions of the presented work including the problem of learning social rules and multiple concurrent tasks.

Chapter 9 summarizes the thesis.

Chapter 4

The Basic Behavior Approach

One of the hardest problems in AI is finding the right level of system description for effective control, learning, modeling, and analysis. This thesis proposes a particular description level, instantiated in so-called **basic behaviors**, building blocks for synthesizing and analyzing complex group behavior in multi–agent systems.

Biology provides evidence in support of basic behavior units at a variety of levels. A particularly clean and compelling case can be found in motor control. Controlling a multi–joint manipulator such as a frog leg or a human arm is a complex task, especially if performed at a low level. In order to cut down the complexity, nature imposes an abstraction. Mussa-Ivaldi & Giszter (1992) show that a relatively small set of basis vector fields, found in the frog's spine, generates the frog's entire motor behavior repertoire by applying appropriate combinations of the basis vectors. Bizzi, Mussa-Ivaldi & Giszter (1991) and Bizzi & Mussa-Ivaldi (1990) discuss control of the human arm with a similar approach. The described motor basic behaviors are a result of the types of constraints: the dynamics of the manipulator and the dynamics of the motor tasks. In the case of motor control, the behaviors are designed for specific optimizations, such as minimizing effort by minimizing jerk, executing straight line trajectories, and using bell–shaped velocity profiles (Atkeson 1989).

Taking the idea from motor control, we define *behaviors* as control laws that encapsulate sets of constraints so as to achieve particular goals. *Basic behaviors* are defined as a minimal set of such behaviors, with appropriate compositional properties, that takes advantage of the dynamics of the given system to effectively accomplish its repertoire of tasks.

Basic behaviors are intended as a tool for describing, specifying, and predicting group behavior. By properly selecting such behaviors one can generate repeatable and predictable group behavior. Furthermore, one can apply simple compositional

Problem	Synthesis and analysis of intelligent group behavior in order to understand the phenomenon (science) and apply it (engineering).
Assertion	Complex group behavior results from local interactions based on simple rules.
Approach	Propose basic behaviors for structuring such simple rules.
Validation	Implement robot group behaviors using a basic behavior set and combinations.

Table 4.1: A summary of the group behavior problem being addressed in the thesis, and the structure of the proposed solution.

operators to generate a large repertoire of higher–level group behaviors from the basic set.

The idea behind basic behaviors is general, but particular sets of such behaviors are domain–specific. In order to demonstrate the methodology, basic behaviors for group interaction in the spatial domain will be derived, combined, analyzed theoretically, and tested empirically. Table 4.1 summarizes the research goals, the approach, and the experimental methodology.

4.1 Selecting and Evaluating Basic Behaviors

This chapter describes how **basic behaviors** are selected, specified, implemented, and evaluated. The idea of basic behaviors is general: they are the intended as primitives for structuring, synthesizing, and analyzing system behavior, as building blocks for control, planning, and learning. Basic behaviors are related to dynamic attractors, equilibrium states, and various other terms used to describe stable, repeatable, and primitive behaviors of any system. This work is concerned with finding ways of identifying such behaviors for a specific system, and using them to structure the rest of the system's behavioral repertoire. The power of basic behaviors lies in their individual reliability and in their compositional properties.

This work focuses on basic behaviors for generating intelligent group interactions in multi–agent systems. It is based on the belief that global behavior of such systems results from local interactions, and furthermore, that those interactions are largely governed by simple rules. Basic behaviors present a mechanism for structuring the space of possible local rules into a small basis set.

This chapter will illustrate the process of selecting basic behaviors on concrete examples of behaviors for a group of agents interacting in physical space. The process of identifying the basic behaviors, formally specifying them, implementing them, testing their properties both theoretically and empirically, and finally combining them, will be carried out. The criteria for selecting basic behaviors for the domain of spatially interacting agents are described first.

4.1.1 Criteria for Selection

We propose that, for a given domain, a small set of basis or *basic behaviors* can be selected, from which other complex relevant and desirable group behaviors can be generated. Basic behavior sets should meet the following criteria:

Necessity: A behavior within a basic behavior set is necessary if it achieves a goal required for the agent's accomplishment of its task(s), and that goal cannot be achieved with any of the other basic behaviors or their combinations. Thus, a basic behavior cannot be implemented in terms of other behaviors and cannot be reduced to them.

Sufficiency: A basic behavior set is sufficient for accomplishing a set of tasks in a given domain if no other behaviors are necessary. The basic behavior set should, under the combination operators, generate all of the desirable higher–level group behaviors.

If such behaviors are designed by hand, as opposed to being observed in an existing system, they should, in addition to the above criteria, also have the following properties:

1. *Simplicity:* the behavior should be implemented as simply as possible,
2. *Locality:* within our framework, the behavior should be generated by local rules, utilizing locally available sensory information,
3. *Correctness:* within the model in which it is tested, the behavior should provably attain (and in some cases maintain) the goal for which it was intended within the set of conditions for which it is designed,
4. *Stability:* the behavior should not be sensitive to perturbations in external conditions for which it is designed,

5. *Repeatability:* the behavior should perform according to specification in each trial under reasonable conditions and error margins,

6. *Robustness:* the performance of the behavior should not degrade significantly in the presence of specified bounds of sensory and effector error and noise,

7. *Scalability:* the behavior should scale well with increased and decreased group size.

It is difficult to imagine any fixed metric for selecting an "optimal" set of behaviors, since the choice of the basic behavior set depends on the task(s) it will be applied to. This work makes no attempt to devise optimality criteria in any formal sense. Furthermore, this work does not provide theoretical proofs of correctness of the algorithms for the presented behaviors. While such proofs may be computable for a simple model of the agents and the environment, they become prohibitively difficult for increasingly more realistic models that include sensors, effectors, and dynamics. As an alternative to simplified modeled environments, the behaviors were tested in the fully complex worlds with all of the error, noise, and uncertainly. In order to make the evaluation more complete, various initial conditions and group sizes were tested, and a large amount of data were obtained for analysis. Behavior evaluation is described in detail in section 4.5.

The next section illustrates the process of selecting basic behaviors for the domain of planar mobile agents.

4.1.2 Basic Behaviors for Movement in the Plane

The experimental work in this thesis is focused on interactions among mobile agents in two–dimensional space. This domain has the desired complexity properties: the number of possible collective behaviors is unbounded. Fortunately, the unbounded space of possible spatial and temporal patterns can be classified into classes, and thus effectively viewed from a lower level of resolution. The classification is based on task and domain–specific criteria which allow for selecting out the (comparatively) few relevant behavior classes to focus on. The proposed basic behaviors impose such classes; they define observable group behaviors without specifying particular rules for implementing them.

Group behaviors in the spatial domain can be viewed as spatio–temporal patterns of agents' activity. Certain purely spatial fixed organizations of agents are relevant, as are certain spatio–temporal patterns. Purely spatial fixed organizations of agents correspond to goals of attainment while spatio–temporal patterns correspond to goals of maintenance.

Safe–Wandering	the ability of a group of agents to move around while avoiding collisions with each other and other obstacles. Here, the homogeneous nature of the agents can be used for inter–agent collision avoidance. Thus, two distinct strategies can be devised; one for avoiding collisions with other agents of the same kind, and another for avoiding collisions everything else.
Following	the ability of two or more agents to move while staying one behind the other.
Dispersion	the ability of a group of agents to spread out over an area in order to establish and maintain some predetermined minimum separation.
Aggregation	the ability of a group of agents to gather in order to establish and maintain some predetermined maximum separation.
Homing	the ability to reach a goal region or location.

Table 4.2: A basic behavior set for the spatial domain, intended to cover a variety of spatial interactions and tasks for a group of mobile agents.

In the process of selecting basic behaviors, the designer attempts to decide what behavior set will suffice for a large repertoire of goals. While the dynamical properties of the system provide bottom–up constraints, the goals provide top–down structure. Both of these influences guide the behavior selection process. Energy minimization is a universal goal of powered physical systems. In the planar motion domain this goal translates into minimization of non–goal–driven motion. Such motion is either generated by poor behavior design, or by interference between agents. Thus, minimizing interference means maximizing goal–driven behavior and minimizing unnecessary motion.

Minimizing interference translates directly into the achievement goal of immediate avoidance and the maintenance goal of moving about without collisions. Avoidance in groups can be achieved by dispersion, a behavior that reduces interference locally. It can also serve to minimize interference in classes of tasks that require even space coverage, such as those involving searching and exploration.

In contrast to various goals that minimize interaction by decreasing physical proximity, many goals involve the exchange of resources through physical proximity. Consequently, aggregation is a useful primitive. Moving in a group requires some form of coordinated motion in order to minimize interference. Following and flocking are examples of such structured group motion.

Table 4.2 shows a list of behaviors that constitutes a basic set for a flexible repertoire of spatial group interactions. Biology offers numerous justifications for these behaviors. Avoidance and wandering are survival instincts so ubiquitous it obviates discussion. Following, often innate, is seen in numerous species (McFarland 1985). Dispersion is commonplace as well. DeScnutter & Nuyts (1993) show elegant evidence of gulls aggregating by dynamically rearranging their positions in a field to maintain a fixed distance from each other. Camazine (1993) demonstrates similar gull behavior on a ledge. People maintain similar arrangements in enclosed spaces (Gleitman 1981). Similarly, Floreano (1993) demonstrates that simulated evolved ants use dispersion consistently. Aggregation, as a protective and resource–pooling and sharing behavior, is found in species ranging from the slime mold (Kessin & Campagne 1992) to social animals (McFarland 1987). The combination of *dispersion* and *aggregation* is an effective tool for regulating density. Density regulation is a ubiquitous and generically useful behavior. For instance, army ants regulate the temperature of their bivouac by aggregating and dispersing according to the local temperature gradient (Franks 1989). Temperature regulation is just one of the many side–effects of density regulation. Finally, homing is a basis of all navigation and is manifested by all mobile species (for biological data on pigeons, bees, rats, ants, salmon, and many others see Gould (1987), Muller & Wehner (1988), Waterman (1989), Foster, Castro & McNaughton (1989), and Matarić (1990*b*)).

Besides the described behavior set, numerous other useful group behaviors exist. For example, biology also suggest surrounding and herding as frequent patterns of group movement, related to a higher level achievement goal, such as capture or migration (McFarland 1987). These and other behaviors can be generated by combining the basic primitives, as will be described and demonstrated in the next chapter.

4.2 Basic Behavior Experiments

The remainder of this chapter describes the experimental environments, presents the algorithms for implementing the proposed basic behaviors, and evaluates their performance based on a battery of tests and a collection of criteria.

4.2.1 Experimental Environments

Behavior observation is one of the primary methods for validating theories in synthetic AI projects like the one described in this thesis. In order to have conclusive results, it is necessary to try to separate the effects caused by the particular experimental environment from those intrinsic to the theory being tested. In order to get to the heart of group behavior issues rather than the specific dynamics of the test environment, two different environments were used, and the results from the two were compared. The two environments are the Interaction Modeler, and a collection of physical robots.

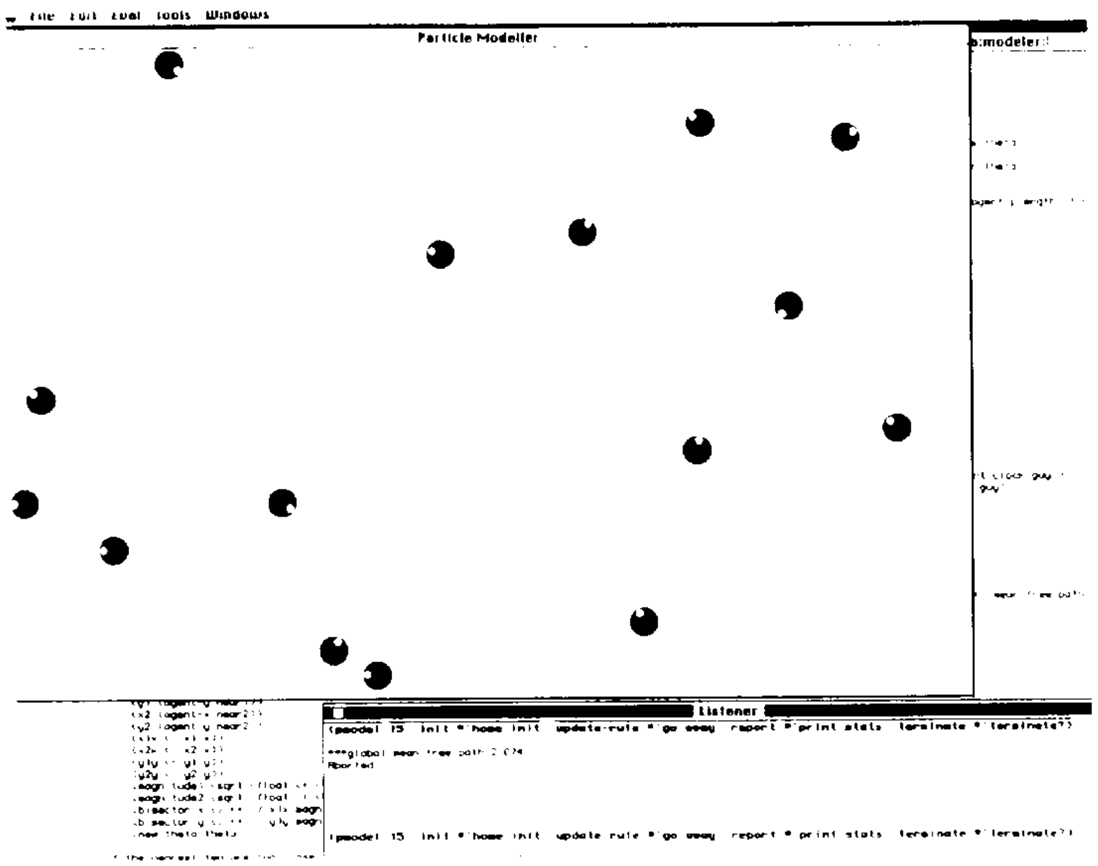
Another motivation for using both a physical and a modeled environment is the attempt to isolate any observable inconsistencies in the performance of the same behaviors in the two different environments. In general, it is difficult to determine what features of the real world must be retained in a simulation and what can be abstracted away. By testing systems in the physical world some of the effects that arise as artifacts of simulation can be identified (Brooks 1991*a*). This is the motivation behind using data from physical robots. By the same token, the current state of the art of physical robot environments imposes many constraints and biases on the types of experiments that can be conducted. Consequently, results from any physical environment must also be validated in an alternative setup. Two different robot types were used, in order to eliminate system–specific biases.

Since this work is concerned with basic principles of interaction and group behavior rather than a specific domain, it is especially concerned with effects that are common to both the modeled and the physical worlds.

4.2.2 The Agent Interaction Modeler

The Interaction Modeler (IM) is a simulator which allows for modeling a simplified version of the physics of the world and the agent sensors and dynamics (Figure 4-1). The Modeler and the control software for the agents are written in Lisp. However, for purposes of realism, the modeler is divided into three distinct components: the simulator, the physics modeler, and the agent specification. The simulator executes the agent specifications and moves the agents according to their control algorithms and their sensory readings. The simulator implements the physics of the sensors, but not the physics of the world. The latter are implemented by the physics modeler that checks the positions and motions computed by the simulator against simplified physical laws, and applies corrections. The IM loops between the simulator and the physics modeler.

The main purpose of the Interaction Modeler is to observe and compare phenomena to those obtained on physical robots. However, the Modeler is also useful for preliminary testing of group behaviors which are then implemented on physical robots. Although it is difficult to directly transfer control strategies from simulations



```
(pmodel 15 init #'home init update-rule #'go away report #'print stats terminate #'terminate?)

***global mean free path 2.074
Aborted
```

Figure 4-1: The interaction modeler environment. The agents are shown as black circles with white markers indicating their heading. The large rectangle indicates the boundaries of their workspace. The agents are equipped with local sensors and simplified dynamics.

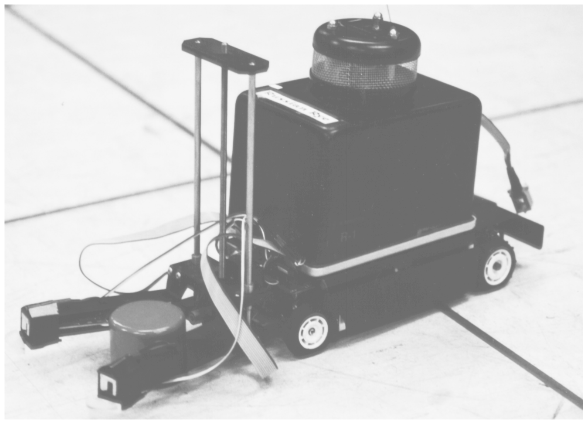


Figure 4-2: Each of the Nerd Herd robots is a 12"–long four–wheeled base equipped with a two–pronged forklift for picking up, carrying, and stacking pucks, and with a radio transmitter and receiver for inter–robot communication and data collection.

to the physical world, the modeler is useful for eliminating infeasible control strategies at an early stage, as well as for testing vastly larger numbers of agents, performing many more experiments, and varying parameter values.

4.2.3 The Mobile Robot Herd

Group behavior experiments are implemented and tested on a collection of 20 physically identical mobile robots affectionately dubbed "The Nerd Herd." Each robot is a 12"–long four–wheeled vehicle, equipped with one piezo–electric bump sensor on each side and two on the rear of the chassis. Each robot has a two–pronged forklift for picking up, carrying, and stacking pucks (Figure 4-2). The forklift contains two contact switches, one on each tip of the fork, six infra–red sensors: two pointing forward and used for detecting objects and aligning onto pucks, two break–beam sensors for detecting a puck within the "jaw" and "throat" of the forklift, and two down–pointing sensors for aligning the fork over a stack of pucks and stacking (Figure 4-3). The pucks are special–purpose light ferrous metal foam-filled disks, 1.5 inches diameter and between 1.5 and 2.0 inches in height. They are sized to fit into the unactuated fork and be held by the fork magnet.

The robots are equipped with radio transceivers for broadcasting up to one byte of data per robot per second. The system uses two radio base stations to triangulate

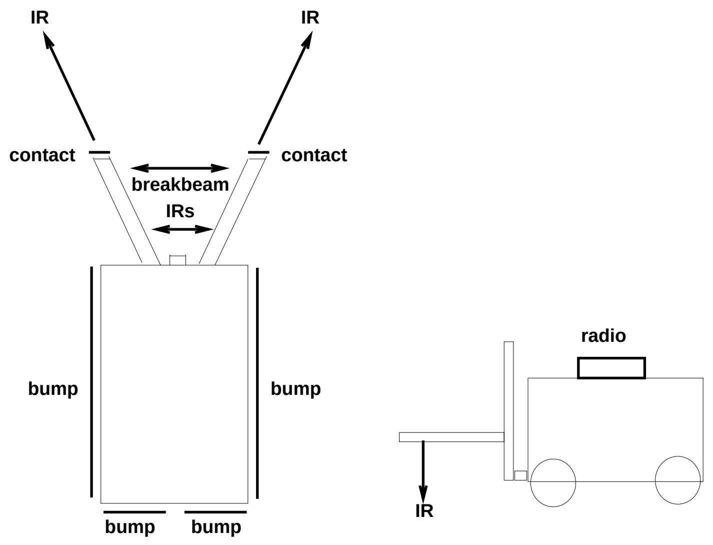


Figure 4-3: Each of the Nerd Herd robots is equipped with contact sensors at the ends of the fork, piezo–electric bump sensors on each side and two on the rear of the chassis, and six infra–red sensors on the fork. Two forward–pointing IRs are located at the ends of the forks, two break–beam IRs in the jaw and throat of the fork, and two down–pointing IR for stacking pucks in the middle of each of the fork arms.

the robots' positions. The radio system is used for data gathering and for simulating additional sensors. In particular, radios are used to distinguish robots from other objects in the environment, an ability that cannot be implemented with the on–board IR sensors[1].

The mechanical, communication, and sensory capabilities of the robots allow for exploration of the environment, robot detection, and finding, picking up, and carrying pucks. These basic abilities are used to construct various experiments in which the robots are run autonomously, with all of the processing and power on board. The processing is performed by a collection of four Motorola 68HC11 microprocessors. Two of the processors are dedicated to handling radio communication, one is used by the operating system, and one is used as the "brain" of the robot, for executing the down–loaded control system used in the experiments. The control systems are programmed in the Behavior Language, a parallel programming language based on the Subsumption Architecture (Brooks 1990*a*).

[1]The IRs are all the same frequency and mechanically positioned for obstacle detection rather than communication.

4.2.4 Hardware Limitations

Properties of physical hardware impose restrictions not only on the control strategies that can be applied, but alson on the types of tasks and experiments that can be implemented. Robot hardware is constrainted by various sensory, mechanical, and computational limitations. This section describes some relevant properties of the hardware we used and their effect.

The robots' mechanical steering system is inaccurate to within 30 rotational degrees. Furthermore, the position triangulation system works sufficiently well when the robots are within the predetermined range of the base stations. However, the exchange of information between the robots, which nominally ought to take place at 1Hz, suffers from extensive loss of data. Consequently, as much as half of the transmitted data were lost or incorrect. The combined effect of steering and positioning uncertainty demanded that the robots move slowly in order to minimize error. Thus, the limiting factor on the robot speed was imposed by sensing and actuation, not by the controller.

The infra–red sensors have a relatively long range (12 inches), and vary in sensitivity. Consequently, not only do different robots have different sensing ranges that cannot be tuned due to hardware restrictions, but the sensitivity between the two sides of the fork on a single robot varies as well. Consequently, the amount of time and effort required for detecting, picking up, or avoiding objects varied across robots and over time. Thus, the control system could not be dependent on uniformity of the group.

This uncertainty and variability, although frustrating, is beneficial to experimental validity. For instance, hardware variability between robots is reflected in their group behavior. Even when programmed with identical software, the robots behave differently due to their varied sensory and actuator properties. Small differences among individuals become amplified as many robots interact over extended time. As in nature, individual variability creates a demand for more robust and adaptive behavior. The variance in mechanics and the resulting behavior provides a stringent test for all of the experimental behaviors.

4.2.5 Experimental Procedure

All robot and modeler programs were archived and all basic behaviors were tested in both domains. All robot implementations of basic and composite behaviors were tested in at least 20 trials[2]. In the case of the Modeler, all behaviors were tested in at least 20 trials, with both identical and random initial conditions. Different strategies for the same group behaviors were tested and compared across the two domains.

Modeler data were gathered by keeping a record of relevant state (such as position, orientation, and gripper state) over time. The same data were gathered in robot experiments through the use of the radio system. The system allowed for recording the robots' position and a few bytes of state over time. For each robot experiment, the robots' IDs and initial positions were recorded. Some of the experiments were conducted with random initial conditions (i.e., random robot positions), while in others identical initial positions were used in order to measure the repeatability of the behaviors. All robot data were also recorded on video tape, for validation and cross referencing.

Throughout this chapter, the Interaction Modeler data are shown in the form of discrete snapshots of the global state of the system at relevant times, including initial state and converged state. The robot data are plotted with the Real Time Viewer (RTV), a special purpose software package designed for recording and analyzing robot data[3]. RTV uses the transmitted radio data to plot, in real–time, the positions of the robots and a time–history of their movements, i.e. a trail, the positions of the previously manipulated pucks, and the position of home. It also allows for replaying the data and thus recreating the robot runs.

The robots are shown as black rectangles aligned in the direction of their heading, with their ID numbers in the back, and white arrows indicating the front. In some experiments robot state is also indicated with a symbol or a bounding box. In all shown data plots, the size of the rectangles representing the robots is scaled so as to maintain the correct ratio of the robot/environment surface area, in order to demonstrate the relative proximity of all active robots. The bottom of each plot shows which of the twenty robots are being run. The corner display shows elapsed time, in seconds, for each snapshot of the experiment. Figure 4-4 shows a typical data plot.

4.3 Basic Behavior Specifications

This section gives formal specifications for each behavior in terms of the goal it achieves and maintains.

Basic behaviors in 2D space are specified in terms of positions p, distances d, and

[2]In the case of foraging, most data were obtained with another set of robots, described in section 8.1.

[3]RTV was implemented and maintained by Matthew Marjanović.

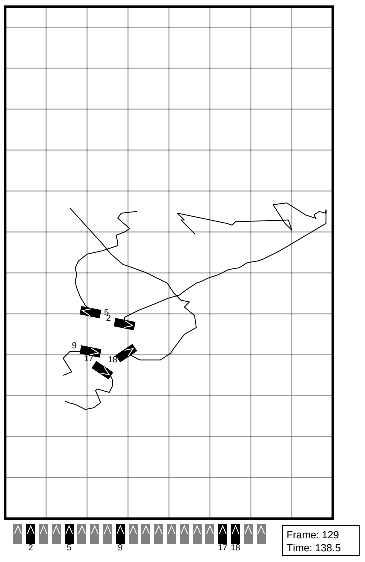


Figure 4-4: An example of a robot data plot: the robots are shown as scaled black rectangles aligned in the direction of their heading, with their ID numbers in the back, and white arrows indicating the front. The bottom of the plot shows which of the twenty robots are being run, and the corner display shows elapsed time in seconds.

distance thresholds δ_{avoid}, $\delta_{disperse}$, and $\delta_{aggregate}$.

$\mathcal{R}$ is the set of robots: $\mathcal{R} = \{R_i\}, \;\; 1 \leq i \leq n$

$$p_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \qquad p_{home} = \begin{pmatrix} x_{home} \\ y_{home} \end{pmatrix}$$

$$d_{home,i} = \sqrt{(x_{home} - x_i)^2 + (y_{home} - y_i)^2}$$

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Using this notation, the following are specifications for the basic behavior goals.

Safe–Wandering:

The goal of safe–wandering is to keep moving while maintaining a minimum distance δ_{avoid} between agents:

$$\frac{dp_j}{dt} \neq 0 \;\; and \;\; \forall(i) \;\; d_{i,j} > \delta_{avoid}$$

Following:

The goal of following is to achieve and maintain a minimum angle θ between the position of the leader i relative to the follower j:

$$i = leader, \;\; j = follower$$

$$0 \;\; \leq \;\; \frac{dp_j}{dt} \cdot (p_i - p_j) \;\; \leq \;\; \| \frac{dp_j}{dt} \| \| (p_i - p_j) \| \cos\theta$$

$$\theta = 0 \;\; \Rightarrow \;\; \cos\theta = 0 \;\; \Rightarrow$$

$$0 \;\; \leq \;\; \frac{dp_j}{dt} \cdot (p_i - p_j) \;\; \leq \;\; \| \frac{dp_j}{dt} \| \| (p_i - p_j) \|$$

Dispersion:

The goal of dispersion is to achieve and maintain a minimum distance $\delta_{disperse}$ between agents:

$$\forall(j) \;\; d_{i,j} > \delta_{disperse} \;\; and \;\; \delta_{disperse} > \delta_{avoid}$$

Aggregation:

The goal of aggregation is to achieve and maintain a maximum distance $\delta_{aggregate}$ between agents:

$$\forall(j) \;\; d_{i,j} < \delta_{aggregate}$$

Homing:

The goal of homing is to decrease the distance between the agent and a goal location called "home":

$$\forall j \ \frac{dp_j}{dt} \cdot (p_j - p_{home}) < 0$$

4.4 Basic Behavior Algorithms

This section presents the algorithms used to implement each of the proposed basic behaviors in the Interaction Modeler and on the robots. The algorithms are given in formal notation and in algorithmic pseudo code. All algorithms are formally expressed as velocity commands of the form:

$$command\,(v)$$

Two operators, $\mathcal{N}$ and $\mathcal{C}$, are used for computing most of the algorithms. $\mathcal{N}$ is the **neighborhood operator** which, given a robot R and a distance threshold δ, returns all other robots within that neighborhood:

$$\mathcal{N}(i, \delta) = \{j \in i, ..n \ | \ d_{i,j} \leq \delta\}$$

C is the **centroid operator** which, given a robot i and a distance threshold δ, returns the local centroid:

$$C(i, \delta) = \frac{\sum_{j \in \mathcal{N}(i,\delta)} p_j}{|\mathcal{N}(i, \delta)|}$$

C_g is the global centroid operator:

$$C_g = \frac{\sum_{j \in \mathcal{R}} p_j}{|n|}$$

4.4.1 Safe–Wandering

Strategies for moving while avoiding collisions are perhaps the most studied topic in mobile robotics. The work in this thesis was concerned with finding avoidance strategies that perform well in group situations and scale well with increased group

```
Avoid-Other-Agents:
If an agent is within d_avoid
   If the nearest agent is on the left
      turn right
      otherwise turn left.
```

Algorithm 4.1:

```
Avoid-Everything-Else:
If an obstacle is within d_avoid
   If an obstacle is on the right only, turn left.

   If an obstacle is on the left only, turn right.
   After 3 consecutive identical turns, backup and turn.

   If an obstacle is on both sides, stop and wait.
   If an obstacle persists on both sides,
      turn randomly and back up.
```

Algorithm 4.2:

sizes. Finding a guaranteed general–purpose collision avoidance strategy for an agent situated in a dynamic world is difficult. In a multi–agent world the problem can become intractable.

Inspired by biological evidence which indicates that insects and animals do not have precise avoidance routines (Wehner 1987), we used the following general avoidance behavior:

$$command\ (\ v \begin{pmatrix} cos(\theta + u) \\ sin(\theta + u) \end{pmatrix}\ )$$

where θ is R's orientation and u is the incremental turning angle away from the obstacle. A simple `Avoid-Other-Agents` rule was devised, as shown in Algorithm 4.1.

The `Avoid-Other-Agents` behavior takes advantage of group homogeneity. Since all agents execute the same strategy, the behavior can rely on and take advantage of the resulting spatial symmetry. If an agent fails to recognize another with its other–agent sensors (in this case radios), it will subsequently detect it with its collision–avoidance sensors (in this case IRs), and treat it as a generic obstacle, using the

```
Safe--Wander:
If an agent is within d_avoid
   If the nearest agent is on the left
      turn right
      otherwise turn left.

If an obstacle is within d_avoid
   If an obstacle is on the right only, turn left.

   If an obstacle is on the left only, turn right.
   After 3 consecutive identical turns, backup and turn.

   If an obstacle is on both sides, stop and wait.
   If an obstacle persists on both sides,
      turn randomly and back up.

Otherwise move forward by d_forward, turn randomly.
```

Algorithm 4.3:

`Avoid-Everything-Else` behavior, as shown in Algorithm 4.2.

A provably correct avoidance strategy for arbitrary configurations of multiple agents is difficult to devise. In order to increase robustness and minimize oscillations, our strategies take advantage of the unavoidable noise and errors in sensing and actuation, which result in naturally stochastic behavior. This stochastic component guarantees that the an avoiding agent will not get stuck in infinite cycles and oscillations. In addition to the implicit stochastic nature of the robots' behavior, `Avoid-Everything-Else` also utilizes an explicit probabilistic strategy by employing a randomized move.

Variations of the above avoidance algorithm were experimented with and compared based on the amount of time the agent spent avoiding relative to the amount of time spent it moving about freely. This ratio is an indirect measure of the quality of the avoiding strategy in that the more time the agents spend avoiding the worse the strategy is. Avoiding time is dependent on the agent density, so it was used as a controlled variable in the experiments. The ratio used to evaluate avoidance is an indirect metric; a direct measure of being stuck would be more useful, but the robots did not have the appropriate sensors for determining this state. No significant performance differences were found among the similar strategies that were tested.

The strategy for *safe–wandering* is the combination of the two avoidance strategies with a default rule for moving with occassional changes of heading, as shown in Algorithm 4.3.

4.4.2 Following

```
Follow:
If an agent is within d_follow
   If an agent is on the right only, turn right.

   If an agent is on the left only, turn left.
```

Algorithm 4.4:

Following is implemented with respect to the follower agent. It is achieved with a simple rule that steers the follower to the position of the leader:

$$\mathit{command}\ (\ \frac{v_0}{\|\ p_{leader} - p_{follower}\ \|}\ (p_{leader} - p_{follower})\ )$$

Following can be implemented as a complement of the `Avoid-Everything-Else` behavior, as shown in Algorithm 4.4.

Figure 4-5 illustrates following on three robots. Additional data on following will be presented and analyzed in the next section.

This approach to *following* models tropotaxic behavior in biology, in which two sensory organs are stimulated and the difference between the stimuli determines the motion of the insect (McFarland 1987). Ant osmotropotaxis is based on the differential in pheromone intensity perceived by the left and right antennae (Calenbuhr & Deneubourg 1992), while the agents described here use the binary state of the two directional IR sensors.

Under conditions of sufficient density, safe–wandering and following can produce more complex global behaviors. For instance, osmotropotaxic behavior of ants exhibits emergence of unidirectional lanes, i.e., regions in which all ants move in the same direction. The same lane–forming effect could be demonstrated with robots executing following and avoiding behaviors. However, more complex sensors must be used in order to determine which direction to follow. If using only IRs, the agents cannot distinguish between other agents heading toward and away from them, and are thus unable to select whom to follow.

algorithm, when tested on the Interaction Modeler, produces more direct homing trajectories. Figure 4-9 shows another robot run of homing with five robots. In this run the entire time history of the robots' positions are shown, and the positioning errors can be easily seen. Nonetheless, all robots reach home. Figure 4-10 illustrates homing in simulation.

Individual homing is effective as long as the density of agents is low. If enough agents are homing within confined space, they interfere with each other. In the case of our non-holonomic robots, interference had even more enduring effects on the group. Figure 4-11 shows the growing interference between robots as they approach the goal region. Entire time–trails are shown to demonstrate how much group interference slows down individual performance.

Simulation and robot experiments described in this work show that interference increases if the agents have non–zero turning radii, unequal velocities, or are subject to sensor and control errors. All of the above conditions are common in situated agents, suggesting the need for some form of group or structured navigation, such as flocking, which will be introduced in an upcoming section.

4.5 Basic Behavior Evaluation

4.5.1 Empirical Evaluation of Basic Behaviors

Evaluation is one of the most difficult components of research, and it is somewhat new to the field of AI and Experimental Robotics. By nature and by design, the two fields are based on building artificial computational and physical systems. However, results from such synthetic endeavors do not fall cleanly into the well defined set of evaluation criteria designed for natural sciences. Analyzing something one has designed is intrinsically different from analyzing something externally imposed.

As a young and diverse field, AI still lacks standardized evaluation criteria. Consequently, it is left to each researcher to establish criteria that are both specific to the project and generally acceptable. The ideas proposed in this thesis are evaluated in two ways. The first addresses the merit of the general approach and its applicability to various domains. This evaluation is performed in the summary of the thesis, after the entire work has been presented. The second type of evaluation addresses the specific instantiation of the ideas in the spatial domain. This chapter presents the evaluation criteria applied to the implementations and performance of spatial basic behaviors and their composites.

AI and Robotics research in general is exploratory and often prone to phenomenological evaluation. To prevent this, all of the evaluation criteria for the experimental part of the work were established prior to testing and were applied to the performance of each of the behaviors as well as to their combinations. An earlier section on basic behavior selection elaborated the criteria for choosing the basic behavior set and hinted at some evaluation procedures. This section gives a detailed illustration of empirical basic behavior evaluation on the example of *following*.

According to our pre-specified definition, a robot was said to be *following* when it maintained a minimal angle θ between itself and the leader. Repeatability and robustness of *following* were evaluated based on its manifested average uninterrupted duration, i.e. average time to failure. This duration was almost completely dependent on how reliably the front–pointing sensors could detect the "leader". Figure 4-12 illustrates continuous *following* behavior of 3 robots over a four minute period. The robot at the "front" of the queue is moving forward with its wheels slightly turned, thus tracing out a circular path. The other two robots follow their local "leader" according to the presented algorithm. The path of the first robot is smooth, while the followers oscillate in order to keep the robot ahead of them within IR range. One of the robots separated after two minutes, while the other two stayed together for the duration of the shown 243.3 second run. Figure 4-13 also illustrates the robustness of *following*; the robot in the lead moves about randomly and the follower keeps up throughout the duration of the run.

The range of the IR sensors used was directed and short, requiring the agents to stay close together within the queue. Consequently, errors in steering could cause a follower to lose sight of the leader if it failed to turn sufficiently in order to maintain the leader in sight. If the two continued to move in the same direction, as they would during a higher–level task, the follower could catch up with the leader again. If not, they would separate.

The narrow IR range explains why long queues and trains of agents were physically difficult to maintain. However, queues were stable and insensitive to dynamic obstacles and sensory and mechanical irregularities in the form of sensor noise, errors in steering, and perturbations in velocity. Figure 4-14 illustrates *following* on three robots in the presence of sensory or effector error. The middle robot stalls due to some error, and the robot behind it stops as well, then turns and follows the leader, as it senses the first robot in its range. The middle robot activates again, senses the second robot within its range, follows it, and the queue is maintained. Figure 4-15 demonstrates *following* in the presence of static constraints in the environment, such as walls and corners. The robots are able to avoid the walls and maintain the queue.

Following was also evaluated based on scalability in order to test its performance as agents are added and removed. The data above demonstrate the behavior that results if an agent stalls, or is removed from the middle of the queue. The next set of data deals with the performance as new agents are added to the queue, the situation that is expected to happen more commonly, since *following* is, at a global level, a recruiting behavior.

Figure 4-16 demonstrates average following time for two robots in multiple runs. Figure 4-17 plots *following* data for three robots. The mean following time for two agents is nearly identical as that for three. This is exactly as expected, since *following* is a completely local behavior between two agents. The failure of any pair is as likely as the failure of any other, and the pairs are mutually independent, soq agents can be dynamically added and removed from the ends of the queue without affecting the rest.

This section has illustrated the criteria we used to evaluate the proposed basic behaviors. The evaluation process was illustrated on the example of *following*. The described criteria were systematically applied to all of the other basic behaviors as well.

4.5.2 Evaluation of Heterogeneous Groups

An obvious alternative for a fully distributed system of identical agents is a hierarchical distributed system. In order to evaluate the performance of the homogeneous basic behaviors, they were compared to particular hierarchical implementations. This section describes the performance of a hierarchical group of agents on two basic behaviors: aggregation and dispersion. These two behaviors were chosen because they can be stated in terms of achievement goals and, given sufficient space, can reach a static state. The algorithms were evaluated based on the time or the number of steps required to reach that well–defined state.

A version of hierarchical agents was implemented by classifying the agents into a total order, based on a randomly assigned unique ID number, thus simulating an established pecking order in the group (Chase, Bartolomeo & Dugatkin 1994, Chase 1993, Chase 1982, Chase & Rohwer 1987). While in homogeneous algorithms all agents moved simultaneously according to identical local rules, in the hierarchical case the ID number determined which agents were allowed to move while others waited. In all cases, a simple precedence order, a spatially–local hierarchy, was established such that within a small radius the agent with the highest ID got to move. Multiple types of dispersion and aggregation algorithms were tested with such hierarchical agents.

Using the Interaction Monitor, 20 experiments were conducted with each group size (3, 5, 10, 15, and 20 agents) and each of the algorithms. Additionally, the algorithms were tested on two different degrees of task difficulty. Aggregation was tested

4.6 Summary

This chapter has introduced the methodology for selecting basic behaviors and demonstrated it on the spatial domain. A basic behavior set consisting of *safe–wandering, following, dispersion, aggregation,* and *homing* was proposed, implemented in two different experimental environments, and tested in simulation and on physical robots. Experimental data were evaluated using a collection of criteria we specified *a priori.* The performance of the basic behaviors was also tested compared against hierarchical and total knowledge approaches.

The next chapter introduces ways in which the described basic behaviors can combined in order to be achieve a variety of higher–level goals and tasks.

Chapter 6

Learning in Situated Systems

So far we have dealt with the problem of synthesizing intelligent group behavior by hand. We now extend the presented ideas to include learning, an ability that allows the agent to acquire new and adapt old behaviors for individual and group benefit.

6.1 Motivation

Why learn?
Learning has two purposes universal across domains. It is useful for:

1. adapting to external and internal changes
2. simplifying built–in knowledge

The ability to cope with changes in the environment is termed *adaptability.* It allows agents to deal with noise in their internal and external sensors, and with inconsistencies in the behavior of the environment and other agents. Adaptability comes at a phenotypical and cognitive cost, so creatures are adapted only to a specific niche. Consequently, all creatures, natural and otherwise, fail at their tasks under certain conditions. The purpose of learning is to make the set of such conditions smaller.

Adaptability does not necessitate learning. Many species are genetically equipped with elaborate "knowledge" and abilities, from the very specific, such as the ability to record and utilize celestial maps (Waterman 1989), to the very general, such as plasticity in learning motor control (McFarland 1987) and language (Pinker 1994). But genetic code is finite. In fact, primate and human cortical neural topology is too complicated to fully specify in the available genome, and is instead established by

Problem	Learning in complex situated domains.
Assertion	Traditional reinforcement learning must be reformulated.
Approach	Replace states, actions and reinforcement with conditions, behaviors, heterogeneous reward functions and progress estimators.
Validation	Implement learning on a group of mobile robots learning to forage.

Table 6.1: A summary of the situated learning problem addressed here, and the structure of the proposed solution.

spontaneous synaptic firing *in utero* and in the first decade of life (Vander et al. 1980). In addition to compensating for genetic parsimony, learning is useful for optimizing the agent's existing abilities, and necessary for coping with complex and changeable worlds. It is often argued that societies exist largely for conservation and propagation of behavior strategies too complex to be passed on genetically.

The answer to the built–in versus learned tradeoff varies across species and environments. The work described here addresses this fundamental tradeoff in the domain of situated multi–agent systems.

The rest of the thesis will address the following problem: how can a collection of situated agents learn in a group environment? This problem will be addressed in a nondeterministic, noisy and error–prone domain with stochastic dynamics, in which the agent does not have an *a priori* model of the world.

We propose a formulation of reinforcement learning that uses a level of description that makes the state space manageable, thus making learning possible. Furthermore, we present two methods for shaping reinforcement to take advantage of information readily available to the agent, and to make learning more efficient. These ideas are validated by demonstrating an effective learning algorithm on a group of robots learning to forage. Table 6.1 summarizes the problem and the approach.

6.2 Relevant Learning Models

There are many things an agent can learn, but not many ways in which it can learn it. According to what is being learned, existing approaches can be classified into the following categories:

- learning declarative knowledge
- learning control
- learning new behaviors
- learning to select behaviors/actions

6.2.1 Learning Declarative Knowledge

Learning declarative knowledge is one of the founding areas of AI but also one that is least directly related to the work in this thesis. The only type of declarative knowledge that situated agents have had to deal with to date are maps of the environment. Much of the robotics literature deals with the problem of constructing and updating such maps in variety of situated domains (see Matarić (1990*a*) for a review of the literature). Maps and world models are closely tied to action in the world, which is why they are the primary type of declarative knowledge so far used in situated agents[1]. In contrast, this thesis focuses on procedural knowledge that is directly tied to acting and interacting in the world. The remaining learning categories are directly tied to action[2].

6.2.2 Learning Control

Learning control is a growing field based on *adaptive control*, a branch of control theory. Problems in adaptive control deal with learning the forward or inverse model of the system, i.e., the plant. Forward models provide predictions about the output expected after performing an action in a given state. Analogously, inverse models provide an action, given the current state and a desired output (Jordan & Rumelhart 1992). Learning control has been applied to a variety of domains and has used a number of different learning methodologies. Connectionist algorithms are most popular, (see Miller, Sutton & Werbos (1990) for a representative collection), but

[1]Note: not all maps are explicit and declarative. See Matarić (1990*a*) for examples.

[2]Author's bias: declarative learning can be further divided into as many interesting categories, but is not the area pursued here.

other approaches have also been studied (e.g., Atkeson, Aboaf, McIntyre & Reinkensmeyer (1988), Atkeson (1990), Schaal & Atkeson (1994)). Adaptive control problems typically deal with learning complex dynamical systems with non–linearly coupled degrees of freedom usually involved in moving multi–jointed manipulators, objects, and physical bodies.

6.2.3 Learning New Behaviors

Learning new behaviors deals with the problem of acquiring strategies for achieving particular goals. Because the notion of behavior is not well defined, neither is the behavior learning problem.

We defined behavior to be a control law with a particular goal, such as *wall–following* or *collision avoidance*. The definition is general and meant to refer to a level of description above basic control without specifying what that level is, since it varies with the domain. Furthermore, the concept of behavior contains informal notions about generality and adaptivity that are difficult to state precisely without domain–specific grounding.

Consequently, most learning control problems appear to be instances of behavior learning, such as learning to balance a pole (Barto, Sutton & Anderson 1983), to play billiards (Moore 1992), and to juggle (Schaal & Atkeson 1994). Furthermore, work on action selection, deciding what action to make in each state, can be viewed as learning a higher–level behavior as an abstraction on the state–action space. For example, a maze–learning system can be said to learn a specific *maze–solving* behavior.

Genetic learning has also addressed learning behaviors in simulated worlds (Koza 1990). Since learning behaviors requires finding appropriate parameter settings for control, it can be cast as an optimization problem, for which genetic algorithms are particularly well suited (Goldberg 1989). However, since genetic algorithms operate on an abstract encoding of the learning problem, the encoding requires a good model of the agent and the environment in order to generate useful behaviors. Since the problem of modeling situated worlds is notoriously difficult, only a few genetic algorithms have produced behaviors that successfully transferred to physical systems (Steels 1994*b*, Cliff, Husbands & Harvey 1993, Gallagher & Beer 1993).

However, none of the above learning approaches can be said to learn new behaviors according to the precise definition of the problem. The posed "behavior learning problem" (Brooks & Matarić 1993) requires that the agent acquire a new behavior using its own perceptual and effector systems, as well as to assign some semantic label to the behavior, in order to later recognize and use it as a coherent and independent unit. Behavior learning appears to require bridging the elusive signal–to–symbol gap, even for the most limited notion of "symbol."

Given this definition, no existing work performs behavior learning. Learning control and learning action selection are not strictly instances of behavior learning because in both cases, by definition, only a single behavior is learned and no further abstraction is performed. Similarly, genetic algorithms do not address the stated behavior learning problem either, because in their domain the semantics are also provided by the designer.

The signal–to–symbol problem is one of the hallmark challenges in AI. Because it bridges a gap between two already estranged communities, it has not received much attention. Another challenge of the problem is setting it up to avoid biasing the learner inappropriately, but still be able to evaluate its performance. It is unlikely that "behaviors", "concepts", and "symbolic representations" that are automatically generated by a situated agent will map neatly from the agent's sensorium into the human observer's semantic space. Nonetheless, the situated domain is particularly well suited for this type of work as it allows for grounding the agents' learning in physical behavior that is observable and thus can be evaluated externally from its mechanism and representation.

6.2.4 Learning to Select Behaviors

If learning new behaviors is learning *how* to do something, then learning to select behaviors is learning *when* to do it. Behavior selection has not been extensively studied so far, largely due to the lack of formalization of "behavior" as a building block for control. The work that has been done on the topic has used reinforcement learning techniques (e.g., Maes & Brooks (1990) and Maes (1991)). Learning behavior selection is by definition a *reinforcement learning problem* as it is based on correlating the behaviors the agent performs and the feedback it receives as a result.

6.3 Reinforcement Learning

Reinforcement learning (RL) is a class of learning methodologies in which the agent learns based on external feedback received from the environment. The feedback is interpreted as positive or negative scalar reinforcement. The goal of the learning system is to maximize positive reinforcement (reward) and/or minimize negative reinforcement (punishment) over time. Traditionally, the learner is given no explicit built–in knowledge about the task. If the learner receives no direct instruction or answers from the environment the learning is considered *unsupervised* (Barto 1990). The learner produces a mapping of states to actions called a *policy*.

Reinforcement learning originated in Ivan Pavlov's classical conditioning experiments (Gleitman 1981). Embraced by behaviorism, stimulus–response learning became the predominant methodology for studying animal behavior in psychology and biology. Ethology, the study of animals in their natural habitats, developed in response to the tightly controlled laboratory experimental conditions commonly used by behaviorists. In the mean time, RL was adopted and adapted by the computational community, and applied to various machine learning problems.

Maze–learning was formulated as a reinforcement learning problem based on reward and punishment in the first well known application of RL (Minsky 1954). Soon thereafter, the problem of learning a scoring functions for playing checkers was successfully addressed with an RL algorithm (Samuel 1959). Subsequently, RL was applied to a variety of domains and problems, most notably in the Bucket Brigade algorithm used in Classifier Systems (Holland 1985), and in a class of learning methods based on Temporal Differencing (Sutton 1988). Reinforcement learning has been implemented with a variety of algorithms ranging from table–lookup to neural networks, and on a broad spectrum of applications, including tuning parameters and playing backgammon.

Our work is concerned with reinforcement learning on situated, embodied agents. In particular, it is focused on issues that arise when traditional models of RL, and algorithms applied to those models, are used in the complex multi–agent domain we are working with. To address these issues, we begin by describing the most commonly, but not exclusively, used RL model.

6.3.1 Markov Decision Process Models

Most computational models of reinforcement learning are based on the assumption that the agent–environment interaction can be modeled as a Markov Decision Process (MDP), as defined below:

1. The agent and the environment can be modeled as synchronized finite state automata.

2. The agent and the environment interact in discrete time intervals.

3. The agent can sense the state of the environment and use it to make actions.

4. After the agent acts, the environment makes a transition to a new state.

5. The agent receives a reward after performing an action.

While many interesting learning domains can be modeled as MDPs, situated agents learning in nondeterministic, uncertain environments do not fit this model. The next section describes the reasons why, by addressing each of the model assumptions in turn.

6.3.2 State

Most RL models are based on the assumption that the agent and the environment are always in a clearly–defined state that the agent can sense. In situated domains, however, the world is not readily prelabeled into appropriate states, and the world state is not readily and consistently accessible to the agent. Instead, the world is continuous and partially observable.

Continuity

The state of a situated agent consists of a collection of properties, some of which are discrete, such as the inputs from binary sensors, others continuous, like the velocities of wheels. Even for the simplest of agents, a monolithic descriptor of all state properties is prohibitively large. It scales poorly with increased sensory capabilities and agent complexity in general, and results in a combinatorial explosion in standard RL.

Most models to date have bypassed continuous state by presuming higher–level sensory operators such as "I see a chair in front of me." But such operators have been shown to be unrealistic and largely unimplementable in systems using physical sensors (Agre & Chapman 1990, Brooks & Matarić 1993). In general, the problem of partitioning continuous state into discrete states is hard (Košecká 1992), and even if a reasonable partitioning of the world is found, there may be no mapping from the space of sensor readings to this partitioning.

Observability

Although continuous and often complex, sensors have limited abilities. Instead of providing descriptions of the world, they return simple properties such as presence of and distance to objects within a fixed sensing region. Consequently, they cannot distinguish between all potentially relevant world states. The collapse of multiple states into one results in partial observability, i.e. in **perceptual aliasing**, a many–to–one mapping between world and internal states. The inability to distinguish different states makes it difficult and often impossible for the learning algorithm to assign appropriate utility to actions associated with such states (Whitehead & Ballard 1990).

Partially Observable Markov Decision Processes (POMDPs) have been developed by the operation research community for dealing with this problem. Partial observability is added into a Markov model by introducing a discrete probability distribution over a set of possible observations for a given state. POMDPs have been studied and successfully applied to theoretical learners (Cassandra, Kaelbling & Littman 1994), but have not yet been used empirically largely due to the fact that observability models of situated systems are not generally available.

Generalization

Any learner is caught in a paradox: it must disambiguate the relevant inputs, but it also must discard all irrelevant inputs in order to minimize its search space. However it may be structured, the learner's space in traditional RL is exponential in the size of the input, and thus marred by the curse of dimensionality (Bellman 1957). Some form of **input generalization**, or collapsing of states into functional equivalence classes, is necessary for almost all problems.

Human programmers perform generalization implicitly whenever they use clever orderings of rules, careful arbitration, and default conditions, in crafting control strategies. They minimize ambiguity and maximize parsimony by taking advantage of their domain knowledge.

In RL, in the absence of domain knowledge, state generalization has been addressed with statistical clustering methods using recursive partitioning of the state space based on individual bit relevance (Chapman & Kaelbling 1991, Mahadevan & Connell 1991*a*, Moore 1991, Moore 1993). It is also confronted in Classifier Systems that use binary strings as state descriptors (Holland 1986). The state can contain wild cards (#'s) that allow for clustering states, with the flexible grouping potential of full generality (all #'s) to full specificity (no-#'s). Generalization results in so-called "default hierarchies" based on the relevance of individual bits changed from #'s to specific values. This process is analogous to statistical RL methods (Matarić 1991).

The input generalization problem is also addressed by the connectionist RL literature. Multi–layer networks have been trained on a variety of learning problems in which the hidden layers constructed a generalized intermediate representation of the inputs (Hinton 1990). While all of the RL generalization techniques are non–semantic, the table–based methods and Classifier System approaches are somewhat more readable as their results are a direct consequence of explicit hand–coded criteria. Connectionist approaches, in contrast, utilize potentially complex network dynamics and produce effective but largely inscrutable generalizations.

All of the described generalization techniques are effective but require large numbers of trials to obtain sufficient statistical information for clustering states. As such, they are an incremental improvement of the overwhelmingly slow exponential learning algorithms. Our work will explore a different alternative, one that takes principled advantage of domain knowledge instead of purely statistical generalization.

Paradoxically, the unwieldy fully–exponential[3] state–action search space used by standard RL models gives them one of their main positive properties: asymptotic completeness. While hand coded reactive policies take advantage of the cleverness of the designer, they are rarely provably complete. Most irrelevant input states are easily eliminated, but potentially useful ones can be overlooked. On the other hand, complete state spaces guarantee that, given sufficient time and sufficiently rich reinforcement, the agent will produce a provably complete policy. Unfortunately, this quality is of little use in time–bounded situated domains.

6.3.3 State Transitions

Simple MDP–based models employ discrete, synchronized state transitions. In contrast, in situated domains the world state and the agent state change asynchronously in response to various events. In dynamic domains, only a subset of those events are directly caused by the agent's actions or are in agent's control. In general, events can take different amounts of time to execute, can have delayed effects, and can have different consequences under identical conditions. In short, situated domains are difficult to model properly.

Deterministic models do not capture the dynamics of most situated domains, so nondeterministic alternatives have been considered (Lin 1991). Unfortunately, most are based on unrealistic models of sensor and effector uncertainty with overly simplified error properties. They are typically based on adding Gaussian noise to each sensed state and each commanded action. However, uncertainty in situated domains does not follow Gaussian distributions but instead results from structured dynamics of interaction of the system and the environment. These dynamics play an important role in the overall behavior of the system, but are generally at a description level too low to be accurately modeled or simulated.

As an example, consider the properties of realistic proximity and distance sensors. The accuracy of ultrasound sensors is largely dependent on the incident angle of the sonar beam and the surface, as well as on the surface materials, both of which are difficult and tedious to model accurately. Infra–red and vision sensors also have similarly detailed yet entirely different properties, none of which are accurately represented with simple models. Simple noise models are tempting, but they produce artificial dynamics that, while potentially complex, do not model the true complexity

[3]In the number of state bits.

of realistic physical systems. Consequently, many elegant results of simple simulations have not been successfully repeated on more complex agents and environments.

Given the challenges of realistic modeling, it is generally very difficult to obtain transition probabilities for nondeterministic models of situated domains. Models for such domains are not readily available, and must be obtained empirically for each system by a process analogous to learning a world model. It is difficult to estimate if obtaining a world model for a given domain requires any more or less time than learning a policy for some set of goals. Consequently, insightful work on learning world models for more intelligent exploration (Sutton 1990, Kaelbling 1990) is yet to be made applicable to complex situated domains.

We have argued that accurate models of situated domains are difficult to obtain or learn. Instead, we will focus in this work on learning policies in systems without explicit world models. The next section describes the general form of RL algorithms that have been used for such policy learning.

6.3.4 Algorithms

Reinforcement learning algorithms have the following general form (Kaelbling 1990):

```
1. Initialize the learner's internal state $I$ to $I_0$.
2. Do Forever:
      a. Observe the current world state $s$.
      b. Choose an action $a = F(I, s)$
         using the evaluation function $F$.
      c. Execute action $a$.
      d. Let $r$ be the immediate reward for
         executing $a$ in world state $s$.
      e. Update the internal state $I = U(I, s, a, r)$
         using the update function $U$.
```

The internal state I encodes the information the learning algorithm saves about the world, usually in the form of a table maintaining state and action data. The update function U adjusts the current state based on the received reinforcement, and maps the current internal state, input, action, and reinforcement into a new internal state. The evaluation function F maps an internal state and an input into an action based on the information stored in the internal state. Different RL algorithms vary in their definitions of U and F.

The predominant methodology used in RL is based on a class of *temporal differencing* (TD) techniques (Sutton 1988). All TD methods deal with assigning credit or blame to past actions by attempting to predict long–term consequences of each action in each state. Sutton's original formalization of temporal differencing (TD(λ)) deals with such predictions in Markovian environments, and covers a large class of learning approaches. For example, Bucket Brigade, the delayed reinforcement learning method used in Classifier Systems, is an instance of TD (Matarić 1991). Q-learning (Watkins 1989), the most commonly known and used TD algorithm, is defined and explained in Appendix A, as background for subsequent comparison.

6.3.5 Learning Trials

Performance properties of various forms of TD applied to Markovian environments have been extensively studied (Watkins & Dayan 1992, Barto, Bradtke & Singh 1993, Jaakkola & Jordan 1993). Provable convergence of TD and related learning strategies based on dynamic programming is asymptotic and requires infinite trials (Watkins 1989). Generating a complete policy, however incorrect, requires time exponential in the size of the state space, and the optimality of that policy converges in the limit as the number of trials approaches infinity. In practice, this translates into hundreds of thousands of trials for up to ten–bit states. Thus, even in ideal Markovian worlds the number of trials required for learning is prohibitive for all but the smallest state spaces.

The situated learning problem is even more difficult, however. Assuming an appropriately minimized state space, a learner may still fail to converge, due to insufficient reinforcement.

6.3.6 Reinforcement

Temporal credit assignment, assigning delayed reward or punishment, is considered to be one of the most difficult and important problems in reinforcement learning[4]. Temporal credit is assigned by propagating the reward back to the appropriate previous state–action pairs. Temporal differencing methods are based on predicting the expected value of future rewards for a given state–action pair, and assigning credit locally based on the difference between successive predictions (Sutton 1988).

Reward functions determine how credit is assigned. The design of these functions is not often discussed, although it is perhaps the most difficult aspect of setting up

[4]The first statement of the problem is due to Samuel (1959), whose checkers–learning program learned to reward moves that eventually lead to "a triple jump."

a reinforcement learning algorithm. The more delayed the reward, the more trials the learning algorithm requires, the longer it takes to converge. Algorithms using immediate reinforcement naturally learn the fastest.

Most reinforcement learning work to date has used one of the following two types of reward: immediate or very delayed. We postulate, however, that situated domains tend to fall in between the two popular extremes, providing some immediate rewards, plenty of intermittent ones, and a few very delayed ones. Although delayed reinforcement, and particularly *impulse reinforcement* that is delivered only at the single goal, eliminates the possibility for biasing the learning, it usually makes it prohibitively difficult. Most situated learning problems do not resemble mazes in which the reward is only found at the end. Instead, some estimates of progress are available along the way. These estimate can be intermittent, internally biased, inconsistent, and occasionally incorrect, but if used appropriately, can significantly speed up learning. The approach presented in the next chapter takes advantage of such intermediate estimates to shape reinforcement and accelerate learning.

6.3.7 Multiple Goals

We have argued that impulse reinforcement related to a single goal makes learning prohibitively slow. Furthermore, single goal agents are rare in situated domains. Instead, situated agents are best viewed as having multiple goals, some of which are maintained concurrently, while others are achieved sequentially. For example, in our previously described foraging task, an agent maintains a continuous low–level goal of collision avoidance, also keeps a minimal distance from other agent in order to minimize interference, may attempt to remain in a flock, and may be heading home with a puck.

Most RL models require that the learning problem be phrased as a search for a single goal optimal policy, so that it can be specified with a global reward function. Not surprisingly, if the world or the goal changes, a new policy must be learned, using a new reward function. The existing policy will conflict with the new learning and will need to be "forgotten."

In order to enable learning of a multi–goal policy, the goals must be formulated as subgoals of a higher–level single optimal policy. Therefore they must be sequential and consistent. To enforce a specific goal sequence, the state space must explicitly encode what goals have been reached at any point in time, thus requiring added bits in the input state vector (Singh 1991). Although a natural extension of the RL framework, this method requires the state space to grow with each added goal, and cannot address concurrent goals. Sequences of goals fail to capture the dynamics of complex situated worlds and agents that may have one or more high–level goals of achievement, and also a number of maintenance goals, the interaction of which has important effects on the agents' behavior and rate of learning.

A more general solution to multiple goals within the traditional framework is to use separate state spaces and reinforcement functions for each of the goals and merge them Whitehead, Karlsson & Tenenberg (1993). However, merging policies assumes that the necessary information for utility evaluation is available to the agent. However, as previously discussed in relation to game–theoretic approaches (see Section 2.4.7), that assumption may not hold in many situated domains.

6.3.8 Related Work

Work in computational RL has been active since the fifties and has become particularly lively in the last decade. The majority of the contributions have been theoretical in nature. For thorough reviews of reinforcement learning as applied to well–behaved learning problems see Watkins (1989) and Sutton (1988). For more recent work on improved learning algorithms for situated agents, largely applied to simulated domains, see Kaelbling (1990) and Whitehead (1992). This section will focus on empirical learning work with situated agents.

Whitehead & Ballard (1990) and Whitehead (1992) addressed the perceptual aliasing problem in situated RL. They proposed an approach to adaptive active perception and action that divided the control problem into two stages: a state identification stage and a control stage, and applied appropriate learning methods to each. The approach was demonstrated on a simulated block stacking task, but has not been tested in an embodied domain.

Kaelbling (1990) used a simple mobile robot to validate several RL algorithms using immediate and delayed reinforcement applied to learning obstacle avoidance.

Maes & Brooks (1990) applied a statistical reinforcement learning technique using immediate reward and punishment in order to learn behavior selection for walking on a six–legged robot. The approach was appropriate given the appropriately reduced size of the learning space and the available immediate and accurate reinforcement derived from a contact sensor on the belly of the robot, and a wheel for estimating walking progress.

More delayed reinforcement was used by Mahadevan & Connell (1991*a*) in a box–pushing task implemented on a mobile robot, in which subgoals were introduced to provide more immediate reward. Mahadevan & Connell (1991*b*) experimented with Q–learning using monolithic and partitioned goal functions for learning box–pushing, and found subgoals necessary.

Chapman & Kaelbling (1991) and Mahadevan & Connell (1991*a*) demonstrated complementary approaches for generalization. Chapman & Kaelbling (1991) started with a single most general state and iteratively split it based on statistics accumulated over time. Splitting is based on the relevance of each state bit; when one is found to be relevant, the state space is split in two, one with that bit on, and the other with it off. In contrast, Mahadevan & Connell (1991*a*) started with a fully differentiated specific set of states, and consolidated them based on similarity statistics accumulated over time.

Aside from traditional unsupervised reinforcement learning methods described above, other techniques have also been explored. Pomerleau (1992) used a supervised connectionist learning approach to train steering control in an autonomous vehicle based on generalizing visual snapshots of the road ahead.

Thrun & Mitchell (1993) demonstrated a connectionist approach to learning visual features with a camera mounted on a mobile robot. The features are not assigned by the designer but are instead selected by the network's intermediate representations. Not surprisingly, the result is not semantically meaningful to a human observer, but is nonetheless well suited for the robot's navigation task.

The work presented here is, to the best of the author's knowledge, the first attempt at applying reinforcement learning to a collection of physical robots learning a complex task consisting of multiple goals. Parker (1994) implemented a non–RL memory–based style of parameter–learning for adjusting activation thresholds used to perform task allocation in a multi–robot system. Tan (1993) has applied traditional RL to a simulated multi–agent domain. Due to the simplicity of the simulated environment, the work has relied on an MDP model that was not applicable to this domain. Furthermore, Tan (1993) and other simulation work that uses communication between agents relies on the assumption that agents can correctly exchange learned information. This often does not hold true on physical systems whose noise and uncertainty properties extend to the communication channels.

6.4 Summary

This chapter has overviewed the key properties of reinforcement learning strategies based on Markov Decision Process models, and their implications on learning in situated domains. Learning algorithms based on dynamic programming and traditionally applied to such Markovian domains were also discussed. Finally, related robot learning and reinforcement learning work was reviewed.

Two main problems arise when the standard MDP formulation is applied to our multi–agent domain: 1) the state space is prohibitively large, and 2) delayed reinforcement is insufficient for learning the foraging task. The next chapter introduces a method of reformulating the learning problem in order to make learning both possible and efficient in the complex domain used in this work.

Chapter 7

The Learning Approach

This chapter describes a formulation of the proposed reinforcement learning problem in order to make learning possible and efficient in the complex situated domain at hand, as well as in situated domains in general.

In order to deal with the complexity and uncertainty of situated domains, a learning algorithm must use an appropriate level of description. A learner using too low a level of description will result in a state space so large as to make the learning prohibitively slow. In contrast, a learner based on too corse a level of description cannot discover any novel and potentially useful strategies outside the structured space allowed by the coarse representation.

An appropriate representation shapes the state space into an expressive but tractable learning space. An effective learning algorithm, then, searches this learning space efficiently. Thus, given the complexities of situated agents and environments, as well as those of reinforcement learning algorithms, any approach to situated learning should have the following properties.

A model for situated learning should:

1. minimize the learner's state space
2. maximize learning at each trial

This chapter will address each of the desired properties in turn. First, an approach will be described for minimizing the state space in order to make the learning problem tractable. Second, an approach for shaping reinforcement will be proposed that makes learning more efficient. In both cases, the traditional primitives of reinforcement learning (states, actions, and reinforcement) will be reformulated ($\longrightarrow$) into subtly different but pragmatically more effective counterparts, as follows:

1. states & actions $\longrightarrow$ conditions & behaviors
2. reinforcement $\longrightarrow$ multi–modal feedback

7.1 Reformulating the Problem

Traditional state–action models used by many RL approaches tend to be based on a level of description inappropriate for complex situated domains. Their representations abstract away important control details, but still must search excessively large state spaces representing the agent's entire world. A large state space is not so much a sign of a difficult problem as it is of a poorly formulated one. We propose the following reformulation that uses a more appropriate representation for the problem of learning in noisy and inconsistent worlds:

> *Reinforcement learning in situated domains can be formulated as learning the conditions necessary and sufficient for activating each of the behaviors in the repertoire such that the agent's behavior over time maximizes received reward.*

This formulation accomplishes the desired goal of diminishing the learning space by using conditions and behaviors instead of states and actions, with the effect of elevating the level of description of the learning problem.

7.1.1 Behaviors

The first part of the thesis has argued that behaviors are an intuitive and effective level of description for control, and described a methodology for selecting and combining basic behaviors for a given domain and set of goals. *Behaviors* were defined as goal–driven control laws that hide the details of control. The same reasons that made behaviors a useful abstraction in control make them an appropriate and efficient basis for learning.

Behaviors are more general than actions because they are not tied to specific detailed states but instead triggered by a set of general conditions. For instance, a *wall-following* behavior applies to any environment and any wall that the agent can sense, and is not dependent on the agent's exact state including such information as its (x, y) position, whether it is carrying a puck, and what is in front or behind it.

It can be said that much of the RL literature already uses behaviors without labeling them as such. For example, an action called "left" which transports an agent to the next square on a grid and turns it by 90 degrees, requires a complex control sequence. It is a control law that guarantees an output, such as the agent's position and orientation, and is thus identical in effect to our definition of behavior. Such a behavior, however, may not be realistic in continuous, noisy domains. In general, atomic actions of simulated grid worlds can translate into arbitrarily complex behaviors on embodied systems. Consequently, situated, embodied agents often use a very different set of behavior primitives, specifically designed for the particular dynamics of the agent and its interaction with the world.

Behaviors elevate control to a higher and more realizable level. However, the complexity of reinforcement learning lies in the size of the learning space, which is traditionally exponential in the state space of the agent. In order to significantly accelerate learning, we must minimize this space as well. We propose to do so by abstracting the learning space to a higher level, structured by the granularity of the conditions necessary for executing each of the behaviors.

Using behaviors abstracts away the details of the low–level controller, while still using realizable units of control, and thus guaranteeing the results, or postconditions, of each behavior. Similarly, conditions abstract away the low–level details of the agent's state space, and define the learning space at a higher level, by state clustering.

7.1.2 Conditions

Conditions are predicates on sensor readings that map into a proper subset of the state space. Each condition is defined as the part of the state that is necessary and sufficient for activating a particular behavior. For instance, the necessary and sufficient conditions for picking up a puck are that a puck is between the fingers of the robot.

The space of conditions is usually much smaller than the complete state space of the agent, resulting in a smaller space for the learning algorithm. Furthermore, the fewer state elements need to be sensed the less the system will suffer from error and uncertainty. Finally, the only events relevant to the agent are those that change the truth value of the predicates, i.e. the current condition. Those events are used to trigger and terminate behaviors.

Reformulating states and actions into conditions and behaviors effectively reduces the state space to a manageable size, thus making learning possible in a complex domain. The next step is to make learning efficient, by using appropriate reinforcement.

7.2 Reinforcement for Accelerated Learning

The amount and quality of the reinforcement determines how quickly the agent will learn. In nondeterministic uncertain worlds, learning in bounded time requires shaping of the reinforcement in order to take advantage of as much information as is available to the agent.

In general, reinforcement learning can be accelerated in two ways: 1) by building–in more information, and 2) by providing more reinforcement. The reward function implicitly encodes domain knowledge and thus biases what the agent can learn. Simplifying and minimizing reinforcement, as practiced by some early RL algorithms (Sutton 1990), does diminish this bias, but it also greatly handicaps, and in situated domains, completely debilitates the learner.

Domain knowledge can be embedded through a reward–rich and complex reinforcement function. This approach is effective, but the process of embedding semantics about the world into the reward function is usually *ad hoc*. In the ideal case, reinforcement is both immediate and meaningful. Immediate error signals that provide not only the sign but also the magnitude of the error result in fastest learning. As in *supervised learning*, then provide the agent with the correct answer after each trial. In learning control (Jordan & Rumelhart 1992, Atkeson 1990, Schaal & Atkeson 1994), such error signals are critical as the learning problem is usually finding a complex mapping between a collection of input parameters and the desired output. Immediate reinforcement in RL is typically a weak version of an error signal, reduced to only the sign of the error but not the magnitude or the direction.

We propose an intermediate solution based on *shaping* as a version of an error signal based on principled embedding of domain knowledge.

7.2.1 Heterogeneous Reward Functions

Monolithic reward functions with a single high–level goal, when applied to situated domains, require a large amount of intermediate reinforcement in order to aid the agent in learning. Intuitively, the more subgoals are used the more frequently reinforcement can be applied, and the faster the learner will converge. We have already argued that situated agents maintain multiple concurrent goals, and that such goals can be achieved and maintained by using behaviors as the basic unit of control and learning. Thus, a task in a situated domain can be represented with a collection of such concurrent goal–achieving behaviors. Reaching each of the goals generates an event[1] that provides primary reinforcement to the learner. The following is the

[1]A change in the conditions.

general form of such event–driven reinforcement functions:

$$R_e(c,t) = \begin{cases} r & \text{if the event E occurs} \\ 0 & \text{otherwise} \end{cases} \qquad e \neq 0$$

Event–driven reinforcement for any event E is a function of conditions c and time t. The received reinforcement r may be positive or negative.

If necessary information about the task and the appropriate sensors are available, each of the goals can be further broken down into one or more subgoals, with associated secondary reinforcement. In general, the specification of a high–level behavior provides a collection of subgoals that need to be achieved and maintained. If the achievement of a subgoal can be detected, it can be directly translated into a reinforcement function.

A general heterogeneous reward function has the following form:

$$R_e(c,t) = \begin{cases} r_{E1} & \text{if event E1 occurs} \\ r_{E2} & \text{if event E2 occurs} \\ . & . \\ . & . \\ . & . \\ r_{En} & \text{if event En occurs} \\ 0 & \text{otherwise} \end{cases}$$

The complete reward function is a sum of inputs from the individual event–driven functions. Thus, if multiple events occur simultaneously, appropriate reinforcement for all of them is received from multiple sources.

Even–driven reinforcement functions are illustrated with the following example:

- A robot receives reward R_a whenever it avoids an obstacle, and reward R_h whenever it reaches home.
- The corresponding reward function appears as follows:

$$R(c,t) = \begin{cases} r_a & \text{if an obstacle is avoided} \\ r_h & \text{if home is reached} \\ 0 & \text{otherwise} \end{cases}$$

- If the robot happens to be avoiding an obstacle and reaches home at the same time, it receives reinforcement from both sources concurrently:

$$R(c,t) = r_a + r_h$$

As the above example illustrates, each of the heterogeneous reward functions provides a part of the structure of the learning task, and thus speeds up the learning.

Event–driven reward functions associate reinforcement with the achievement of goals and subgoals through the application of associated behaviors. They deliver reward or punishment in response to events, i.e. between behaviors. The next section describes a shaping mechanism for providing reinforcement during the execution of a behavior.

7.2.2 Progress Estimators

Many goals have immediately available measures of progress, since few tasks need to be defined as long sequences of behaviors without any feedback. Progress estimators use domain knowledge to measure progress during a behavior and, if necessary, to trigger principled behavior termination.

Feedback as a learning signal can be received from a one or more goals. Consider the following example:

- The robot's task is to learn to take pucks home.
- Having found a puck, the robot can wait until it accidentally finds home and then receives a reward.
- Alternatively, it can use a related subgoal, such as getting away from the food/puck pile, for feedback.
- In such a scheme, the longer the robot with a puck stays near food, the more negative reinforcement it receives.
- This strategy will encourage the behaviors that take the robot away from the food, one of which is *homing*.

While immediate reinforcement is not available in many domains, intermittent reinforcement can be provided by estimating the agent's progress relative to its current goal and weighting the reward accordingly. Measures of progress relative to a particular goal can be estimated with standard sensors, and furthermore feedback is available from different sensory modalities.

The following are the two general forms of progress estimator functions.

$$R_p(c,t) = \begin{cases} m & \text{if } c \in C' \wedge \text{progress is made} \\ n & \text{if } c \in C' \wedge \text{no progress} \end{cases} \quad m > 0, \quad n < 0, \quad C' \subset C$$

$$R_s(c,t) = \begin{cases} i & \text{if } c \in C' \wedge \text{progress is made} \\ j & \text{if } c \in C' \wedge \text{regress is made} \\ 0 & \text{otherwise} \end{cases} \quad i > 0, \quad j < 0, \quad C' \subset C$$

C is the set of all conditions, and C' is the set of conditions associated with the given progress estimator, i.e. those conditions for which the given progress estimator is active.

R_p and R_s have different dynamics. R_p is a two–valued function that monitors only the presence and absence of progress. R_s is a three–valued function that monitors the presence and absence of progress, as well as negative progress or regress.

Progress estimators diminish brittleness of the learning algorithm in the following ways:

- decrease sensitivity to noise
- encourage exploration in the behavior space
- decrease fortuitous rewards

Each is described in turn.

Decreasing Sensitivity to Noise

Progress estimators provide implicit domain knowledge to the learner. They strengthen appropriate condition–behavior correlations and serve as filters for spurious noise. Noise–induced events are not consistently supported by progress estimator credit, and thus have less impact on the learner. Consider the following example:

- Agent A is executing behavior B in condition c and receives positive reinforcement r_p by the progress estimator R_p.
- A receives negative reinforcement r_e from R_e as a result of an event induced by a sensor error.
- The impact of the negative reinforcement is diminished by the continuous reinforcement received from R_e throughout the execution of B.

The domain knowledge behind progress estimators provides a continuous source of reinforcement to counter intermittent and potentially incorrect credit.

Encouraging Exploration

Exploration versus exploitation is one of the critical tradeoffs in machine learning. The agent must do enough exploration to discover new and potentially more efficient condition–behavior combinations, but must also optimize its performance by using the best known pairings. Ineffective exploration results in thrashing, repeatedly attempting of one or more inappropriate behaviors.

Since situated environments are event–driven, any given behavior may persist for a potentially long period of time. An agent has no impetus for terminating a behavior and attempting alternatives, since any behavior may eventually produce a reward. The learning algorithm must use some principled strategy for terminating behaviors in order to explore the condition–behavior space effectively. Progress estimators provide such a method: if a behavior fails to make progress relative to the current goal, it is terminated and another one is tried. By using domain knowledge to judge progress, progress estimators induce exploration by terminating behaviors according to common sense, rather than according to an arbitrary internal clock or some *ad hoc* heuristic.

Decreasing Fortuitous Rewards

A *fortuitous reward* is one received for an inappropriate behavior that happened to achieve the desired goal in the particular situation, but would not have that effect in general. Consider the following scenario:

- The agent has a puck and is attempting various behaviors.
- While executing avoidance in $safe-wandering$, A fortuitously enters the home region.
- Without a progress estimator, A will receive a reward for reaching home, and will thus positively associate the avoiding behavior with the goal of getting home. It will require repeated trials in order to discover, implicitly, that the correlation is based on the direction it is moving rather than on $safe-wandering$.
- Now suppose a progress estimator H is added into the learning algorithm. H generates a reward when the agent decreases its distance to home. If it fails to do so in a given time interval, the behavior is terminated.
- Although A can still receive fortuitous rewards, their impact will be smaller compared to that of the consistent progress estimator. The continuous reward for approaching home will have a discounting effect on any

fortuitous rewards the agent receives. Thus, H will bias the agent toward behaviors that decrease the distance to home.

In general, the only way to eliminate fortuitous rewards is to know the relevance of context *a priori*. Progress estimators achieve this effect incrementally, because behaviors have some measurable duration which allows progress estimators to contribute reinforcement.

7.3 Summary

This chapter has introduced a formulation of reinforcement learning based on conditions, behaviors, and shaped reinforcement in order to: 1) make learning possible and 2) make learning efficient in complex situated domains.

The described formulation is a direct extension of behavior–based control (Matarić 1992*a*, Brooks 1991*b*, Brooks 1986). The presented heterogeneous reward functions are related to subgoals (Mahadevan & Connell 1991*a*) as well as subtasks (Whitehead et al. 1993). However, unlike previous work, which has focused on learning action sequences, this work used a higher level of description. The proposed subgoals are directly tied to behaviors used as the basis of control and learning. Similarly, progress estimators are mapped to one or more behaviors, and expedite learning of the associated goals, unlike a single complete external critic used with a monolithic reinforcement function (Whitehead 1992).

Elevating the description, control, and learning level of the system to one based on perceptual conditions and behaviors instead of perceptual states and atomic actions greatly diminishes the agent's learning space and makes learning tractable. The use of heterogeneous reward functions and progress estimators builds in domain knowledge and contextual information thus making learning more efficient.

The proposed reformulation forms a better foundation for situated learning, but does not impose any constraints on the kind of learning algorithm that can be applied. Indeed, it is completely general and compatible with any reinforcement learning approaches.

The next chapter demonstrates how this formulation was applied to the task of learning foraging in a situated, multi–robot domain.

Chapter 8

Learning Experiments

This chapter describes the learning experiments conducted to test the presented approach to setting up the learning space to enable learning, and shaping reinforcement to accelerate learning in situated domains.

8.1 The Robots

The learning experiments were performed on a group of up to four fully autonomous R2 mobile robots with on–board power and sensing (Figure 8-1). Each robot consists of a differentially steerable wheeled base and a gripper for grasping and lifting objects (Figure 8-2). The robots' sensory capabilities include piezo–electric bump sensors for detecting contact–collisions and monitoring the grasping force on the gripper, and a set of infra–red (IR) sensors for obstacle avoidance and grasping (Figure 8-3).

Finally, the robots are equipped with radio transceivers, used for determining absolute position and for inter–robot communication. Position information is obtained by triangulating the distance computed from synchronized ultrasound pulses from two fixed beacons. Inter–robot communication consists of broadcasting 6–byte messages at the rate of 1 Hz. In the experiments described here, the radios are used to determine the presence of other nearby robots. As in the first set of robot experiments, the robots are programmed in the Behavior Language (Brooks 1990*a*).

8.2 The Learning Task

The learning task consists of finding a mapping of all conditions and behaviors into the most effective policy for group foraging. Individually, each robot learns to select desirable, and we hope to address it in future work.

8.7 Summary

The goal of the described learning work has been to bring to light some of the important properties of situated domains, and their impact on reinforcement learning strategies. We have described why MDP models of agent–world interactions are not effective in the noisy multi–agent domain, how the traditional notions of state and action present an inappropriately low level of system description for control and learning, and how delayed reinforcement is not sufficient for learning in our domain and other domains of similar level of complexity.

We introduced a higher–level description of the learning system, based on conditions and behaviors, that greatly diminishes the learner's state space and results in more robust control. We also introduced a methodology for shaping reinforcement in order to take advantage of more information available to the agent. In our domain shaping was necessary given the complexity of the environment–agent and agent–agent interactions. The approach consists of two methods: one that partitions the learning task into natural subgoals (behaviors) and reinforces each separately, and one that employs progress estimators to generate more immediate feedback for the agent.

The proposed formulation was evaluated on a group of physical robots learning to forage and was shown to be effective as well as superior to two alternatives. The approach is general and compatible with the existing reinforcement learning algorithms, and should thus serve to make learning more efficient in a variety of situated domains and with a variety of methodologies.

Chapter 9

Summary

The aim of this thesis has been to gain insight into intelligent behavior by increasing the level of complexity of the systems being designed and studied. In contrast to many AI systems that have focused either on complex cognition situated in simple worlds, or vice versa, the work described here has addressed situated, embodied agents coexisting and interacting in a complex domain (Figure 9-1). We hope that the methodologies and results presented here have extended the understanding of synthesis, analysis, and learning of group behavior.

Selection of the appropriate representation level for control, planning, and learning is one of the motivating forces behind this work. We have proposed a methodology for using constraints in order to derive *behaviors*, control laws that guarantee the achievement and maintenance of goals. Furthermore, we described a methodology for selecting *basic behaviors*, a basis set of such behaviors to be used as a substrate for control and learning for a given agent and environment.

We demonstrated these ideas on the problem of synthesizing coherent group behavior in the domain of planar spatial interactions. We devised a basic behavior set and showed that it meets the defining criteria, including no mutual reducibility and simple combination. We then showed how basic behaviors and their conditions can be used as a substrate for learning. Furthermore, we described a methodology for shaping reinforcement by using *heterogeneous reinforcement functions* and *progress estimators* in order to make learning possible and more efficient in dynamic multi–agent domains.

The main idea behind this work is the approach to combining constraints from the agent, such as its mechanical and sensory characteristics, and the constraints for the environment, such as the types of interactions and sensory information the agent can obtain, in order to construct constraint–based primitives for control. At the

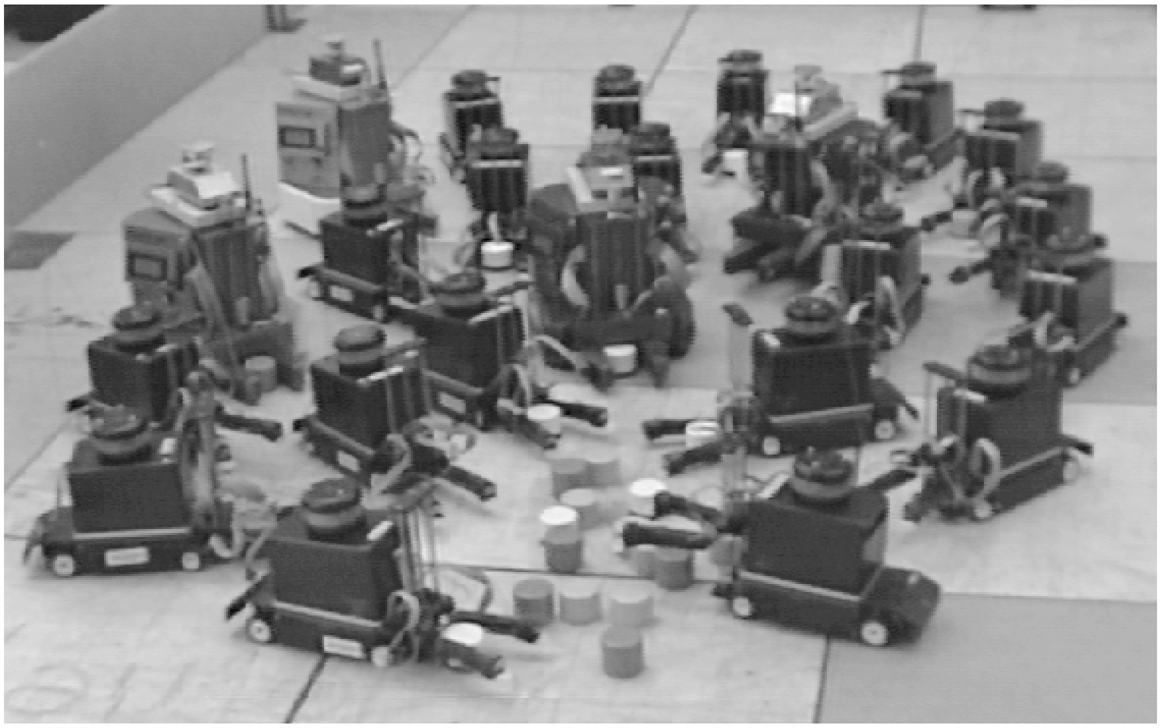


Figure 9-1: A family photo of the physical experimental agents used to demonstrate and verify the group behavior and learning work described in this thesis.

sensory end we called these primitives *conditions* and at the action end we referred to them as *behaviors*. In both cases they are a clustering of constraints that provide an abstraction at a level that makes control and learning efficient.

We have dealt with a complex multi–agent domain and a complex learning problem in order to fully confront the issues in selecting the right abstraction and representation level for situated agents. The complexity of our chosen environment, combined with the requirement of acting in real time, enforced the necessity for using a representation level that was not so low as to be computationally intractable or so high as to remove the potential of novel behavior strategies to be designed or learned by the agents.

This work is intended as a foundation in a continuing effort toward studying increasingly more complex behavior, and through it, more complex intelligence. The work on basic behaviors distills a general approach to control, planning, and learning. The work also brings to light some theoretically and empirically challenging problems and offers some effective solutions to situated learning. Future work should both analytically tighten and experimentally broaden our understanding of all those issues. The demonstrated results in group behavior and learning are meant as stepping stones toward studying increasingly complex social agents capable of more complex learning, ultimately leading toward better understanding of biological intelligence.