INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
The evolution of communication in adaptive agents

Saunders, Gregory Morris, Ph.D.

The Ohio State University, 1994
Copyright by
Gregory Morris Saunders
1994
To Kathy
ACKNOWLEDGEMENTS

This research would not have been possible without the help of many people. First and foremost, I wish to thank my advisor Dr. Jordan Pollack, for his constant enthusiasm, for his intellectual guidance, and for providing an environment flexible enough to accommodate both research and family. Thanks to the other members of my reading committee, Dr. B. Chandrasekaran and Dr. Feng Zhao, for how each has inspired me in his own way. I also thank the agencies which have given me financial support throughout graduate school: the National Science Foundation and the Office of Naval Research (through grants N00014-89-J-1200, N00014-92-J-1195, and N00014-93-1-0059).

I thank all those who made the Laboratory for Artificial Intelligence Research a more enjoyable place to work, especially Barbara Becker, for always answering "How's this sound?" for making sure I always picked up my paycheck, and for increasing my vocabulary. Thanks to LAIR connectionists past and present: Edward Large, Viet-Anh Nguyen, and David Stucki.

Thanks to the founder of the Dead Researcher's Society, Dr. John Kolen, for his willingness to answer questions and for always warning me what would come next. I had the privilege to work closely with him on several projects, one of which became Chapter III (Saunders, Kolen, and Pollack, to appear), and another which helped shape my ideas in Chapter V (Saunders, Kolen, and Pollack, 1994).
I especially thank my good friend Dr. Peter “Ole Wisey” Angeline, for giving me many gems to “keep under my hat,” for showing me how to simulate intelligence on a bowl of tapioca, and in general for making my life GNARLY. His offer to collaborate on a “simple” project became Chapter IV, and I still haven’t forgiven him.

I thank Bruce Ottens and Gary Stottlemyer at Shawnee State University, for cheerfully providing invaluable resources as this document came together.

I thank my various teachers over the years, including Drs. James McCracken and Philip Jastram, for introducing me to physics, Judge Earl E. Stephenson, for showing me how to always keep thinking, and my good friend Pat Dills, for sharing her lifetime of wisdom. Thanks to my mother-in-law Phyllis Doersch, for her support, encouragement, and sense of humor, and to my father-in-law Richard Doersch, whose genuine passion for learning generated an unending stream of questions about my work, about half of which I could answer. (I still don’t know how to apply this to the horses, and wouldn’t say if I did.)

I thank my grandfather Terenzio P. Tebaldi, who taught me the meaning of “piano piano,” my father, who taught me how to be precise, my brother Jeff, who introduced me to soccer, my sister Deb, who taught me how to “pedal, steer, and keep my balance,” and my brother Mark, who taught me all I know about backgammon. To Mom, who taught me to spell “Antarctica,” I am a doctor!

To Kathy, who taught me that refills on iced tea are free, thanks for helping even more than you know these 1 1/2 years, for everything in the past, and for everything yet to come. And finally, to Peter William, thanks for enduring so many swing rides in the last two weeks, though you never seemed to mind.
VITA

November 18, 1965 ........................................ Born – Piqua, Ohio

1988 ................................................. B.S., Mathematics, The Ohio State University, Columbus, Ohio

1988-1991 ............................................. National Science Foundation Graduate Fellow, tenure taken at The Ohio State University, Columbus, Ohio

1989 ................................................. M.S., Computer and Information Sciences, The Ohio State University, Columbus, Ohio

1991-1992 ........................................... OSU Fellow, The Ohio State University, Columbus, Ohio

1992-Present ......................................... Graduate Research Assistant, The Ohio State University, Columbus, Ohio

PUBLICATIONS


FIELDS OF STUDY

Major Field:  Computer and Information Sciences

Studies in:

Connectionism and Cognition  Prof. J. Pollack
Artificial Intelligence       Prof. B. Chandrasekaran
Theory of Computation        Prof. T. Long
TABLE OF CONTENTS

DEDICATION ............................................................................................................................. ii

ACKNOWLEDGEMENTS ...................................................................................................... iii

VITA ............................................................................................................................................... v

LIST OF TABLES ...................................................................................................................... xi

LIST OF FIGURES .................................................................................................................. xii

CHAPTER PAGE

I. INTRODUCTION ................................................................................................................. 1

1.1 Overview ............................................................................................................................. 1

1.2 The Problem of Communication ...................................................................................... 2

1.3 Motivation ........................................................................................................................... 4

1.4 Methodology ....................................................................................................................... 7

1.4.1 Preliminary Work ............................................................................................................. 7

1.4.1.1 Adaptive Behavior using Behavioral Decompositions ........................................ 8

1.4.1.2 Adaptive Behavior without Behavioral Decompositions .................................... 9

1.4.2 The Evolution of Communication .................................................................................. 9

1.4.2.1 The Tracker Task, Revisited .................................................................................... 10

1.4.2.2 Architecture and Evolution ....................................................................................... 11

1.4.2.3 Experiments .............................................................................................................. 11

1.5 Preview of Results ........................................................................................................... 12

1.6 Organization ...................................................................................................................... 12

II. BACKGROUND .................................................................................................................. 14

2.1 Introduction ....................................................................................................................... 14

2.2 Adaptive Behavior ............................................................................................................ 15

2.2.1 Autonomous Agents ................................................................................................... 16

2.2.2 The Importance of the Environment .......................................................................... 16

2.2.3 Adaptation as Structural Coupling ............................................................................ 17

2.2.4 A Philosophy on Intelligence ....................................................................................... 18
### 2.3 Reactive planning .......................................................... 20
### 2.4 Behavior-Based Approaches to Adaptive Behavior .......... 21
#### 2.4.1 The Design of Behavior-Based Agents ....................... 21
#### 2.4.2 The Learning of Behaviors ........................................ 26
### 2.5 The Design of Group Behavior ............................................. 28
#### 2.5.1 Behavior-Based AI, Revisited ........................................ 29
#### 2.5.2 Colonies ........................................................................ 31
#### 2.5.3 Designing Emergent Behaviors ..................................... 33
### 2.6 The Evolution of Communication ........................................ 36
#### 2.6.1 Adaptive Communication Protocols ............................. 37
#### 2.6.2 Evolving Communication Between the Sexes .................. 39
#### 2.6.3 BioLand ........................................................................ 42
#### 2.6.4 Synthetic Ethology .......................................................... 42
#### 2.6.5 AntFarm ......................................................................... 44
### 2.7 Remarks ............................................................................ 46

### III. A CONNECTIONIST APPROACH TO ADAPTIVE BEHAVIOR ..... 48

#### 3.1 Introduction ................................................................. 48
#### 3.1.1 Leaky Levels ............................................................... 49
#### 3.1.2 Connectionist Approaches to Multiple Behaviors ............. 50
#### 3.1.3 Behavior-Based AI ....................................................... 52
#### 3.1.4 An Alternative Approach ............................................. 54
#### 3.2 Additive Adaptive Modules .............................................. 55
#### 3.2.1 The Addam Architecture .............................................. 55
#### 3.2.2 Preemption ................................................................. 58
#### 3.3 Training Addam .............................................................. 59
#### 3.3.1 Level 0 Competence .................................................... 60
#### 3.3.2 Higher-Level Competencies ......................................... 61
#### 3.4 Results .......................................................... 62
#### 3.4.1 Performance ............................................................... 62
#### 3.4.2 Network Activations .................................................... 64
#### 3.5 Remarks .......................................................... 65
#### 3.5.1 The Appeal of Preemption ........................................... 65
#### 3.5.2 Potential Drawbacks ................................................... 67
IV. AN EVOLUTIONARY APPROACH TO ADAPTIVE BEHAVIOR

4.1 Introduction ........................................... 69
4.2 The Network Induction Problem .................. 70
4.3 The Evolution of Neural Networks ................ 72
4.3.1 Networks and Genetic Algorithms ............ 73
4.3.2 Networks and Evolutionary Programming ...... 77
4.4 The GNARL Algorithm .............................. 79
4.4.1 Selection, Reproduction and Mutation of Networks .... 81
4.4.1.1 Severity of Mutations ....................... 82
4.4.1.2 Parametric Mutation of Networks .......... 83
4.4.1.3 Structural Mutation of Networks .......... 83
4.4.2 Fitness of a Network ............................. 85
4.5 Results .................................................. 86
4.5.1 The Enable-Trigger Task ....................... 86
4.5.2 The Tracker Task .................................... 91
4.6 Remarks .................................................. 98
4.6.1 The Network Induction, in Retrospect .......... 99
4.6.2 The Induction of Modules ....................... 101
4.6.3 GNARL and Communication ................... 102

V. THE EVOLUTION OF COMMUNICATION

5.1 Introduction ........................................... 103
5.1.1 A Shift in Environments ......................... 104
5.1.2 Review of Related Work ......................... 104
5.1.3 An Alternate Approach ............................ 105
5.2 Communication with Continuous Symbols .......... 105
5.2.1 The Tracker Task, Revisited ................... 106
5.2.2 An Architecture for Communication ............ 107
5.3 Results, Part I .......................................... 109
5.3.1 Experiment 1: The Curmudgeon Strategy ........ 110
5.3.2 Experiment 2: The Diagonal Strategy .......... 114
5.4 The Tracker Task, Rethought ....................... 121
5.4.1 Analysis of First Results ......................... 121
5.4.2 A Shift in Architectures ......................... 122
5.5 Results, Part II ......................................... 123
5.5.1 Experiment 3: Constant Communication Schemes .... 124
5.5.2 Experiment 4: Oscillatory Communication Schemes ............ 128
5.5.3 The Evolution of Communication ........................................ 136
5.6 Remarks ........................................................................... 143

VI. REMARKS .............................................................................. 144

6.1 Introduction ........................................................................ 144
6.2 Dissertation Review .............................................................. 145
6.3 Beyond Structural Coupling ................................................... 146
  6.3.1 Levels of Explanation ...................................................... 147
  6.3.2 The Communicative Stance .............................................. 148
6.4 Beyond the Communicative Stance? .................................... 150
  6.4.1 Relativized Semantics ..................................................... 150
  6.4.2 The Role of Syntax ......................................................... 152
6.5 Conclusions ........................................................................ 155

LIST OF REFERENCES .................................................................. 157
LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The basic interaction primitives (from Mataric, 1993a, p. 435)</td>
<td>33</td>
</tr>
<tr>
<td>2. Initial training data for enable-trigger task of Wilson (1990)</td>
<td>88</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

FIGURE .......................................................... PAGE

1. The model of signal propagation used for my study of communication, adopted from Shannon and Weaver (1948). Vectors x and y are chosen from \{0, 1\}^n ............................................................... 2

2. My model of communication among m agents. Note three main differences between this and the Shannon/Weaver model of Figure 1: signal vectors are chosen from \( \mathbb{R}^n \), signals decay over distance, and each agent receives a summation of signals. The length of each dark arrow indicates the distance between an agent \( i \) and agent \( m \); decay is inversely proportional to distance. .............................................................. 3

3. Traditional decomposition of an agent into functional modules (from Brooks, 1986). .................................................. 21

4. Decomposition of an agent into task-achieving behaviors (from Brooks, 1986). ................................................................. 22

5. An abstract subsumptive architecture. The three boxes represent modules (implemented as FSAs), the “subsumption” bus represents the way the modules interact (via primitive operations suppression and inhibition). ............................... 23

6. A specific behavioral decomposition (Brooks, 1986). ................................. 24

7. Learning to produce the correct behaviors (from Maes, 1991a). .................. 27

8. Sets of subsumptive modules (adapted from Parker, 1993). Each module is a subsumptive system, similar to that of Figure 6. Note the similarity to the connectionist methods of capturing multiple behaviors (Pollack, 1987). In this case, however, the gating is absolute, by design. In the connectionist system, gating is trained, with the goal of making it absolute.......... 30

9. Collection of food by groups of agents (from Deneubourg, 1991). (a) Food is distributed randomly throughout the environment; (b) Clusters beginning to form; (c) Two distinct clusters emerging; (d) Two clusters evident ............ 31

10. Foraging behavior, built from Mataric’s basic interaction primitives. Italicized behaviors are basic; shaded states that indicate the agent is carrying food. (From Mataric, 1993b.) .......................... 34

11. Vector field for the avoid-static-obstacle behavior (from Arkin, 1993). ...... 35

12. Architecture for an adaptive communication protocol. Only the leader (agent 0) receives the task signal. Both agents must learn a language which
allows the leader to communicate the task to the follower (agent 1). Positive reinforcement is given iff both agents perform an action appropriate to the task. (From Yanco and Stein, 1993.)

13. Werner and Dyer's (1992) scenario for evolving communication. A 200 x 200 toroidal grid is populated with 800 male and 800 female agents. Males wander around the environment, and although they cannot “see” females, they can “hear” their signals. Each female is stationary, but can “see” males in a 5 x 5 square centered at the her location, and output a 2 or 3-bit signal (depending on the experiment) to attempt to communicate her location to the male.

14. MacLennan's topology for language learning. Each agent sees its own local environment; the task is to communicate this local state to other agents by writing symbols to the global environment. (Adopted from MacLennan, 1992, p. 640.)

15. An 8 x 8 subset of the 16 x 16 AntFarm environment. Although pheromone diffuses throughout the environment, this figure (taken from Jefferson and Collins, 1992) shows all pheromone as equivalent.

16. One of the various network architectures used by the agents of AntFarm (Collins and Jefferson, 1992).

17. A connectionist approach to modularity. The input vector is fed to two experts, each of which captures a specific behavior. Simultaneously, a task bit is fed into a gating network, which selects which expert's output is appropriate for the task. Note the similarity to Parker's (1993) method of capturing multiple behaviors within the subsumptive framework (Figure 8).

18. Subsumption's problem with sequential construction. If design of layer 1 is not begun until design of layer 0 is completed, then we will not know until layer 0 is fixed whether method $M_1$ or $M_2$ is the better choice.

19. Addam's internal architecture. Each module possesses only limited information about the activity of its predecessors. Layer 1 receives only the sum-squared activation of layer 0, implemented as a1. Similarly, layer 2 monitors the activity of its predecessors through a single input a2. Through training, each layer learns to exert control only when relevant based on the current sensors, and when none of its predecessors is active.

20. Training scenarios for level 0 behavior, along with desired responses. Circles denote patches of ice. The scenarios capture a range of situations; from each, Addam's target response moves it away from the ice.

21. An example of preemption. (a) Addam's emergent behavior in a complex
environment. The dots in the upper figure trace Addam's path as it moves through the environment in search of food. (b) Activity of each of Addam's layers over time.  

22. The dual representation scheme used in genetic algorithms. The interpretation function maps between the elements in recombination space on which the search is performed and the subset of structures that can be evaluated as potential task solutions.  

23. The competing conventions problem (Schaffer, Whitley, and Eshelman, 1992). Bit strings A and B map to structurally and computationally equivalent networks that assign the hidden units in different orders. Because the bit strings are distinct, crossover is likely to produce an offspring that contains multiple copies of the same hidden node, yielding a network with less computational ability than either parent.  

24. The evolutionary programming approach to modeling evolution. Unlike genetic algorithms, evolutionary programs perform search in the space of networks. Offspring created by mutation remain within a locus of similarity to their parents.  

25. Sample initial network. The number of input nodes ($m_{in}$) and number of output nodes ($m_{out}$) is fixed for a given task. The presence of a bias node ($b = 0$ or $1$) as well as the maximum number of hidden units ($h_{max}$) is set by the user. The initial connectivity is chosen randomly (see text). The disconnected hidden node does not affect this particular network's computation, but is available as a resource for structural mutations.  

26. An FSA that defines the enable-trigger task (Williams, 1990). The system is given a data stream of bit pairs $\{(a_1, b_1), (a_2, b_2), \ldots\}$, and produces an output of 0's and 1's. To capture this system's input/output behavior, a connectionist network must learn to store state indefinitely.  

27. Connectivity of two recurrent networks found in the enable-trigger experiment. (a) The best network of generation one. (b) The best network of generation 765. This network solves the task for all strings of length eight.  

28. Different network topologies explored by GNARL during the first 540 generations on the enable-trigger problem. The presence of a link between node i and j at generation $g$ is indicated by a dot at position $(g, 10 \ast i + j)$ in the graph. Note that because node 3 is the output node, there are no connections from it throughout the run. The arrow designates the point of transition between the first two training sets.  

29. The ant problem. The trail is connected initially, but becomes progressively more difficult to follow. The underlying 2-d grid is toroidal, so that position
"A" is the first break in the trail – it is simple to reach this point. Positions "B" and "C" indicate the only two positions along the trail where the ant discovered in run 1 behaves differently from the 5-state FSA of Jefferson, et al., (1992) (see Figure 32).

30. The semantics of the I/O units for the ant network. The first input node denotes the presence of food in the square directly in front of the ant; the second denotes the absence of food in this same square. This particular network finds 42 pieces of food before spinning endlessly in place at position P, illustrating a very deep local minimum in the search space.

31. The Tracker Task, first run. (a) The best network in the initial population. Nodes 0 & 1 are input, nodes 5-8 are output, and nodes 2-4 are hidden nodes. (b) Network induced by GNARL after 2090 generations. Forward links are dashed; bidirectional links & loops are solid. The light gray connection between nodes 8 and 13 is the sole backlink. This network clears the trail in 319 epochs. (c) Jefferson et al.'s fixed network structure for the Tracker task.

32. FSA hand-crafted for the Tracker task in Jefferson, et al., (1992). The large arrow indicates the initial state. This simple system implements the strategy "move forward if there is food in front of you, otherwise turn right four times, looking for food. If food is found while turning, pursue it, otherwise, move forward one step and repeat." This FSA traverses the entire trail in 314 steps, and gets a score of 81 in the allotted 200 time steps.

33. Limit behavior of the network that clears the trail in 319 steps. Graphs show the state of the output units Move, Right, Left. (a) Fixed point attractor that results for sequence of 500 "food" signals; (b) Limit cycle attractor that results when a sequence of 500 "no food" signals is given to network; (c) All states visited while traversing the trail; (d) The path of the ant on an empty grid. The Z axis represents time. Note that $x$ is fixed, and $y$ increases monotonically at a fixed rate. The large jumps in $y$ position are artifacts of the toroidal grid.

34. Limit behavior of the network of the second run. Graphs show the state of the output units Move, Right, Left. (a) Fixed point attractor that results for sequence of 3500 "food" signals; (b) Limit cycle attractor that results when a sequence of 3500 "no food" signals is given to network; (c) All states visited while traversing the trail; (d) The path of the ant on an empty grid. The $z$ axis represents time. The ant's path is comprised of a set of "railroad tracks." Along each track, tick marks represent back and forth movement. At the junctures between tracks, a more complicated movement occurs.
There are no artifacts of the toroidal grid in this plot, all are actual movements (cf. Figure 33d).

35. Environment for the modified Tracker task.

36. The semantics of the I/O units for evolving communication. The "food/nofood" inputs and "move/left/right/noop" outputs are from the Tracker task. The additional nodes, described in the text, give the agent the ability to perceive and generate signals.

37. The "sound-like" model of signal propagation created by Equation 10. Agent 1 outputs a relative signal strength of 1; agent 2 outputs a relative signal strength of 2; and agent 3 outputs a relative signal strength of 3. The topological map indicates how these signals interact, with the highest elevations corresponding to the strongest signals.

38. The "Curmudgeon" strategy (2 agents, 1 communication channel, no noise). Primitives are "Move," "Left," "Right," and "Noop." (a) t=1; (b) t=25; (c) t=50; (d) t=75; (e) t=100; (f) t=125; (g) t=150; (h) t=175; (i) t=200. See text for details.

39. Profile of curmudgeon 0 (200 time steps). The lowest graph is the food input: when food is detected, the value spikes to one; otherwise it is zero. The input signal reflects communication from the other agent. The value of the hidden unit reflects internal state. Behaviors are "Move," "Turn left," "Turn right," and "Noop," with values 0-3, respectively.

40. Profile of curmudgeon 1 (200 time steps). This agent's behavior is sensitive to the input signal from curmudgeon 0, but not in a way that qualifies as communication. See text for details.

41. Three FSA-initialized agents, before any evolution (one communication channel, no noise). Primitives are "Move," "Left," "Right," and "Noop." (a) t=1; (b) t=25; (c) t=50; (d) t=75; (e) t=100; (f) t=125; (g) t=150; (h) t=175; (i) t=200.

42. Three FSA-initialized agents, after evolution (one communication channel, no noise). Primitives are "Move," "Left," "Right," and "Noop." (a) t=1; (b) t=25; (c) t=50; (d) t=75; (e) t=100; (f) t=125; (g) t=150; (h) t=175; (i) t=200.

43. Profile of the black agent (FSA-initialized, 200 time steps). Behaviors are "Move," "Turn left," "Turn right," and "Noop," with values 0-3, respectively. This agent has learned to correlate oscillation of its output signal with the absence of food.

44. Profile of the white agent (FSA-initialized, 200 time steps). The behavior of this agent depends only upon the presence of food, and not upon the varying input signal.
45. Profile of the gray agent (FSA-initialized, 200 time steps). Despite the large variation in its input signal, this agent has constant behavior. This is due to the fact that it never encounters food.

46. An agent architecture which prevents pseudo-random solutions of Section 5.3. The “Move, Left, Right, and Noop” actions are condensed into “Follow FSA,” one particular strategy found by GNARL. The “Follow gradient” nodes give the agent an Addam-like ability to respond to environmental signals.

47. Scenes of evolved communication, 2 agents, 1 communication channel, no noise. Primitives are “Follow FSA” or “Follow signal.” (a) Initial positions: neither agent can sense food; (b) The black agent just reaches food, time is t=20; (c) Recruitment – first agent attracting the second, t=40; (d-g) Scenes at t=60, 80, 100, 140, respectively; (h-i) Recruitment again, though now in reverse, t=180 and 200, respectively.

48. Profile of the black agent (200 time steps). Behavior is either “Follow FSA” (low) or “Follow Signal” (high). This agent has learned to correlate the magnitude of its output signal with the presence of food.

49. Profile of the white agent (200 time steps). The point where recruitment occurs (Figure 47c) corresponds to the first spike in the behavior profile of this agent. When this agent reaches food (between Figures 47c and d), the behavior reverts to following the FSA.

50. Scenes of evolved communication, 3 agents, 1 communication channel, no noise. Primitives are “Follow FSA” or “Follow signal.” (a) Initial condition, circles denote signal 0; (b) After one time step, signal 0 has shrunk to its minimum value. It oscillates between the two extremes when no food is present; (c-i) t=25, 50, ..., 300.

51. Agent paths for the 3-agent case. Dots indicate path of agents (food and signals have been removed for clarity). (a) Agent 0 recruiting the others. After food has been consumed, agents 1 and 2 stay together, but agent 0 strikes out on a different path.

52. Profile of the black agent (300 time steps). This agent has learned to correlate oscillation of its output signal 0 with the presence of food.

53. Profile of the white agent (300 time steps). The five spikes in behavior indicate points where the agent follows signal 0, as can be seen in Figure 50. Because the agent perceives no food during this time, the resulting behavior occurs due to the agent’s input signals.

54. Profile of the gray agent (300 time steps). Although its initial inputs (food & signals) look identical to that of the white agent, this agent’s initial
behavior oscillates between “Follow food” and “Follow signal.” The difference is resolved in Figure 55.

55. Magnified view of the first input signal of agents 1 and 2 (50 time steps). The white agent’s input begins oscillating between .03 and .04. The gray agent’s input begins oscillating between .06 and 0. It is not the magnitude, but the difference in phase which is responsible for the agents’ different behaviors.

56. Gross characteristics of the ancestors of the agents which demonstrated a constant communication scheme (Section 5.5.1). This figure shows generations 0 - 700. (The arrow denotes generation 475, used later.)

57. Evolution of network weights $w_{00} - w_{21}$, generations 0-475. Non-links are shown as a zero weight.

58. Evolution of network weights $w_{22} - w_{43}$, generations 0-475. Non-links are shown as a zero weight.

59. Snapshots of the agents in their environment at t=10 and t=30 (cf. Figure 47). The agents come from the final solution at generation 700. The particular time steps, chosen arbitrarily, simply capture the situation both before and after the black agent finds the food.

60. Functional difference of the agents over time (generations 0 - 475). Two events precede the phase transition: a distinction in the food/no-food signals by agent 0, and a recognition of this distinction by agent 1.

61. Comparison of input streams across experiment 3 (Section 5.5.1) and experiment 4 (Section 5.5.2) for the evolved white agent, t=0-50.

62. Two grammars, written to capture the language of the agents in experiments 3 (Section 5.5.1) and 4 (Section 5.5.2), respectively. S=sentence, FP=food-phrase, NP=nofood-phrase, V=value.
CHAPTER I
INTRODUCTION

1.1 Overview

The field of adaptive behavior holds that higher-level cognitive skills arise from the more primitive ability of an agent to adapt to its environment. Although many behaviors have been studied in this bottom-up fashion (e.g., obstacle avoidance, wandering, environment exploration, food collection, planning, predator avoidance, locomotion, action selection, flocking, etc.), relatively few people have studied communication as adaptive behavior. In this dissertation, I explore how communication can be understood as an adaptation by agents to their environment.

I begin by looking at behavior-based methods for agent evolution, and propose a connectionist version of subsumption which supports learning. I reject this as a method of evolving communication, however, because of certain assumptions the approach requires. I then turn to evolutionary programming, a population-based search technique, and show how it can be used to evolve agents with far fewer assumptions.

With this background, I move on to the evolution of communication. I begin with a set of independent agents, instantiated as recurrent neural networks. After arguing against systems which use discrete symbols to evolve communication, I supply my agents with a number of continuous communications channels. The agents use these channels to initiate real-valued signals which propagate through the environment, decaying over distance.
Figure 1: The model of signal propagation used for my study of communication, adopted from Shannon and Weaver (1948). Vectors $\mathbf{x}$ and $\mathbf{y}$ are chosen from $\{0, 1\}^n$.

perhaps being perturbed by environmental noise. Initially, the agents' signals appear random; over time, a structure emerges as the agents learn to communicate task-specific information about their environment. I demonstrate how different communication schemes can evolve for a task, and then discover a commonality between the schemes in terms of information passed between agents. From this I discuss what it means to communicate, and describe how a semantics emerges in the agents' signals relative to their task domain.

1.2 The Problem of Communication

Many problems impede the design of multi-agent systems, not the least of which is the nature of communication among agents. Indeed, one finds even a good definition of "communication" elusive, in part because the term spans such diverse phenomena as pheromone signaling in insects, birdsong, gestures in primates, and natural language in humans. My definition of communication will be refined throughout this dissertation, but as a first approximation I propose the following. Consider the Shannon/Weaver (1948) model of communication: one agent sends a signal down a channel, where it is possibly distorted by noise, and then perceived by another agent (Figure 1). Shannon and Weaver showed how redundant codes can allow perfect transmission of signals despite noisy
channels, but their work focuses on the \textit{syntactic} nature of the signals, and is mute as to the issue of what the signals convey to the agents.

To study the evolution of communication among agents, I extend the Shannon/Weaver model in three ways (Figure 2). First, I switch from binary to continuous channels, since I will show in Chapter III that a continuous substrate is important for evolution of agents. Second, I remove the directedness of communication. Rather than have one particular agent send a signal to another, an agent initiates an environmental signal whose amplitude decays over distance and is perturbed by environmental noise. Other agents perceive not any particular signal, but a summation of all signals along a channel. Inspired by communication through speech and audition, this method of signal propagation assumes little about the nature of communication except matched channels. Finally, I embed the agents in an environment, and evolve them to perform a task. Communication is not an end, but a means (Winograd and Flores, 1986), and the real concern is how such a channel can be exploited by a group of agents in the context of problem-solving.

Given this model, I define \textit{communication} as the transmission of a signal from one agent to another through a noisy environment, and a \textit{communication scheme} as the set of signals passed among agents under various conditions of the task. Note that for now I place no restrictions on the content of the signals; in fact, under the current definition random noise qualifies as communication. I will revise this definition in later chapters, but for present purposes it will suffice.

Implicit in Figure 2 is a certain plasticity of communication in terms of what information an agent may transmit to the others, and in terms of how an agent may respond to the signals of its peers. The model provides only the potential for meaningful
Communication channels, still of width n, though now signals decay over distance

Agent 1 transmits
\[ \vec{x} = (x_1, x_2, \ldots, x_n) \]

Agent 2 transmits
\[ \vec{y} = (y_1, y_2, \ldots, y_n) \]

Agent \( m-1 \) transmits
\[ \vec{z} = (z_1, z_2, \ldots, z_n) \]

Agent \( m \) perceives
\[ \vec{s} = (s_1, s_2, \ldots, s_n) \]

**Figure 2:** My model of communication among \( m \) agents. Note three main differences between this and the Shannon/Weaver model of Figure 1: signal vectors are chosen from \( \mathbb{R}^n \), signals decay over distance, and each agent receives a summation of signals. The length of each dark arrow indicates the distance between an agent \( i \) and agent \( m \); decay is inversely proportional to distance.

communication: the evolutionary scheme must evolve a useful communication scheme for a task. Thus the problem of evolving communication can be restated more precisely: **how can a set of agents, transmitting and receiving signals amongst each other as in Figure 2, evolve a communication scheme which aids in the performance of a given task?** If we were designing a communication scheme, there would be two aspects to this problem: determining the information conveyed between agents, and determining how to encode that information into signals. My approach makes no distinction between these aspects; the evolutionary algorithm governs both of them simultaneously.

**1.3 Motivation**

Problem-solving by distributed agents has been studied in many fields, including AI (e.g., Huhns, 1987), but the role of communication in multi-agent systems remains one of the
most important open issues in multi-agent system design (Brooks, 1991b; Arkin and Hobbs, 1993). This section provides my motivation for turning to evolution as a way of studying the communication problem.

In many works, the issues involved in communication are ignored, and researchers design by hand the interagent communication scheme (e.g., Brooks and Flynn, 1989; Parker, 1993; Deneubourg, et al., 1991; Colomni, Dorigo, and Maniezzo, 1992; Goss and Deneubourg, 1992). A designed communication scheme offers the advantage of being well-understood; furthermore, it may yield quick solutions to a given problem. But design offers little information to the larger problem of understanding the range of possible schemes, and when faced with many different tasks, designing individual communication schemes for each may be quite time-consuming. And because engineered solutions more often reflect the biases of the engineer than the demands of the task, a designed communication scheme may lack efficiency or robustness for novel tasks.

Evolution, or genetic search, lends insight into the nature of communication in several ways. As a practical matter, evolution opens the door to task-specific languages, i.e., communication schemes in which the information conveyed is tuned to the demands of a particular task. Discourse within a field, replete with jargon and acronyms, reminds us that natural language adapts wonderfully to meet people's communicative needs. Communication within a group of agents, robotic or simulated, should possess similar flexibility. Rather than adopting a single, fixed language for all tasks or implementing a new language for every task, we can allow languages to emerge based on the communicative needs of the tasks. Evolutionary algorithms can facilitate this process by adapting and refining communication schemes as they are used in different contexts.

1. These works, among others, will be discussed in detail in Section 2.5.
needs defined by a given task. This potentially allows increased problem-solving efficiency in terms of packing only relevant information in a communication channel.

Evolution of communication offers a certain robustness in the sense that it can adapt to noisy channels. If the noise in the channels is known in advance, then robust communication schemes can be designed using redundant codes (Shannon and Weaver, 1948), but designing specific responses for all potential noise levels is too cumbersome. Giving a set of agents the ability to adapt their communication scheme allows them to independently respond to dynamic noise.

Turning to evolution as a designer removes the constraint of understandability from communication; potentially, communication schemes can emerge from the communicative needs of the agents actually solving a given problem, with little or no regard for explanatory clarity. Of course nothing prevents us from analyzing a seemingly inscrutable evolved communication scheme, and from this we may learn an approach to distributed problem-solving otherwise overlooked.

Engineering considerations aside, evolution of communication schemes offers insight into language development; in particular, into the issue of bootstrapping. I begin with a set of random agents, each emitting and responding to signals in its own way. The path from this cacophony to conveyance of task-specific information is by no means clear, either in simulation or reality. Studying this process in simulation affords the advantage of detailed analysis: once (if?) useful communication is achieved, it is a simple matter to backtrack to the Tower of Babel, examining how the information conveyed changes over time.
Finally, studying the evolution of communication should shed light on the more general problem of intelligence. My approach to evolving communication, described below, will focus on *adaptation*, which has itself been proposed as the basis for intelligence (e.g., Beer, 1990). Characterizing those aspects of intelligence best understood as adaptation remains an open issue (Beer, 1990; Wilson, 1991); exploring how (or if) communication may be understood in this manner should help define an important boundary.

1.4 Methodology

I approach the evolution of communication as a problem in *adaptive behavior*, a discipline which explores methods by which simple creatures adapt to their environment as a way of studying intelligence (Beer, 1990; Meyer and Guillot, 1991; Wilson, 1991). Under this view, an agent *adapts* to its environment by modifying its internal structure in such a way that allows the agent to achieve its goals:

> Adaptive behavior derives from a structural congruency between the dynamics of an intelligent agent’s internal mechanisms and the dynamics of its external environment (Beer, 1990, p. 17).

An environment in which multiple agents simultaneously evolve communication possesses complex dynamics, as I will describe shortly. The path to this case will wind through various environments, each demonstrating an important point about adaptive behavior. Before embarking on this journey, however, I first describe the stops.

1.4.1 Preliminary Work

This subsection describes two major experiments regarding adaptive behavior. Although preliminary to the issue of communication, they each aid our understanding about
achieving desired agent behavior, and they will both be used in the work which follows below.

1.4.1.1 Adaptive Behavior using Behavioral Decompositions

My first experiment explores one particular issue in agent evolution: how to learn to satisfy multiple behavioral constraints. To solve the task, the agent must learn to modulate its behavior according to its current input.

Connectionists have attempted to solve this type of problem by building modularity into their architecture (e.g., Jacobs, Jordan, and Barto, 1990), often explicitly discouraging interactions between modules (e.g., Nowlan and Hinton, 1990). Similar isolation of modules has been adopted by the subsumption architecture (see Brooks, 1986, 1991a).

My work, drawing from the philosophical discussion of Chandrasekaran and Josephson (1993), adopts a multi-level view of behavior. There is no single level of description which captures all the aspects of an agent’s behavior. Any given level has its own merits and drawbacks: some behaviors will be easily describable at that level, while other behaviors will require an appeal to a different level of description. Chandrasekaran and Josephson propose that agents be described by a set of “leaky levels” where the underlying levels can produce behaviors not captured by the description of a single layer.

Recognizing the incompleteness of isolated behavioral descriptions, I explore the implications of allowing behaviors to “leak into” one another. The result is Addam, an agent built from Additive Adaptive Modules. Because of the connectionist implementation, no module is ever truly inactive – some low-level activation is always present. This leaking
profoundly affects overall behavior, in particular, in the way that the agent must be trained to accomplish a set of behaviors.

1.4.1.2 Adaptive Behavior without Behavioral Decompositions

Although Addam offers an interesting approach towards training agents, it makes two strong assumptions about the nature of the solution to a task; namely, that both a behavioral decomposition and the proper network structure are known. My second experiment explores a method of removing these assumptions.

Because I am eliminating such strong assumptions, I move down to a simpler environment. I adopt the Tracker task (Jefferson, et al., 1991), in which a simulated ant placed on a two-dimensional grid must learn to follow a broken trail of food. As in Addam’s world, the agent causes all changes to the environment. Unlike Addam, however, the ant’s input is deterministic.

Jefferson, et al., explored fixed-architecture solutions to this problem, i.e., connectionist networks with the number of units and their connectivity set a priori. In contrast, I introduce an evolutionary program which evolves the structure of connectionist networks along with their weight values. GNARL, Generalized Acquisition of Recurrent Links, employs a population of networks and uses a fitness function’s unsupervised feedback to guide search through network space. Annealing is used in generating both gaussian weight changes and structural modifications. Applying GNARL to the Tracker task demonstrates that the system is capable of inducing networks with complex internal dynamics, without assuming a behavioral decomposition or a fixed network structure.
1.4.2 The Evolution of Communication

The GNARL algorithm works fairly well with deterministic, static environments, but its utility in evolving communication among multiple agents remains questionable. When multiple agents are involved, the environment's dynamics clearly evolve over time from the perspective of a single agent, since the behavior of the other agents changes over time. Furthermore, when I add noise to the signals transmitted between agents, the environment becomes nondeterministic, a condition not yet untested ground.

This section summarizes my main experiments on the evolution of communication in multi-agent systems. First, I briefly describe an extension of the Tracker task, which serves as a substrate for my experiments. Next, I describe the architecture and evolutionary mechanism under investigation. Finally, I describe the set of experiments to perform.

1.4.2.1 The Tracker Task, Revisited

To study the evolution of communication in groups of agents, I extend the Tracker task of the GNARL experiment. First, I increase the number of agents. Second, I increase the size of the grid to accommodate these agents. Finally, I concentrate all the food in a small area in the center of the environment. (This task will be described in detail in Section 5.2.1.)

I assume that these modifications will shift the emphasis of the task from evolution of local internal state to evolution of communication. I concentrate the food within one area so that when an agent finds it and communicates, some food remains by the time other agents arrive. The size of the environment and the amount of food it contains far exceed the capabilities of a single ant: in the limited time available, an ant can neither search the entire
space nor consume all the food therein. Thus the task design ensures that the only method of complete success necessarily involves communication among the agents.

1.4.2.2 Architecture and Evolution

When faced with a task requiring communication, the agents of Jefferson, et al., will clearly fail – their architecture in no way supports communication. I therefore modify the agents by giving them a basic ability to communicate, i.e., I allow them to generate and perceive signals in terms of real-valued vectors, as in Figure 2. Because I do not dictate what types of information one agent will pass to the others, or how the others will respond, my architecture provides only a substrate for exploring various communication schemes through evolution. (I offer more details of the architecture along with specific motivations for the changes in Section 5.2.2.)

For the actual evolution of agents, I use the GNARL algorithm described briefly in Section 1.4.1.2 (and in detail in Chapter IV).

1.4.2.3 Experiments

With this experimental setup, my thesis can be restated more precisely as follows. Task-specific communication schemes can indeed be evolved for cooperative multi-agent systems. In particular, given the modified Tracker task (Section 1.4.2.1) and a set of agents instantiated as recurrent neural networks communicating with real-valued vectors (Section 1.2), an evolutionary algorithm (Section 1.4.1.2) is capable of evolving a communication scheme which aids the agents in performing their task.
I begin testing this claim with a very simple case: 2 agents, each with one hidden unit, capable of passing one real number between each other, with no noise. From there, I branch to other cases, adding more agents, more channels, and noise. My motivation for making these changes is to test the evolution of communication under a variety of conditions.

1.5 Preview of Results

In Section 5.5, I show how different communication schemes can evolve for the modified Tracker task. More precisely, the agents evolve different systems of signals and response. But is this really communication? Communication can occur across vast distances of both space and time, as can random noise. Given this, how do I differentiate one from the other? This question will be addressed in Chapter VI.

1.6 Organization

This chapter has provided a brief overview of the entire dissertation. The remaining chapters are organized as follows.

Chapter II provides a background on the study of adaptive behavior as a method of understanding intelligence. It begins by describing four main properties of adaptive behavior systems: autonomous agents, an environment, adaptation, and a philosophy on intelligence. The philosophy focuses on structural congruency, as described above. Finally, Chapter II reviews several works on adaptive behavior. Beginning with Brooks' behavior-based approach towards developing single-agent adaptive systems, it then describes attempts to design group behavior. Relatively few works deal with the evolution of communication, and these are reviewed in detail.
With that background, I move on to discuss my own work at developing adaptive-agent systems. Chapter III describes Addam, my first attempt to evolve an agent to have dynamics coupled to its environment (Saunders, et al., 1992). I use Brooks’ behavior-based approach as motivation, but show how to reduce the subsumption architecture’s high reliance on design.

Addam provides one approach to training an agent to survive in an environment, but the work requires several assumptions, including a behavior-based decomposition and an *a priori* fixed structure. In Chapter IV, I introduce GNARL, an algorithm for evolving agents without these assumptions (Saunders, Angeline, and Pollack, 1994; Angeline, Saunders, and Pollack, 1994). I show how GNARL can evolve agents with complex internal state, endowing the agents with dynamics which allow them to perform well in the Tracker task (Jefferson, et al., 1992; Koza, 1992).

Finally, in Chapter V, I attack the problem of evolving communication in multi-agent systems. For this task, however, the environment of the Tracker system is too simple: not only is it deterministic, it also involves only a single agent. In my system, noise perturbs signals transmitted between agents, and from the perspective of any single agent, the environment’s dynamics change as the agents evolve. I show how to evolve agents for this situation, comparing several evolved communication schemes.

Chapter VI summarizes my work, discusses whether I have evolved communication or just interesting noise, and proposes directions for future research.

---

2. GNARL resulted from an extensive collaboration with my colleague Dr. Peter Angeline, and Chapter IV comes largely from the joint work referenced above.
[Computational neuroethology] is concerned only with science; not with engineering. In particular, it is concerned with the relationship of computational neuroscience to cognitive science. It is not concerned with any attempt to create "intelligent" artefacts that employ optimal processing strategies: that is a matter for artificial intelligence (AI); the processing strategies of naturally occurring cognitive activity may well be suboptimal, and (for the purposes of this paper) cognitive science will be considered as studying only naturally plausible models of cognitive processing. That is, I'll assume that the creation of what Dennett calls "cognitive wheels" is properly only the domain of AI; although, of course, some cognitive wheels may be lurking within the cognitive science canon, yet to be refuted.

the start attempted to capture problem solving methods employed by humans (see, e.g., Newell and Simon, 1963).

Several reasons for these sorts of misconceptions exist. First, the study of adaptive agents is a very new field (or, from my perspective, a relatively new area of focus within AI). Second, because the nature of the work is so eclectic, it requires a great deal of effort to become familiar with all the research in the field. Third, even with familiarity, researchers from different areas come with what Agre calls different worldviews, i.e., different vocabularies and assumptions, making transfer between fields even more difficult (Agre, 1993). But surmounting the differences in worldviews opens the door to meaningful communication between researchers, and the possibility of cross-fertilization between diverse disciplines.

With that said, this chapter covers a broader background than just “communication in multi-agent systems.” It reviews work in adaptive agent systems in general, and then at the end focuses on studies involving communication.

2.2 Adaptive Behavior

Adaptive behavior emerged as a discipline in 1990, with the first conference on the simulation of adaptive behavior. With a second conference held in 1992, and a third scheduled for 1994, and with a journal now in its second year (Adaptive Behavior), the field seems to be thriving. But just what is the study of “adaptive behavior;” and how does it differ from the study of artificial intelligence (or does it)? This section attempts to clarify my views on these issues.
As with AI, the study of adaptive behavior defies precise definition, but the field can be characterized by four main properties (adapted from Meyer and Guillot, 1991; Wilson, 1991; Beer, 1990):

- an emphasis on autonomous agents
- an emphasis on the environment
- an emphasis on adaptation
- a philosophy on intelligence as adaptation

The following subsections explore each of these properties in detail, and previews at a high level how they will be present in my work.

2.2.1 Autonomous Agents

Adaptive behavior focuses on autonomous agents, which are "embodied systems capable of satisfying, by their own actions, given internal or external goals under the range of conditions imposed by the environments in which they are situated" (Beer, 1993, p. 1). In contrast to systems which display isolated competencies associated with intelligence (e.g., the ability to play chess), "agent" implies an entire organism, for instance an animal, a robot, or even an animat (Wilson, 1991), a simulated "critter".

One of the main benefits of the agent-based view, often understated in the literature, is that of reproducibility. Defining an agent implies a precise characterization of the system's level of abstraction in terms of inputs and outputs, allowing others to easily explore similar agents. In fact, I do just this below in Chapter IV and Chapter V, when I adopt the agent definition of Jefferson, et al. (1991). These agents will be instantiated as neural networks together with functions relating the state of the environment to the network's input, and the network's output to the agent's actions.
2.2.2 The Importance of the Environment

In a still oft-quoted parable, Simon (1969) describes the importance of the environment in artificial intelligence. The lesson remains so important today that it merits repeating here. Consider an ant walking along a beach towards some goal. As the ant encounters rocks or large grains of sand, its path will veer to the left or the right in a seemingly erratic way. To an outside observer, the ant’s path looks quite complicated; but the ant itself is simply following the most direct route towards its goal, avoiding obstacles as necessary. In this way the complexity of the ant’s behavior arises from the complexity inherent in the environment.

Adaptive behavior stresses the importance of the environment as well, but with a slightly different emphasis. Complexity arises neither from the agent, nor from the environment, but from the interaction of the two.

2.2.3 Adaptation as Structural Coupling

As with “agent” and “environment,” “adaptation” is a broad term, encompassing changes ranging in time scale from milliseconds to millennia, and behaviors ranging from reflexes to learning to evolution.

As I described briefly in Section 1.4, adaptive behavior views changes in an agent as structural coupling, a term adopted from Maturana and Varela (1980). Winograd and Flores (1986) relate that work from an AI perspective, and first describe how the structure of an agent relates to the environment (or, in their language, “medium”):

The structure of the organism at any moment determines a domain of perturbations — a space of possible effects the medium could have on the sequence of structural states that it could follow. The medium selects
among these patterns, but does not generate the set of possibilities. In understanding an organism as a structure-determined system we view it in terms of its components and the interactions among them (Winograd and Flores, 1986, pp. 43-44).

The possible structural states, and the consequent behaviors, are determined by what they call selection:

The mechanism by which an organism comes to function adequately in its medium is one of selection, which includes both the selection of structural changes within an individual and the selection of individuals by the possibilities of survival and disintegration. A plastic, structure-determined system (i.e., one whose structure can change over time while its entity remains)... will by necessity evolve in such a way that its activities are properly coupled to its medium. Its structure must change so that it generates appropriate changes of state triggered by specific perturbing changes in its medium; otherwise it will disintegrate (Winograd and Flores, 1986, p. 45).

This type of change in structure, i.e., one which allows an agent to perform well in its environment, is what Beer referred to as "structural congruency" (Section 1.4). In my work below, structural coupling will be achieved through changes in the neural network which defines an agent (both in terms of structure and weight values), and selection will be achieved by either backpropagation or evolutionary programming (search techniques loosely modeled after Hebbian learning and evolution, respectively).

2.2.4 A Philosophy on Intelligence

The study of adaptive behavior may seem focused on exploring the ways in which an agent can adapt to its environment, but the field's loftier goal is an understanding of intelligence through such studies. The bottom-up philosophy of this approach finds support from many different researchers (e.g., Brooks, 1986, 1991a; Wilson, 1987, 1991; Agre, 1993; Dennett, 1987, Cliff, 1991; Meyer and Guillot, 1991; Maes, 1993; Braitenberg, 1984), but no one
provides the clarity and completeness of Beer (Beer, 1990; Beer, 1992; Beer and Gallagher, 1992; Beer, 1993).

Beer's stance on adaptive behavior grew from his work on computational neuroethology, "the computer modeling of the neural control of behavior in simpler whole animals" (Beer, 1990, p. 17). Using the cockroach as motivation, Beer shows how seemingly high-level behavior (e.g., goal following while responding to changes in the environment) can be built up from a simple, network-based agent. In later work, Beer (1992, 1993) adopts a dynamical-systems perspective to interpret his evolved agents. The agent's behavior can be explained as a dynamical system; the neural network is simply one instantiation of the system. Similarly, the environment can be modeled as a dynamical system. The adaptation which occurs through evolution is best viewed as a "coupling" between the structure of the agent and the structure of the environment, as described in the previous section.

In many ways, the study of adaptive behavior mirrors the study of artificial intelligence, not just in its goals, but also in the path of the field towards maturity. As with the early history of AI, adaptive behavior currently both enjoys and suffers from being an experimental science (Beer, 1993). Experimentation provides a large degree of freedom, allowing researchers to explore many alternative mechanisms for adaptation. But freedom can also be viewed as a lack of discipline, and without efforts to the contrary, the "field will become a cacophony, with as many voices and languages as there are individual researchers" (Beer, 1993, p. 2).

My personal view is that the study of adaptive behavior is not radically different from the study of artificial intelligence; in fact, the emphasis on adaptation is but one means
towards creating an understanding of devices with cooperative behaviors. That means adaptation has been underutilized in the study of communication; my goal in this dissertation is to explore whether the reason is because adaptive methods are incapable of evolving communication, or because the techniques have not yet been tried.

2.3 Reactive planning

In Section 2.2.2, I described Simon's parable about the importance of the environment. Seventeen years passed, however, before his lesson found its way into a working AI system. Agre and Chapman (1986) developed Pengi, a system which controlled the simulated penguin in a video game where the penguin must avoid aggressive bees. Instead of viewing planning as a purely cerebral activity, Agre and Chapman emphasized the visceral aspect of existing in an environment. In a fast-changing world, an agent does possess the time required to contemplate all implications of an action. Instead, it must simply react to a given set of inputs. Reactive planning, as the idea came to be called, demonstrated an alternative to complex planning: responding to a dynamic environment via perception-mediated reactions.

Part of Pengi's relevance to this work lies in the type of environment defined by the system. Only one agent, the penguin, actively learns during the game; the behavior of the other agents (the bees) is fixed. The world changes constantly regardless of the penguin's actions; therefore, the environment is dynamic. However, because the bees do not change over time, the environment's dynamics are easily predictable. In contrast, in the evolution of communication, all agents change over time – a much more difficult environment, as we will see in Chapter V.
2.4 Behavior-Based Approaches to Adaptive Behavior

Reactive planning has strong roots in AI: it grew out of a dissatisfaction with STRIPS-like planners (Agre, 1993). The next approach to constructing agents, however, has its genesis in engineering, in particular, in the design of autonomous robots.

2.4.1 The Design of Behavior-Based Agents

In a seminal work, Brooks (1986) introduced a new method for designing intelligent agents. Before that time, most work in robotics assumed a functional decomposition, in which an agent was constructed by modules, each capable of performing a particular function (Figure 3). This leads to a problem for AI; namely, that of dubious origin of the modules. They do not arise from task constraints directly, but from the designer’s analysis of the task.

Brooks proposed instead a behavioral decomposition as an alternative design methodology. For his domain of mobile robot control, he considered not the functionality of a agent’s subsystems, but the various behaviors needed by the agent, from very simple “avoid crashing into objects” to “reasoning” (Figure 4). The behaviors form a hierarchy in that higher-level abilities require lower-level ones for successful execution. For example,
unless an agent possesses the ability to avoid crashing into other objects, it would be better off remaining still than attempting to explore its environment.

Next, Brooks considered the question of how to map from the behavioral constraints of Figure 4 into an actual robot implementation. The result was *subsumption*, an architecture consisting of layers of modules, each capable of subsuming the behaviors of the layers below. For instance, Figure 5 shows an abstract subsumptive architecture for the behaviors "avoid objects," "wander," and "explore." The sensors feed into the three modules, each of which is responsible for performing a particular behavior. Each module possesses the ability to affect the actuators, but the actual output depends upon the "assertiveness" of each module, which in turn depends upon the input.

Brooks implemented each module as an FSA. Conflicts among modules are resolved via two primitive communication operators. *Suppression*, replaces the inputs to a
module with a value determined by the suppressor. Similarly, inhibition overrides the outputs of a module.

For a concrete example, consider the system in Figure 6, an expansion of the abstract version of subsumption given above. Once again, there are three behaviors: “avoid,” “wander,” and “explore,” each implemented by a separate layer. Ignoring layers 1 and 2 for the moment (as well as connections to/from the higher layers), consider how “avoid” is accomplished. The robot receives input from its environment, in terms of a sonar picture which relates the presence of obstacles. If the module detects an imminent collision, the “collide” node becomes active, sending a signal which halts the robot. Otherwise, the “feelforce” node becomes active, calculating the location of nearest object, and sending a command to “runaway.” This node sends a turn command to the robot, effecting a move away from the obstacle.
Figure 6: A specific behavioral decomposition (Brooks, 1986).
Methodologically, layer 0 is designed, debugged, and implemented before consideration of any other behaviors. Layer 1 is built on top of this as follows. "Wander" generates a random heading, and feeds it to the "avoid" node. "Avoid" takes as input not only this desired heading, but also the result of "feelforce," which was the direction required to avoid the nearest obstacle. "Avoid" incorporates the two in calculating a heading which not only avoids obstacles, but also points in a random direction. The result suppresses the inputs to the "turn" node, i.e., it overrides any previous inputs along that line. As a result, the robot moves off in the random direction.

Note, however, that if layer 0 detects an imminent collision, "collide" still sends a halt signal to the robot, preventing any damage. In this way, layer 0 can usurp control from layer 1 if necessary. More precisely, the system \{layer 0, layer 1\} subsumes the behavior of layer 0 alone, so that the robot wanders around the environment, but still avoids any obstacles. Layer 2, capturing the ability to explore the environment in a directed sense, is built upon the previous layers in a similar, though slightly more complicated, manner.

The ideas embodied by subsumption far exceed just the architecture itself. In particular, the methodology emphasizes:

- systems embedded in the real world
- a focus on behaviors rather than functions
- modeling a single layer of competence in toto before adding additional behaviors
- limited interaction between layers, through subsumption and inhibition
- an assumption of limited (or no) representation
- a distributed approach to intelligence
Embedding the systems in the real world reflects Brook's engineering background. To his credit, he and his group have designed and implemented a wide range of robots, serving as a strict test of the utility of subsumption. The emphasis on behaviors, in particular the way in which the behaviors are modeled sequentially, further serves to test the outcome of their efforts. The interactions between layers, though limited to just two operations, are more complex than they first seem. As may have been noticed in Figure 6, there is no internal state. The emphasis in subsumption is similar to that of reactive planning: the environment contains enough information to push the agent around in an intelligent way. Finally, also noticeable in Figure 6, the subsumption treats intelligent behavior as the interaction of several distributed modules.

Because it encompasses so many unique ideas, Brook's methodology has recently received much attention from the AI community (e.g., see Brooks, 1991a; Kirsch, 1991). These ideas are relevant to my work because subsumption was the first method I considered for evolving communication among agents. I rejected it, however, for reasons which will be detailed in Chapter III. For now, though, I point out only the major one: there has been only limited success at getting agents to learn with this architecture. The next subsection describes some of these efforts.

2.4.2 The Learning of Behaviors

Maes (Maes and Brooks, 1990; Maes, 1990; Maes, 1991a; Maes, 1991b; Maes, 1992) focuses on the question of learning within the behavior-based framework. She assumes that an agent is represented by a set of modules \( \{M_0, M_1, \ldots\} \), where each \( M_i \) contains:

- a set of preconditions which must be true for the module to become active
Figure 7: Learning to produce the correct behaviors (from Maes, 1991a).

- a set of expected effects if the module is activated
- an activation level

Three types of relationships may hold between modules. A "successor" link between modules $i$ and $j$ denotes that $M_j$ should become activated after $M_i$. A "predecessor" link denotes just the opposite. A "conflictor" link denotes two modules with conflicting preconditions.

For instance, a set of modules to achieve a high-level goal such as "relieve thirst" might be "pour liquid" into a cup, "pickup cup," "drink," etc., as shown in Figure 7. Given a set of data about the environment, activation flows forward throughout the behavioral modules. Simultaneously, activation flows backward from any active goals. The resulting spreading activation finds a path linking the two, allowing the agent to achieve its goals.
Maes’ algorithm holds interest when viewed from the context of action selection, which typically assumes a strict hierarchy of behaviors (e.g., Tinbergen, 1951). Spreading activation allows non-hierarchical action selection. It has been criticized, however, because it is particularly sensitive to parameter settings (Tyrrell, 1993a, 1993b, 1993c).2

From the perspective of attempting to evolve communication, the difficulty with Maes’ approach, and other action selection approaches, is that they assume a detailed high-level set of behavioral modules. This creates a problem of origin: how are such behaviors selected? Moreover, from the standpoint of evolving communication, the behaviors should not be given a position of primacy; rather, they should be coevolved with the system of communication.

2.5 The Design of Group Behavior

Brooks’ work described above deals with the creation of a single agent. When multiple agents exist in the same environment, the world suddenly become more interesting. Agents can cooperate on a single task, or compete for limited resources, or operate in separate niches with little or no interaction.

Given these differences between single and multi-agent environments, it comes as no surprise that a large portion of work focuses on the design of group behavior. Interestingly, the emphasis on design mirrors the early history of connectionism: when

---

2. Tyrrell has studied several different action selection mechanisms (Tyrrell 1993a; Tyrrell and Mayhew, 1991), and released the code for his simulation environment so that others may use it to evaluate different action selection mechanisms. However the documentation unknowingly relates the motivation for looking at methods of evolving agents. The first sentence of the section entitled “How to Design Your Own Action Selection Mechanism” states: “You should think twice about doing this since it will probably involve a large amount of time and effort.”
faced with the novelty of nodes and links, researchers first showed explored the computational power of their architecture by designing networks to perform tasks such as word recognition (e.g., McClelland and Rumelhart, 1981). The analogy should be kept in mind in this section, where I review efforts which explore the computational abilities of groups of agents.

From the point of view of evolving communication among agents, the same criticism applies to all the work presented in this section; namely, that all the systems suffer from too strong a reliance on human design, and consequently shed little light on how communication schemes come into being.

2.5.1 Behavior-Based AI, Revisited

Although Brooks' work above has focused on a single agent, there is nothing to prevent the application of subsumption to groups of agents. In fact, Brooks himself has suggested exactly this (Brooks and Flynn, 1989; Brooks et al., 1990). He proposes herds of small, homogeneous robots. Collectively, they would have the power to explore a large area (of another planet, for instance), but individually, each would be cheap and expendable.

From a less philosophical viewpoint, Parker (1993) offers an implementation of a multi-agent system built from the subsumption architecture. She begins with a subsumptive architecture, similar to Figure 6. This provides a set of behaviors for an agent. But because an agent should possess multiple sets of behaviors, Parker extends the base architecture with other behavior sets (Figure 8). At any one time, only one behavioral set is active. In this way, agents can modulate each other's behavior. The problem with this approach, as with original subsumption, is that each module is fully designed.
Figure 8: Sets of subsumptive modules (adapted from Parker, 1993). Each module is a subsumptive system, similar to that of Figure 6. Note the similarity to the connectionist methods of capturing multiple behaviors (Pollack, 1987). In this case, however, the gating is absolute, by design. In the connectionist system, gating is trained, with the goal of making it absolute.
2.5.2 Colonies

Several researchers, finding motivation from the field of ethology, have studied insect-like colonies of agents. This section describes a representative sample of the work.

Deneubourg, et al., (1991) show how a group of simple ant-like agents can work together to collect food from an environment. Initially, the environment contains a collection of randomly-distributed food (Figure 9a). Each agent (not shown in figure) wanders around the environment randomly, searching for food. If food is discovered, the agent picks it up with probability \( p \), determined by the density of food in the agent's area. If the density of the food is low, there is a high probability the agent will pick up the piece encountered; if the density is high, the probability is low. Once an agent has a piece of food, it continues to wander around the environment, dropping the food with probability \( q \).
time, however, $q$ varies directly with the density of the food in an area, so that each agent tends to move isolated food into piles. A sample result is shown in Figure 9d.

Colomi, Dorigo, and Maniezzo (1992) show how ant-like agents can perform the more traditional Traveling Salesman Problem (TSP). Given a collection of agents and a set of cities, the agents begin wandering from city-to-city randomly, dropping pheromone trails as they go. Each agent’s path is determined probabilistically, in part by the density of pheromone along the potential trails. Agents follow the paths with the strongest pheromone deposits; because inefficient paths require more time to traverse, the pheromones will disperse there more quickly. Over time, the agents learn to traverse the cities in a very efficient manner.

Goss and Deneubourg (1992) focus more on communication between agents. They propose a communication scheme which allows a group of agents to search a large area despite severe limitations on the signal ranges of agents. In the center of the environment is a base which emits a homing signal. The agents spread out from the base, staying within range of the beacon. When a particular agent reaches the outer limits of the beacon’s signal, it stops, and then rebroadcasts the signal so that other agents may hear it. The other agents, using the retransmitted signal for direction, can safely move out of range of the original beacon.

These works demonstrate the benefits and dangers of multi-agent systems. A large number of agents, each following relatively simple strategies, can quickly perform a given task. But all of these schemes rely on clever design, and more often reflect the skill of the programmer than any aspect of how groups of agents can learn to interact in an intelligent manner.
Table 1: The basic interaction primitives (from Mataric, 1993a, p. 435).

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collision Avoidance</td>
<td>the ability of an agent to avoid colliding with anything in the world...</td>
</tr>
<tr>
<td>Following</td>
<td>the ability to stay behind or along side of another agent without colliding</td>
</tr>
<tr>
<td>Dispersion</td>
<td>the ability of a group of agents to spread out over an area in order to establish and locally maintain some predetermined separation</td>
</tr>
<tr>
<td>Aggregation</td>
<td>the ability of a group of agents to gather in order to establish and maintain some predetermined distance...</td>
</tr>
<tr>
<td>Homing</td>
<td>the ability of one of a group of agents to reach a goal region or location</td>
</tr>
<tr>
<td>Flocking</td>
<td>the ability of a group of agents to move as a coherent aggregate without prespecified leaders and followers. Flocking includes components of collision avoidance, following, dispersion, and aggregation.</td>
</tr>
</tbody>
</table>

2.5.3 Designing Emergent Behaviors

Finally, two last researchers have studied group behaviors, but their efforts extend beyond a single task. Both share an emphasis on the emergence (Forrest, 1991a, 1991b) of group behavior from local interactions.

Mataric (1993a) adopts the goal of trying to determine how simple, local interactions among agents can lead to complex, purposive group behavior. She develops a set of basic interaction primitives, reproduced in Table 1. Mataric implements each of these primitives using the subsumption architecture (Section 2.4.1), and then combines them to achieve more complex behavior. For instance, Figure 10 shows how foraging, the ability of a group of agents to collect food and return it to a centralized location, is accomplished through the other primitives.
Two main problems limit the applicability of Mataric’s work to the evolution of communication. First, she does not address the issue of “communication primitives;” her agents do not communicate directly. Second, the criteria for determining a “primitive” interaction behavior is unclear. Mataric’s primitives are not atomic behaviors; instead, a primitive is “any group behavior constructed with simple local interactions that can be used as a building block” (Mataric, 1993a, p. 434), which implies that primitives are highly subjective, and somewhat task-dependent. In fact, determining interactions which hold across different tasks is part of Mataric’s current research effort: “Our continuing work attempts to formalize what basic interactions are necessary for achieving particular types of collective behavior” (Mataric, 1993a, p. 434).
Figure 11: Vector field for the avoid-static-obstacle behavior (from Arkin, 1993).

Arkin (1989, 1992a, 1992b, 1993) has done extensive work on the issues of group behavior, with and without direct communication between agents. His main idea is to develop a set of motor schemas, each of which captures a particular robotic behavior. In any given situation, each motor schema generates its particular reaction in the form of a vector. The reactions from the agent's set of motor schemas is summed, and the resultant vector is used to determine the response of the agent. For instance, one particular motor schema is to avoid-static-obstacles:

A repulsive disk encircles any detected obstacle. The repulsive force drops off linearly out to a certain distance from the robot (the sphere of influence). The repulsion is infinitely high within the diameter of the obstacle itself. The gravitational analog can be viewed as a mountain whose slopes depend on the certainty of detection. (Arkin, 1992a, p. 205.)

Figure 11 shows a graphical representation of this motor schema, a repulsive vector field centered on the obstacle. When placed anywhere in an environment, this vector field gives the output of the avoid-static-obstacle motor schema.
Arkin has developed motor schemas for two-dimensional (Arkin, 1989) and three-dimensional (Arkin, 1992a) navigation. From our point of view, however, the more interesting work is his efforts focusing on communication. Using motor schemas as a base, Arkin has studied the foraging task. In this work, an agent can be in one of three behavioral states: \textit{forage} (searching for food), \textit{acquire} (moving towards observed food), and \textit{deliver} (carrying food to a base). Arkin (1992b) demonstrates that even without direct communication, agents can cooperate.

Arkin, Balch, and Nitz (1993) add communication in the following way. Each agent writes its behavioral state (forage, acquire, deliver) to a blackboard. An agent foraging for food has the option of reading the blackboard, and if it finds another agent “acquiring” or “delivering,” it can move towards that agent to assist. Arkin shows that even this simple model of communication increases task performance.

Arkin and Hobbs (1993) stress the importance of communication in multi-agent systems, and they lay out an agenda of issues to explore, including allowing varying communication signals to express various communicative needs. I strongly agree with the goal, but the approach I advocate below differs significantly from their method due to my goal of understanding how it could come about.

\textbf{2.6 The Evolution of Communication}

The previous section presented a range of designed multi-agent systems, both for specific tasks like TSP, and for more basic interactions and communication. Although many studies address various aspects involved in the evolution of multiple agents (e.g., Theraulaz, et al., 1991; Grefenstette, 1992; Assad and Packard, 1992; Beni and Hackwood, 1992; Meuleau,
1992; Axelrod, 1984; Hillis, 1990), relatively few studies address the evolution of communication among multiple agents. In this section, I present the existing work on the evolution of communication among agents. Because of its strong relevance to my work below, I present this work in detail. Although each of these studies appear very different at the surface, a common thread of discrete communication binds them together, about which I will say more later.

2.6.1 Adaptive Communication Protocols

Yanco and Stein (1993) investigate a simple “follow-the-leader” task in which one agent, the leader, receives a command which must be followed by both the leader and a second agent (Figure 12). Commands are given in pairs \((a_0, a_1)\), where \(a_i\), chosen from a set of potential actions \(\{A_1, ..., A_n\}\), indicates the desired action of agent \(i\). The leader chooses one of \(n\) symbols to represent the command, sends this symbol to the other agent, and both the...
leader and the follower execute an action. The agents receive positive reinforcement only when both agents execute the proper action. Initially, the encodings of the leader and responses of the follower are random. Under a standard reinforcement algorithm, a consensus emerges between the two over time, in that agent 0 and agent 1 come to agree on a common interpretation of the communication signals.

To give a concrete example, Yanco and Stein’s initial experiment involves only two possible agent actions, spin or straight, and a language of only two symbols, low and high. Commands are (spin, spin) or (straight, straight), i.e., the follower should mimic the action of the leader always. This case is quite simple, and accordingly, the agents evolve a common language in ~15 iterations, on average. Note that two dialects are possible: \{low \rightarrow spin; high \rightarrow straight\} and \{low \rightarrow straight; high \rightarrow spin\}. As expected, the reinforcement algorithm evolves both dialects equally often.

Yanco and Stein explore variations on this theme, adding another agent\(^3\), and investigating languages with between 2 and 20 symbols \(n \in \{2, ..., 20\}\). Note that because the leader sends only a single symbol to its subordinates, the size of the language (i.e., then number of unique encodings) is equal to the number of symbols. Furthermore, note that the number of possible actions in each experiment is equivalent to the size of the language. Thus the languages that emerge are simply one-to-one correspondences between actions and symbols. Yanco and Stein themselves recognize the limitations of such a scheme (p. 483), and suggest the study of compositional languages (i.e., more than one symbol per word) as future work.

---

3. Although not clear from their paper, it appears that for three agents, the leader still receives a command as a pair \((a_0, a_1)\), where now \(a_0\) represents a desired action for the leader, and \(a_1\) represents a desired action for both subordinates.
2.6.2 Evolving Communication Between the Sexes

Werner and Dyer (1992) describe a more complex task in which simulated animals must learn to communicate to find mates. They populate their environment, a 200 x 200 grid, with male and female agents. Females have no capacity for movement; after being initialized randomly, each remains in a fixed grid position, with no particular orientation. In contrast, males possess an orientation (north, south, east, or west), and wander about the environment using four primitive actions: move-forward, turn-left, turn-right, and stand-still. When a male encounters a female (i.e., when a male enters a grid position occupied by a female), they "mate," producing two offspring (male and female) which are added to the current population. To keep the number of agents constant, two randomly-selected agents (male and female) are removed from the population whenever mating occurs.

Each sex possesses only limited abilities to sense its environment, necessitating communication between males and females for efficient mating to occur. Each female can sense potential mates within a 5 x 5 square centered at the female’s fixed location, and output an n-bit signal to all males within her square. Males lack the capacity to produce signals or see the females directly, but they can sense a female's signal (Figure 13). This unequal distribution of information creates a condition which fosters communication: females can "shout out" to males with their signals, and males can respond by moving towards the females. However, the agents possess no such ability initially; the paper focuses on how such a system of communication evolves.

Both males and females share a recurrent neural-network representation, although the difference in functionality between the sexes is accompanied by an architectural difference as well. Males possess n input units, used to represent the signal of a nearby
Figure 13: Werner and Dyer's (1992) scenario for evolving communication. A 200 x 200 toroidal grid is populated with 800 male and 800 female agents. Males wander around the environment, and although they cannot "see" females, they can "hear" their signals. Each female is stationary, but can "see" males in a 5 x 5 square centered at her location, and output a 2 or 3-bit signal (depending on the experiment) to attempt to communicate her location to the male.

female, and 4 output units, one for each of the male's potential actions. Females have 100 input units, one for each <location, orientation> pair within her 5 x 5 visual field, and $n$ output units, used to send an $n$-bit signal to all males within her visual field. Both males and females possess $h$ fully-connected hidden units. For example, in Figure 13, the female $F$ can sense the male $M_1$ at relative location (1, -2), activating her input unit for $<(1, -2), \text{north}>$. Her output signal, "010," registers on $M_1$'s input units.

In their first experiment, Werner and Dyer set $n=3$ and use a genetic algorithm to search the space of networks, and find that, as expected, the sexes learn to agree on a
common language. Females signal nearby males which actions to take to move closer, and males respond appropriately.

In a second set of experiments, they set $n=2$ and $h=0$, i.e., communication involves only 2 bits, and networks have neither hidden units nor recurrence. Once again, the GA evolves males and females which communicate appropriately, but in all their runs, a single communication protocol always emerges. In other words, initially one female might output "01" when a male is just to her right facing east, and another might output "11" in this same situation. Similarly, one male might respond to "11" by turning right, while another responds by moving forward. After evolution, however, all males and females are behaviorally equivalent.

Werner and Dyer provide a good start for the evolution of communication. They do not assume any particular communication protocol (i.e., they do not dictate that an male hearing "00" should move forward, etc.); and their network representation resides at a good level for evolving agents (more on this in Chapter III).

However, there are several problems with this work. First, the architectural distinction built into males and females seems suspect. We would like any two agents in the environment to be able to communicate potentially, not just ones of different sexes. Second, the method by which agents sense one another seems wrong. If two males enter a female's field of vision, she senses only the closest one. If a male lies within range of two females, he hears only the closest one. We would rather have the agents decide for themselves to which other agent they should attend. Finally, the number of potential signals between

---

4. Once again, however, they are not particularly clear about the details of their networks, and $h$, the number of hidden units, is not given for this experiment.
agents seems exceedingly small: 8 in the first experiment, and 4 in the second set. This leads to a small number of potential agents (e.g., 256 different males in the second experiment), and is responsible for the fact that a single communication protocol always emerged to dominate all others.

2.6.3 BioLand

Werner and Dyer themselves recognize some of the problem I have pointed out, and have proposed a more interesting model of communication (Werner and Dyer, 1993). Their environment, "BioLand," consists of not just males and females, but males and females of two types (wolf and deer), along with various other objects such as plants. Each object emits a specific sound or smell which diffuses throughout the environment (unlike the method of sensing described above). In addition, animate objects can produce voluntary sounds of varying frequency and loudness, providing the basis for communication.

BioLand is staggering not just for the types of objects it contains, but also for its size: 8000 predators and 8000 prey on a 1000 x 1000 toroidal grid. Unfortunately, this prodigious size also impedes analysis. Currently, Werner and Dyer are focusing on the evolution of herding behavior (e.g., how groups of deer learn to move together), and communication seems to have become secondary. Although I expect further emphasis on communication as their research progresses, their current work does not provide any details regarding how the signals generated by the agents affect their behavior.

2.6.4 Synthetic Ethology

In contrast to the detailed model of BioLand, MacLennan (1992) adopts a very high-level view of language. He defines an abstract task in which a group of agents must learn to
communicate. Each agent possesses local information in terms of one of \( n \) symbols; it chooses a second symbol (from a set of \( n \)) to convey that information, and other agents must respond appropriately. For example, in Figure 14, agent \( A_0 \)'s local environment contains the symbol "3"; \( A_0 \) responds by writing a "2" to the global environment. The goal is for the other seven agents to interpret the "2" as a "3."

MacLennan describes his approach as *synthetic ethology* (MacLennan, 1992; MacLennan and Burghardt, 1993), an attempt to simulate the behavior of organisms within artificial worlds. Although different in name, synthetic ethology is very close to the study of intelligence though adaptive behavior, with a similar emphasis on agents existing in an environment (as described in Section 2.2.).

Using finite state machines to represent agents and a genetic algorithm for search, MacLennan shows how the group of agents evolve a common symbol-symbol mapping.
This result is very similar to Werner and Dyer’s evolution of communication between the sexes (Section 2.6.2), and although MacLennan’s task is seems very different, the same critique applies; namely, that the number of potential messages between agents is too restricted. One agent must recognize one of eight symbols; encode it using one of eight symbols; and other agents must learn to respond appropriately. I will say more about the discrete nature of this task below.

2.6.5 AntFarm

Collins and Jefferson (1991, 1992) study AntFarm, an environment not so abstract as MacLennan’s, yet not so detailed BioLand. In AntFarm, a group of simulated ants must learn to cooperatively forage for food, i.e., search the environment, and return any food found to a centralized nest. The colony lives on a 16 x 16 toroidal grid. At each time step, an ant drops between 0 and 64 units of pheromone, which then diffuses throughout the environment as a signal to other ants (Figure 15).

Figure 15: An 8 x 8 subset of the 16 x 16 AntFarm environment. Although pheromone diffuses throughout the environment, this figure (taken from Jefferson and Collins, 1992) shows all pheromone as equivalent.
Ants are represented as neural networks; the semantics of an ant’s input and output nodes are shown in Figure 16. Each ant can sense the presence of food in a 3 x 3 square centered at the ant’s current location. Similarly, each ant can sense the presence of the nest or the amount of pheromone in this same area. Because the task is to return food to the nest, each ant also possesses an input which indicates whether it is carrying food. Finally, the “nest direction” input gives the direction of the nest (north, south, east, or west). Outputs indicate the direction to move, whether to pickup or drop food, and the amount of pheromone to release (0-64 units).

Collins and Jefferson confess an inability to evolve cooperative foraging among ants, and explore several different network architectures to find the problem. Their best candidate is a dual representation scheme: each ant possesses two of the networks depicted in Figure 16. The first is automatically invoked when the ant is searching for food; the second is automatically invoked when the ant is carrying food. Collins and Jefferson explain their motivation for this scheme as follows:

...the problem with [the single network] encodings was that they have difficulty evolving discrete behavior (where a small change in the inputs leads to a large change in behavior). These representations “generalize,” so small changes in the inputs are smoothed away, making the evolution of discrete behavior unlikely (Collins and Jefferson, 1992, p. 593).

In Chapter III, I will argue that Collins and Jefferson have things exactly backwards: rather than attempting to completely insulate one network from the other, they should be looking at ways for one network to influence the other in beneficial ways. Furthermore, I will present an architecture which allows exactly this.
Figure 16: One of the various network architectures used by the agents of AntFarm (Collins and Jefferson, 1992).

2.7 Remarks

This chapter has covered a wide range of work. I began by discussing adaptive behavior's emphasis on agents, environments, adaptation. I then discussed the design of a single agent, the design of group behavior, and, finally, the evolution of communication. I also briefly discussed adaptive behavior's strong bottom-up philosophy. In the words of one prominent member of field:

The basic strategy of the animat approach is to work toward higher levels of intelligence "from below" - using minimal ad hoc machinery (Wilson, 1991, p. 16).

The advice is sound, but in terms of communication, it is only partially heeded. As we saw above, most of the research involving multi-agent systems assumes a designed method of communication (Parker, 1993; Deneubourg, et al., 1991; Coloni, Dorigo, and Maniezzo, 1992; Goss and Deneubourg, 1992; Mataric, 1992; and Arkin and Hobbs, 1993).
Even those works which attempt to evolve communication often make strong assumptions. Yanco and Stein (1991) assumed a distinction between the message sender and receiver; Werner and Dyer (1991) made a similar architectural distinction between male and female; MacLennan (1992) assumed that at any given time, there is a privileged agent attempting to convey its local information to the others.

Furthermore, all the work at evolving communication assumes that agents transmit discrete signals, with ensuing finite-sized languages (2-20 for Yanco and Stein; 4-8 for Werner and Dyer; 8 for MacLennan; and 65 for Collins and Jefferson). This is a strong assumption about communication; namely, that for a given task, the size of the language needed to solve the task is known \textit{a priori}. Consequently, rather than assume the transmission of discrete signals between agents, I provide my agents with continuous channels capable of supporting a wide variety of communication schemes. Furthermore, I make no architectural distinctions between transmitter and receiver.

My first approach at evolving agents with continuous dynamics was to combine some of the studies above with a connectionist implementation; in particular, I explored a connectionist version of the subsumption architecture. While this proved useful as a way of adapting a single agent to its environment, it still assumed too much about the task, as I will describe next.

---

5. The size of a language is the number of distinct signals an agent may produce.
CHAPTER III

A CONNECTIONIST APPROACH TO ADAPTIVE BEHAVIOR

...how information is represented can greatly affect how easy it is to do different things with it.

- David Marr

3.1 Introduction

In this chapter, I present my first approach to adaptive behavior. Using Chandrasekaran and Josephson's multilevel perspective on artifacts as philosophical motivation, and Brooks' behavior-based approach to AI as an architectural motivation, I show how to incorporate multiple behaviors within a connectionist framework. Marr's quote, obvious though it may seem, helps me avoid an implementation problem with subsumption, as I will describe below. My particular agent, Addam, is comprised of a set of connectionist networks, a substrate which promotes the automatic design of subsumptive systems. Moreover, the implementational choice has important behavioral consequences - some complex behaviors emerge due to interactions among networks and need not be specified explicitly. In this way, the underlying layers leak into one another, each affecting the others in subtle and desirable ways.

In this section I present the background for my work. First I describe the philosophy of leaky levels, and then both connectionist and behavior-based approaches to achieving multiple behavioral constraints. Finally, I preview my alternate approach.

3.1.1 Leaky Levels

Historically, AI has viewed agents from the Knowledge Level (Newell, 1982), in which an individual is characterized by its knowledge, goals, and rationality. The abstract nature of this level has been called into question from many different directions: e.g., connectionism (Hinton et al., 1986; McClelland et al., 1986), situated action (compare Vera and Simon, 1993, with Agre, 1993), the observers’ paradox (Kolen and Pollack, 1993, to appear), and others (e.g., Searle, 1993). Most recently, those studying the simulation of adaptive behavior have stressed that intelligence should not be viewed simply as knowledge and goals held together with procedural glue; there is much to learn from studying intelligence through self-sufficient agents competent to exist in the world (Meyer and Guillot, 1991; Wilson, 1991).

Yet we often forget that agents can be viewed at multiple levels of description, and as Chandrasekaran and Josephson (1993) point out, there is no single level of description which captures all aspects of an agent’s behavior. To borrow their example, a simple coin sorter can be described as an abstract machine which classifies coins based on their weight and diameter, but if a lever jams, then the physical nature of the device becomes particularly important. Chandrasekaran and Josephson propose that agents be described by a set of "leaky levels," where each level of description contributes to the overall story of agent behavior, but the total picture arises due to the way the various levels interact.
This lesson is an important one, but it fails to address an closely related question: how does the recognition of multiple levels of description help one to implement an intelligent agent? In particular, how should one approach the task of constructing an agent which satisfies multiple behavioral constraints at different levels?

Before answering this question myself, I first present two very different methods, the first from the perspective of connectionism, the second from the perspective of engineering.

3.1.2 Connectionist Approaches to Multiple Behaviors

Many of the benefits of connectionism stem from a single property of feedforward networks: similar inputs map to similar outputs.\(^2\) This allows a network to induce a continuous mapping from a finite training set (interpolation of data), and creates a certain robustness in the face of noisy inputs. But this same continuity principle creates difficulties for researchers trying to capture multiple behaviors in a single network, for depending upon the desired behavior, we may want similar inputs to map to very different outputs.

Connectionist researchers remedy this shortcoming by appealing to a set of networks, mirroring the various behaviors of interest. The question of multiple behaviors is recast as one of modularity: how can modules be developed within a connectionist network so as to capture modularity of the input?

One genre of work (Jacobs, Jordan, and Barto, 1990; Jacobs, et al., 1991; Nowlan and Hinton, 1991) answers this question of modularity with the architecture of Figure 17.

\(^2\) More precisely, the mapping \( f \) embodied by a feedforward network is \textit{continuous}: given \( \varepsilon > 0, \exists \delta > 0 \) such that \( d(x, y) < \varepsilon \Rightarrow d(f(x), f(y)) < \delta \).
Figure 17: A connectionist approach to modularity. The input vector is fed to two experts, each of which captures a specific behavior. Simultaneously, a task bit is fed into a gating network, which selects which expert's output is appropriate for the task. Note the similarity to Parker's (1993) method of capturing multiple behaviors within the subsumptive framework (Figure 8).
For $n$ desired behaviors, the system consists of $n+1$ networks: $\{E_1, E_2, \ldots, E_n, G\}$, where each $E_i$ is an expert capturing one of the behaviors, and $G$ is a "gating network" which chooses the appropriate expert based on the current task.

There are several problems with this architecture. First, it suffers from a lack of distributed control: without the centralized gating network, the system would fail completely. Second, the experts themselves have no say as to when their actions are appropriate. Third, separating a "task bit" from the input vector permits different inputs to the experts and gating network, simplifying the modularization process by effectively preselecting which expert is appropriate. Finally, this architecture does not exploit the fact that the outputs of the gating network are continuous; instead, the interactions between modules are encouraged to be binary so that module $i$ has no appreciable influence on the output when module $j$ is active. In fact, Nowlan and Hinton (1991) focuses on explicitly training away these interactions.

3.1.3 Behavior-Based AI

Brooks (1986, 1991a) proposes an very different approach to modular behavior. Rather than observing a set of behavioral constraints and reasoning "The agent must have functional modules for perception, planning, etc.," one can remain more faithful to the actual observations by constructing an agent which satisfies the first behavioral constraint, and then incrementally adding layers of structure to satisfy the remaining constraints sequentially. This behavior-based stance removes a large bias on the part of the designer: modules arise from directly observable constraints on behavior rather than functional constraints implicit in the mind of the designer.
Unfortunately, Brooks does not go far enough. After performing a behavioral decomposition to define the functionality of a layer, he then proceeds to design a set of finite state automata to implement that layer (e.g., Figure 6, p. 24). Yet, this is precisely the type of functional decomposition he warns against (Brooks, 1991a, p. 146). One might appeal to learning to avoid performing this functional decomposition by hand, but current work in automating behavior-based design focuses instead on learning the interactions between preexisting behavioral modules (e.g., Maes, 1991a, Section 2.4.2).

The reliance upon designed modules arises from choosing FSAs as the level in which to implement subsumptive systems; in particular, from the arbitrary ways in which FSAs interact. Brooks achieves modularity through task-based decomposition of complex behavior into a set of simpler behaviors. In his system, for example, layer 0 implements obstacle avoidance and layer 1 controls wandering. Activity in layer 1 suppresses the activity of layer 0, and yet obstacles are still avoided because layer 1 subsumes the obstacle avoidance behavior of layer 0. In order to avoid duplication of lower layers as subparts of higher layers, he allows the higher layers to randomly access the internal components of any lower level FSAs. This fact, combined with the existence of multiple implementations for a given layer, forces us to question Brooks’ design methodology: development of single layer competence, freezing it, and then layering additional competencies on top of the first. If layer 0 can be realized equally well by method M1 or M2, then under Brooks’ methodology we will not know until layer 0 is fixed which methodology’s internal modules better facilitate the design of layer 1 (Figure 18).

Furthermore, Brooks fails to limit the suppression and/or inhibition which may occur between layers, so that a higher-level may randomly modify a lower-level’s
Layer 1, "simple"

Layer 1, "complex"

Layer 0, method $M_1$

Layer 0, method $M_2$

**Figure 18:** Subsumption's problem with sequential construction. If design of layer 1 is not begun until design of layer 0 is completed, then we will not know until layer 0 is fixed whether method $M_1$ or $M_2$ is the better choice.

computation. This unrestricted suppression/inhibition combined with the unrestricted access problem described above permit complicated interactions among layers. In Brooks' case, careful design keeps the interactions under control, and the resulting behavioral modules work well together. Without additional constraints, however, the space of subsumptive systems is far too large to search in a reasonable time.

**3.1.4 An Alternative Approach**

In this chapter, I present an alternative approach to subsumptive learning. Recognizing the multitude of formalisms with which to describe behaviors (Chandrasekaran and Josephson, 1993), I explore the merits and drawbacks of adopting a connectionist implementation for my layers. As will be discussed below, my version of subsumption replaces Brooks' FSAs with feedforward networks and additional circuitry, combined so that each module in a hierarchy respects the historical prerogatives of those below it, and only asserts its own control when confident. Given this basic architecture, I demonstrate how multiple behavioral constraints can be translated into network-level constraints. Finally, I discuss the
importance of the connectionist substrate for the implementation of leaky levels which produce emergent behavior in an agent.

### 3.2 Additive Adaptive Modules

This section presents my connectionist version of subsumption in detail. First, I describe the architecture itself, and then comment on how its relation to Brooks' method of combining behaviors.

#### 3.2.1 The Addam Architecture

My control architecture consists of a set of *Additive Adaptive Modules*, instantiated as *Addam*, an agent which lives in a world of ice, food, and blocks. To survive in this world, Addam possesses 3 sets of 4 (noisy) sensors distributed in the 4 canonical quadrants of the plane. The first set of sensors is tactile, the second olfactory, and the third visual (implemented as sonar that passes through transparent objects). Unlike other attempts at learning that focus on a single behavior such as walking (Beer and Gallagher, 1992), I chose to focus on the subsumptive interaction of several behaviors; hence, Addam's actuators are a level of abstraction above leg controllers (similar to Brooks, 1986). Thus, Addam moves by simply specifying $\delta x$ and $\delta y$.

---

3. Cliff (1991) makes a similar proposal from the context of computational neuroethology, but does not offer an implementation:

The intention then in closed-environment simulator computational neuroethology is to create a subsumption architecture where the modules forming each layer are not formal (symbolic) finite state automata, but small artificial neural networks. A network model constructed according to a subsumption architecture is an explicit recognition of the inhomogenous [sic] nature of natural nervous systems (Cliff, 1991, p. 36).
Internally, Addam consists of a set of feedforward connectionist networks, connected as shown in Figure 19. The 12 input lines come from Addam's sensors; the 2 output lines are fed into actuators which perform the desired movement ($\delta x, \delta y$). Note that we desire $\delta x, \delta y \in (-1, 1)$ so that Addam may move in the positive or negative direction. Initially, I implemented desired movement as a single scalar value, but this proved inadequate. It did not permit zero as a stable output as the network outputs tended to saturate with training. I then switched to a difference scheme in which the actual movement control was the difference between two outputs ($+\delta x$ and $-\delta x$). This configuration allows the system to learn and generate positive and negative movement, as well as no movement at all.

Addam controls its movements as follows. First, the 12 sensors are sampled and fed into layer 0, placing its suggestion for $\delta x$ and $\delta y$ on the output lines. Layer 1 combines these same 12 sensor readings with the sum squared output of layer 0, calculates its suggestions for $\delta x$ and $\delta y$, and adds these to the output lines. Layer 2 works similarly, and the final $\delta x$ and $\delta y$ values are translated automatically to motor controls which move Addam the desired amount and direction.

Note that I could have avoided feeding the sum-squared activation line into each module $M_i$ by gating the output of $M_i$ with the sum-squared line. I did not do this because my architecture is more general; gating can be learned as one of many behaviors by each $M_i$. My goal was to have each module decide for itself whether it should become active – had I used gating, this decision would have been made by $M_i$'s predecessors.
Figure 19: Addam's internal architecture. Each module possesses only limited information about the activity of its predecessors. Layer 1 receives only the sum-squared activation of layer 0, implemented as \( a_1 \). Similarly, layer 2 monitors the activity of its predecessors through a single input \( a_2 \). Through training, each layer learns to exert control only when relevant based on the current sensors, and when none of its predecessors is active.
3.2.2 Preemption

Instead of lumping Addam with other subsumptive systems, I prefer to identify my architecture as preemptive. The modules are prioritized such that the behaviors associated with the lower levels may take precedence over those associated with the higher levels. Prioritization is reflected both architecturally as well as functionally. Architecturally, a lower level provides its outputs to higher levels. Functionally, higher-level modules are trained to relinquish control if a lower-level module is active. For example, suppose that layer 0 behavior is to avoid predators, and layer 1 behavior is to seek out food. In the absence of any threatening agents, layer 0 would remain inactive and layer 1 would move Addam towards food. However if a predator suddenly appeared, layer 0 would usurp control from layer 1 and Addam would flee.

Earlier I criticized Brooks' method of subsumption for two of its freedoms: unrestricted access by one layer to another's internal state, and unrestricted modulation of a lower-layer's computation by suppression/inhibition from a higher-layer. Neither problem is present in Addam. A higher layer has access only to the sum-squared output of all previous layers, and any preemption of layer $i$ results from a single real value ($a_i$). This eliminates the methodological problem with sequential construction: the input $a_i$ to a layer depends only on what is computed below, not on how it is being computed.

A few more things should be noted about Addam's architecture. First, it has no internal state (or equivalently Addam's entire state is stored external to the agent in the environment, as in Simon, 1969). Second, a few of Addam's connections are fixed a priori. (The changeable connections are those in the boxes labelled layer 0, 1, and 2, above.) This
minimal structure is the skeleton required for preemption, but it does not assume any prewired behaviors.

Finally, I should acknowledge the similarity of Addam's internal structure to the cascade correlation architecture of Fahlman and Lebiere (1990). There are several important differences, however. First, my system is comprised of several cascaded modules instead of cascaded hidden units. Second, Fahlman and Lebiere's higher-level hidden units function as higher-level feature detectors and hence must receive input from all the preceding hidden units in the network. This can lead to a severe fan-in problem. Due to the preemptive nature of my architecture, higher-level modules need only know if any lower-level module is active, so they require only a single additional input measuring total activation of the previous modules. Third, Fahlman's system grows more hidden units over time, correlating each to the current error. The nodes of my architecture are fixed throughout training, so that modularity is not achieved by simply adding more units. Finally, there is a difference in training: Fahlman gives his network a single function to learn, whereas my system attempts to learn a series of more and more complex behaviors. (More on this below.)

3.3 Training Addam

As mentioned above, Addam's environment consists of three types of objects: ice, food, and blocks. Ice is transparent and odorless, and is hence detectable only by the tactile sensors. Blocks trigger both the tactile and visual sensors, and food emits an odor which diffuses throughout the environment and triggers the olfactory sensors. Addam eats (in one time step) whenever it comes into contact with a piece of food.
Addam's overall goal is to move towards food while avoiding the other obstacles. This makes training problematic – the desired response is a complex behavior indexed over many environmental configurations, and yet I do not wish to restrict the possible solutions by specifying an entire behavioral trajectory for a given situation. Beer and Gallagher (1992) attempted to solve this problem by using genetic algorithms, which respond to the agent's overall performance instead of to any particular movement. Initially, I take a different approach, namely, I train Addam on single moves for a given number of scenarios, defined as one particular environmental configuration. Under this methodology, the extended moves which define Addam's behavior emerge from the complex interactions of the adaptive modules and the environment.

3.3.1 Level 0 Competence

Training begins with level 0 competence, defined as the ability to avoid ice. The training scenarios are shown in Figure 20, along with the desired response for each scenario. Module 0 can successfully perform this behavior in about 600 epochs of backpropagation.
(adjusted so that the fixed +1/-1 connections remain constant), and the connections of this module are then frozen.

3.3.2 Higher-Level Competencies

I next train Addam on level 1 behavior, defined as the ability to move towards food, *assuming no ice is present*. Once again, training is problematic, because there are a combinatorial number of environmental configurations involving food and ice. I solve this problem as follows. First, I define 14 scenarios as above, but with food replacing ice. This defines a set $S$ of $\{(\text{SensorValues}, \text{MoveToFoodOutput})\}$ pairs. Note that this does not define a value for $a_I$, the activation of the system prior to module 1. (See Figure 19.) Instead of forcing module 1 to recognize the presence of ice, I assume that module 0 is doing its job, and that when ice is present $a_I$ will be $>> 0$. This allows us to define a training set $T$ for level 1 behavior by prepending the extreme values of $a_I$ to the SensorValues in $S$, thus doubling the number of configurations instead of having them grow exponentially:

$$T = \{ (0, \text{SensorValues}, \text{MoveToFoodOutput}),
            (1, \text{SensorValues}, \text{ZeroOutput}) \}$$

Thus layer 1 (which is initially always active) must learn to suppress its activity in cases where it is not appropriate.

After level 1 competence is achieved (about 3500 epochs), a training set for level 2 competence (avoid blocks) is obtained in a similar manner. Note again that this avoids the combinatorial explosion of specifying the many possible combinations of ice, food, and blocks. Level 2 competence is achieved in about 1000 epochs.
3.4 Results

This section explains my results, first in terms of Addam's overt behavior, and then in terms of the network activations which produce this behavior.

3.4.1 Performance

To measure Addam's performance when trained, I placed it in the complex environment of Figure 21a, where the small 1 dots trace out Addam's path. Each dot is one time step (defined as one application of the trained network to move one step), so the spacing indicates Addam's speed.

Addam begins at (3.5, 1) touching nothing, so its tactile sensors register zero and layer 0 is inactive. The olfactory sensors respond slightly to the weak odor gradient, causing a slight activation of layer 1, disabling the block-avoidance behavior of layer 2. Thus we observe a constant eastward drift, along with random north-south movements due to the noise inherent in the sensors. As Addam approaches the food, the odor gradient increases, the olfactory sensors become more and more active, and layer 1 responds more and more strongly. When the random noise becomes negligible at about (6.5, 1), Addam speeds up and reaches the food, which is consumed.

Subsequently, Addam detects the faint odor of another piece of nearby food, and once again layer 1 controls its movement. However, at about (9, 5.5) Addam's tactile sensors detect the presence of a piece of ice, activating layer 0, and usurping control from layer 1. In other words, Addam's aversion to ice overcomes its hunger, and it moves southeast. After "bouncing off" the ice, the tactile sensors return to zero, and layer 1 regains control, forcing Addam back towards the ice. This time it hits the ice just a little farther
Aversion to ice overcomes attraction to food when the tactile sensors detect the ice.

Figure 21: An example of preemption. (a) Addam’s emergent behavior in a complex environment. The dots in the upper figure trace Addam’s path as it moves through the environment in search of food. (b) Activity of each of Addam’s layers over time.
north than the last time, so that when it bounces off again, it has made some net progress towards the food. After several attempts, Addam successfully passes the ice and then moves directly towards the food.

To reach the third piece of food, Addam must navigate down a narrow corridor, demonstrating that its layer 1 behavior can override its layer 2 behavior of avoiding blocks (which would repel it from the corridor entrance). After finishing the last piece of food, Addam is left near a wall, although it is not in contact with it. Thus both the tactile and olfactory sensors output zero, so both layers 0 and 1 are inactive. This allows Addam's block avoidance behavior to become activated. The visual sensors respond to the open area to the north, so Addam slowly makes its way in that direction. When it reaches the middle of the enclosure, the visual sensors are balanced and Addam halts (except for small random movements based on the noise in the sensors).

### 3.4.2 Network Activations

Figure 21b shows how preemption occurs to create Addam's behavior. I graphed the activation of each layer $i$ of the system (where the activation of layer $i$ is $||\delta x,\delta y||$, the norm of layer $i$'s contribution to the output lines). $L_0$ is generally quiet, but becomes active between time $t=52$ and $t=64$ when Addam encounters an ice patch, and shows some slight activity around $t=140$ and $t=168$ when Addam's tactile sensors detect blocks. $L_1$ ("approach food" behavior) is active for most of the session except when preempted by the "avoid ice" behavior of $L_0$, as between $t=52$ and $t=64$. The 5 peaks in $L_1$'s activity correspond to Addam's proximity to the 5 pieces of food as it eat them; when the last piece of food is consumed at $t=164$, $L_1$'s activity begins to decay as the residual odor disperses. Finally, we see that $L_2$ ("avoid blocks" behavior) is preempted for almost the entire session.
It starts to show activity only at about t=160, when all the food is gone and Addam is away from any ice. The activity of this layer peaks at about t=190, and then decays to 0 as Addam reaches the center of its room and the visual sensors balance.

3.5 Remarks

Addam was trained on only 42 simple scenarios, yet it was able to perform well in a complex environment. This section evaluates how Addam achieves this result.

3.5.1 The Appeal of Preemption

Unlike other connectionist modular systems, my method of control is distributed – each module decides for itself whether it should exert control in any given situation. Furthermore, there is no gating network which receives a specialized task bit – Addam has three sets of sensors all treated equally and must learn the proper behavior on the basis of these inputs. Finally, instead of limiting activations of the modules to being 0 or 1, I exploited the underlying connectionist nature of my architecture, allowing us to produce interactions between modules more interesting than absolute preemption. For example, the presence of ice overrode Addam’s attraction to food, yet Addam’s “go-get-it” response to the food had a slight influence on its “runaway” response to the ice. Had preemption been absolute, Addam’s aversion to ice and attraction to food would have alternately controlled its movement, with layer 0 exactly countering the effect of layer 1, and Addam would have slowly starved to death as it bounced off the ice indefinitely.

The behavior of Chandrasekaran and Josephson’s coin sorter is best described by appealing to multiple levels of behavior (Chandrasekaran and Josephson, 1993). Addam is best described in a similar way. At one level, it is an agent which exhibits only three
behaviors: avoid ice, go to food, and avoid blocks. But the underlying connectionist levels leak through in the complex interactions of the modules (Figure 21). Had Addam been implemented as a set of FSAs, such complex behavior would not have emerged; it would have required explicit design.

This performance benefit of simplified subsumption is complemented by a benefit in training. As mentioned above, Brooksian FSAs are difficult to train because of the complicated ways in which they may interact. My connectionist networks, on the other hand, permit a host of training algorithms. In fact, the work of Beer and Gallagher (1992) or Maes and Brooks (1990) is really complementary to ours, for although Addam's modules were instantiated with feedforward networks trained by backpropagation, they could have just as easily been trained by either genetic or correlation algorithms. Moreover, feedforward networks need not have been used either. I could have substituted sequential cascaded networks (Pollack, 1987), endowing Addam with internal state (cf. Kirsh, 1991) and allowing even more complex behaviors.

My work also sheds light on the issue of neural network representations for agents. Collins and Jefferson (1991) explored such representations, but found them lacking because of their inability to shift behavior based on changing inputs. Preemption offers one way in which these shifts may be obtained.

Many of the problems of training behavior-based systems stem from the failure to recognize the multiplicity of levels in agents. I whole-heartedly agree with Brooks that the level of behaviors is particularly useful for the expression of design constraints. The level of FSAs may also be useful for refining the behavioral description. Yet, in the context of evolving agents, the network level is more appropriate. My connectionist approach
maintains the benefits of subsumption: a behavior-based view, incremental construction of
the agent, and distributed control. But, in addition to the performance and training benefits
described above, the neural network substrate offers a many-to-many mapping between
structure and behavior: a single module can affect multiple behaviors, and a single behavior
can arise from the interaction of multiple modules. Chandrasekaran and Josephson
proposed such leaking from a philosophical point of view; here I have shown how leaking
occurs naturally and aids performance in an evolved connectionist system.

3.5.2 Potential Drawbacks

Five drawbacks, or at least causes for concern, deserve attention. First, as with Brooks'
subsumption, I used a behavioral decomposition to define the number of modules for my
system. Second, I assumed a fixed network architecture for each module. Third, Addam’s
success partially depends upon finding scenarios representative of the desired composite
behavior, something I did myself. Fourth, Addam has no internal state. Finally, although
Addam’s world contains several different types of objects, it is a single-agent domain.

The first two problems arise from the way in which the structural modules were
defined. Addam’s approach, however, represents just one way of dealing with an unknown
structure. Elsewhere, I have explored with others how the structure of a module (i.e.,
number of hidden units and network connectivity) can arise from an evolutionary program
(Saunders, Angeline, and Pollack, 1994). I present this approach in the next chapter.
Furthermore, Angeline (1994) has explored how modularization can arise without a
behavioral decomposition, but I postpone the discussion of that work until the next chapter
as well, where it will be more appropriate.
I do not really view my last two drawbacks (scenarios, no internal state, and a single-agent domain) as problems; rather, I see them as necessary steps towards learning how to evolve agents in a dynamic environment. Addam represents my first approach to understanding intelligence through the study of adaptive agents and focuses on the potential of using traditional network training methods to evolve agents. Defining scenarios proved helpful here, as did limiting myself to a single agent with no internal state.

I will eliminate all these assumptions, though not all at once. In the next chapter, I do away with the behavioral decomposition, fixed network architecture, scenarios, and lack of internal state. This represents quite a step, so I still assume a single-agent environment. In fact, the environment of Chapter IV will be even simpler than Addam’s, as I will describe below. In Chapter V, however, when I attempt to evolve communication, I will of course eliminate the single-agent assumption as well.
CHAPTER IV

AN EVOLUTIONARY APPROACH TO ADAPTIVE BEHAVIOR

One thing that connectionist networks have in common with brains is that if you open them up and peer inside, all you see is a big pile of goo.

– Michael Mozer and Paul Smolensky

4.1 Introduction

A subsumptive system, such as that of Figure 6 in Chapter II, appears quite well-structured: each layer is responsible for a particular behavior, and within each layer the FSA which performs the behavior is well-understood. Addam, on the other hand, appears less structured: although the system is still neatly divided into modules, the means by which each module accomplishes its behavior is less obvious. In this chapter, I move even closer to Mozer and Smolensky's "goo," by eliminating the assumption of known structure common to both subsumption and preemption. Simultaneously, I eliminate the assumption of an a priori behavioral decomposition, for when faced with task $X$, both subsumption and preemption fail miserably if no such decomposition is forthcoming.

Once again, I will adopt neural networks as an agent representation, this time adding recurrence to allow internal state. I then introduce GNARL an evolutionary program which induces network-based agents. In contrast to constructive and destructive algorithms

for inducing structurally unconstrained networks, GNARL employs a population of networks and uses a fitness function’s unsupervised feedback to guide search through the search space. Annealing is used in generating both gaussian weight changes and structural modifications. Applying GNARL to a complex search and collection task demonstrates that the system is capable of inducing networks with complex internal dynamics, in particular, dynamics which perform well in the training environment. In this way, structural coupling is achieved.

4.2 The Network Induction Problem

In its complete form, network induction entails both parametric and structural learning (Barto, 1990), i.e., learning both weight values and an appropriate topology of nodes and links. Current methods to solve this task fall into two broad categories. Constructive algorithms initially assume a simple network and add nodes and links as warranted (Ash, 1989; Frean, 1990; Hanson, 1990; Fahlman and Lebiere, 1990; Fahlman, 1991; Chen, et al., 1993; Azimi-Sadjadi, Sheedvash, and Trujillo, 1993), while destructive methods start with a large network and prune off superfluous components (Mozer and Smolensky, 1989; Cun, Denker, and Solla, 1990; Hassibi and Stork, 1993; Omlin and Giles, 1993). Though these algorithms address the problem of topology acquisition, they do so in a highly constrained manner. Because they monotonically modify network structure, constructive and destructive methods limit the traversal of the available architectures in that once an architecture has been explored and determined to be insufficient, a new architecture is adopted, and the old becomes topologically unreachable. Also, these methods often use only a single predefined structural modification, such as “add a fully connected hidden unit,” to generate successive topologies. This is a form of structural hill climbing, which is
susceptible to becoming trapped at structural local minima. In addition, constructive and destructive algorithms make simplifying architectural assumptions to facilitate network induction. For example, Ash (1989) allows only feedforward networks; Fahlman (1991) assumes a restricted form of recurrence, and Chen, et al., (1993) explore only fully connected topologies. This creates a situation in which the task is forced into the architecture rather than the architecture being fit to the task.

These deficiencies of constructive and destructive methods stem from inadequate methods for assigning credit to structural components of a network. As a result, the heuristics used are overly-constrained to increase the likelihood of finding any topology to solve the problem. Ideally, the constraints for such a solution should come from the task rather than be implicit in the algorithm.

This chapter presents GNARL, a network induction algorithm that simultaneously acquires both network topology and weight values while making minimal architectural restrictions and avoiding structural hill climbing. The algorithm, described in section 3, is an instance of evolutionary programming (Fogel, Owens, and Walsh, 1966; Fogel, 1992a), a class of evolutionary computation that has been shown to perform well at function optimization. Section 2 argues that this class of evolutionary computation is better suited for evolving neural networks than genetic algorithms (Holland, 1975; Goldberg, 1989a), a more popular class of evolutionary computation. Finally, section 4 demonstrates GNARL's ability to create recurrent networks for a variety of problems of interest.
4.3 The Evolution of Neural Networks

*Evolutionary computation* provides a promising collection of algorithms for agent evolution, in particular, for agents represented as recurrent neural networks (Fogel, 1994). These algorithms are distinguished by their reliance on a *population* of search space positions, rather than a single position, to locate extrema of a function defined over the search space. During one search cycle, or *generation*, the members of the population are ranked according to a *fitness function*, and those with higher fitness are probabilistically selected to become *parents* in the next generation. New population members, called *offspring*, are created using specialized *reproduction heuristics*. Using the population, reproduction heuristics, and fitness function, evolutionary computation implements a nonmonotonic search that performs well in complex multimodal environments. Classes of evolutionary computation can be distinguished by examining the specific reproduction heuristics employed.

*Genetic algorithms* (GAs) (Holland, 1975; Goldberg, 1989a) are a popular form of evolutionary computation that rely chiefly on the reproduction heuristic of *crossover*.2 This operator forms offspring by recombining representational components from two members of the population without regard to content. This purely structural approach to creating novel population members assumes that components of all parent representations may be freely exchanged without inhibiting the search process.

Various combinations of GAs and connectionist networks have been investigated. Much research concentrates on the acquisition of parameters for a fixed network

---

2. Genetic algorithms also employ other operators to manipulate the population, including a form of mutation, but their distinguishing feature is a heavy reliance on crossover.
architecture (e.g., Wieland, 1990; Montana and Davis, 1989; Whitley, Starkweather, and Bogart, 1990; Beer and Gallagher, 1992). Other work allows a variable topology, but disassociates structure acquisition from acquisition of weight values by interweaving a GA search for network topology with a traditional parametric training algorithm (e.g., backpropagation) over weights (e.g., Miller, Todd, and Hegde, 1989; Belew, McInerney, and Schraudolf, 1992). Some studies attempt to coevolve both the topology and weight values within the GA framework, but as in the connectionist systems described above, the network architectures are restricted (e.g., Torreele, 1991; Potter, 1992; Karunanithi, Das, and Whitley, 1992). In spite of this collection of studies, current theory from both genetic algorithms and connectionism suggests that GAs are not well-suited for evolving networks. In the following section, the reasons for this mismatch are explored.

4.3.1 Networks and Genetic Algorithms

Genetic algorithms create new individuals by recombining the representational components of two members of the population. Because of this commitment to structural recombination, GAs typically rely on two distinct representational spaces (Figure 22). Recombination space, usually defined over a set of fixed-length binary strings, is the set of structures to which the genetic operators are applied. It is here that the search actually occurs. Evaluation space, typically involving a problem-dependent representation, is the set of structures whose ability to perform a task is evaluated. In the case of using GAs to evolve networks, evaluation space is comprised of a set of networks. An interpretation function maps between these two representational spaces. Any set of finite-length bit strings cannot represent all possible networks, thus the evaluation space is restricted to a predetermined set of networks. By design, the dual representation scheme allows the GA
to crossover the bit strings without any knowledge of their interpretation as networks. The implicit assumption is that the interpretation function is defined so that the bit strings created by the dynamics of the GA will map to successively better networks.

The dual representation of GAs is an important feature for searching in certain environments. For instance, when it is unclear how to search the evaluation space directly, and when there exists an interpretation function such that searching the space of bit strings by crossover leads to good points in evaluation space, then the dual representation is ideal. It is unclear, however, that there exists an interpretation function that makes dual representation beneficial for evolving neural networks. Clearly, the choice of interpretation function introduces a strong bias into the search, typically by excluding many potentially interesting and useful networks (another example of forcing the task into an architecture).
Moreover, the benefits of having a dual representation hinges on crossover being an appropriate evolutionary operator for the task for some particular interpretation function; otherwise, the need to translate between dual representations is an unnecessary complication.

Characterizing tasks for which crossover is a beneficial operator is an open question. Current theory suggests that crossover will tend to recombine short, connected substrings of the bit string representation that correspond to above-average task solutions when evaluated (Holland, 1975; Goldberg, 1989a). These substrings are called building blocks, making explicit the intuition that larger structures with high fitness are built out of smaller structures with moderate fitness. Crossover tends to be most effective in environments where the fitness of a member of the population is reasonably correlated with the expected ability of its representational components (Goldberg, 1989b). Environments where this is not true are called deceptive (Goldberg, 1989c).

There are three forms of deception when using crossover to evolve connectionist networks. The first involves networks that share both a common topology and common weights. Because the interpretation function may be many-to-one, two such networks need not have the same bit string representation (see Figure 23). Crossover will then tend to create offspring that contain repeated components, and lose the computational ability of some of the parents' hidden units. The resulting networks will tend to perform worse than their parents because they do not possess key computational components for the task. Schaffer, et al. (1992) term this the competing conventions problem, and point out that the number of competing conventions grows exponentially with the number of hidden units.
Figure 23: The competing conventions problem (Schaffer, Whitley, and Eshelman, 1992). Bit strings A and B map to structurally and computationally equivalent networks that assign the hidden units in different orders. Because the bit strings are distinct, crossover is likely to produce an offspring that contains multiple copies of the same hidden node, yielding a network with less computational ability than either parent.

The second form of deception involves two networks with identical topologies but different weights. It is well known that for a given task, a single connectionist topology affords multiple solutions for a task, each implemented by a unique distributed representation spread across the hidden units (Hinton, McClelland, and Rumelhart, 1986; Sejnowski and Rosenberg, 1987). While the removal of a small number of nodes has been shown to effect only minor alterations in the performance of a trained network (Hinton, McClelland, and Rumelhart, 1986; Sejnowski and Rosenberg, 1987), the computational role each node plays in the overall representation of the task solution is determined purely by the presence and strengths of its interconnections. Furthermore, there need be no
correlation between distinct distributed representations over a particular network architecture for a given task. This seriously reduces the chance that an arbitrary crossover operation between distinct distributed representations will construct viable offspring regardless of the interpretation function used.

Finally, deception can occur when the parents differ topologically. The types of distributed representations that can develop in a network vary widely with the number of hidden units and the network's connectivity. Thus, the distributed representations of topologically distinct networks have a greater chance of being incompatible parents. This further reduces the likelihood that crossover will produce good offspring.

In short, for crossover to be a viable operator when evolving networks, the interpretation function must somehow compensate for all the types of deceptiveness described above. This suggests that the complexity of an appropriate interpretation function will more than rival the complexity of the original learning problem. Thus, the prospect of evolving connectionist networks with crossover appears limited in general, and better results should be expected with reproduction heuristics that respect the uniqueness of the distributed representations. This point has been tacitly validated in the genetic algorithm literature by a trend towards a reduced reliance on binary representations when evolving networks (e.g. Koza and Rice, 1991; Collins and Jefferson, 1991). Crossover, however, is still commonplace.

4.3.2 Networks and Evolutionary Programming

Unlike genetic algorithms, evolutionary programming (EP) (Fogel, 1992a; Fogel, 1992b) defines representation-dependent mutation operators that create offspring within a specific
Figure 24: The evolutionary programming approach to modeling evolution. Unlike genetic algorithms, evolutionary programs perform search in the space of networks. Offspring created by mutation remain within a locus of similarity to their parents.

locus of the parent (see Figure 24). EP’s commitment to mutation as the sole reproductive operator for searching over a space is preferable when there is no sufficient calculus to guide recombination by crossover, or when separating the search and evaluation spaces does not afford an advantage.

In each of the above studies, the mutation operator alters the parameters of network \( \eta \) by the function:

\[
    w = w + N(0, \alpha \varepsilon(\eta)) \quad \forall w \in \eta
\]

(Eqn 1)

where \( w \) is a weight, \( \varepsilon(\eta) \) is the error of the network on the task (typically the mean squared error), \( \alpha \) is a user-defined proportionality constant, and \( N(\mu, \sigma^2) \) is a gaussian variable with mean \( \mu \) and variance \( \sigma^2 \). The implementations of structural mutations in these studies differ somewhat. McDonnell and Waagen (1992) randomly select a set of weights and alters their values with a probability based on the variance of the incident nodes' activation over the training set; connections from nodes with a high variance having less of a chance of being altered. The structural mutation used by Fogel (1992a; 1993) adds or deletes a single hidden unit with equal probability.

Evolutionary programming provides distinct advantages over genetic algorithms when evolving networks. First, EP manipulates networks directly, thus obviating the need for a dual representation and the associated interpretation function. Second, by avoiding crossover between networks in creating offspring, the individuality of each network's distributed representation is respected. For these reasons, evolutionary programming provides a more appropriate framework for simultaneous structural and parametric learning in recurrent networks.

### 4.4 The GNARL Algorithm

GNARL (GeNeralized Acquisition of Recurrent Links) is an evolutionary program that non-monotonically constructs recurrent networks to solve a given task. The name GNARL reflects the types of networks that arise from a generalized network induction algorithm...
performing both structural and parametric learning. Instead of having uniform or symmetric topologies, the resulting networks have “gnarled” interconnections of hidden units which more accurately reflect constraints inherent in the task.

The algorithm begins with an initial population of $n$ random individuals; a sample network $N$ is shown in Figure 25. The number of input nodes ($m_{in}$) and number of output nodes ($m_{out}$) are fixed for a given task; the number of hidden nodes as well as the connections among them are free to vary from 0 to a user-supplied maximum $h_{\text{max}}$. Bias is optional; if provided in an experiment, it is implemented as an additional input node with constant value one. All non-input nodes employ the standard sigmoid activation function. Links use real-valued weights, and must obey three restrictions:

$R_1$: There can be no links to an input node.

$R_2$: There can be no links from an output node.

$R_3$: Given two nodes $x$ and $y$, there is at most one link from $x$ to $y$. 

**Figure 25**: Sample initial network. The number of input nodes ($m_{\text{in}}$) and number of output nodes ($m_{\text{out}}$) is fixed for a given task. The presence of a bias node ($b = 0$ or 1) as well as the maximum number of hidden units ($h_{\text{max}}$) is set by the user. The initial connectivity is chosen randomly (see text). The disconnected hidden node does not affect this particular network's computation, but is available as a resource for structural mutations.
Thus GNARL networks may have no connections, sparse connections, or full connectivity. Consequently, GNARL’s search space is:

\[ S = \{ \eta : \eta \text{ is a network with real-valued weights,} \]
\[ \eta \text{ satisfies } R_1-R_3, \]
\[ \eta \text{ has } m_{in} + b \text{ input nodes, where } b=1 \text{ if bias is provided, and } 0 \text{ otherwise,} \]
\[ \eta \text{ has } m_{out} \text{ output nodes,} \]
\[ \eta \text{ has } i \text{ hidden nodes, } 0 \leq i \leq h_{max} \} \]

\( R_1-R_3 \) are strictly implementational constraints. Nothing in the algorithm described below hinges on \( S \) being pruned by these restrictions.

4.4.1 Selection, Reproduction and Mutation of Networks

GNARL initializes the population with randomly generated networks (Figure 25). The number of hidden nodes for each network is chosen from a uniform distribution over a user-supplied range. The number of initial links is chosen similarly from a second user-supplied range. The incident nodes for each link are chosen in accordance with the structural mutations described below. Once a topology has been chosen, all links are assigned random weights, selected uniformly from the range \([-1, 1]\). There is nothing in this initialization procedure that forces a node to have any incident links, let alone for a path to exist between the input and output nodes. In the experiments below, the number of hidden units for a network in the initial population was selected uniformly between one and five and the number of initial links varied uniformly between one and 10.

In each generation of search, the networks are first evaluated by a user-supplied fitness function \( f : S \rightarrow \mathbb{R} \), where \( \mathbb{R} \) represents the reals. Networks scoring in the top 50% are designated as the parents of the next generation; all other networks are discarded. This
selection method is used in many EP algorithms although competitive methods of selection have also been investigated (Fogel, 1992a).

Generating an offspring involves three steps: copying the parent, determining the severity of the mutations to be performed, and finally mutating the copy. Network mutations are separated into two classes, corresponding with the types of learning discussed in (Barto, 1990). Parametric mutations alter the value of parameters (link weights) currently in the network, whereas structural mutations alter the number of hidden nodes and the presence of links in the network, thus altering the space of parameters.

4.4.1.1 Severity of Mutations

The severity of a mutation to a given parent, $\eta$, is dictated by that network's temperature, $T(\eta)$:

$$T(\eta) = 1 - \frac{f(\eta)}{f_{\text{max}}}$$  \hspace{1cm} (Eqn 2)

where $f_{\text{max}}$ is the maximum fitness for a given task. Thus, the temperature of a network is determined by how close the network is to being a solution for the task. This measure of the network's performance is used to anneal the structural and parametric similarity between parent and offspring, so that networks with a high temperature are mutated severely, and those with a low temperature are mutated only slightly. This allows a coarse-grained search initially, and a progressively finer-grained search as a network approaches a solution to the task. $T(\eta)$ is related to the concept of temperature in simulated annealing (Kirkpatrick, Gelatt, and Vecchi, 1993) where a higher temperature indirectly increases the variety of states that can be visited by the system.
4.4.1.2 Parametric Mutation of Networks

Parametric mutations are accomplished by perturbing each weight $w$ of a network $\eta$ with gaussian noise, a method motivated by (Fogel, 1993; Fogel, 1992a). In that body of work, weights are modified as follows:

$$w = w + N(0, \alpha T(\eta)) \quad \forall w \in \eta$$

(Eqn 3)

where $\alpha$ is a user-defined proportionality constant, and $N(\mu, \sigma^2)$ is a gaussian random variable as before. While large parametric mutations are occasionally necessary to avoid parametric local minima during search, it is more likely they will adversely affect the offspring’s ability to perform better than its parent. To compensate, GNARL updates weights using a variant of Equation 3. First, the instantaneous temperature $\hat{T}$ of the network is computed:

$$\hat{T}(\eta) = U(0, 1) T(\eta)$$

(Eqn 4)

where $U(0, 1)$ is a uniform random variable over the interval [0, 1]. This new temperature, varying from 0 to $T(\eta)$, is then substituted into Equation 3:

$$w = w + N(0, \alpha \hat{T}(\eta)) \quad \forall w \in \eta$$

(Eqn 5)

In essence, this modification lessens the frequency of large parametric mutations without disallowing them completely. In the experiments described below, $\alpha$ is one.

4.4.1.3 Structural Mutation of Networks

The structural mutations used by GNARL alter the number of hidden nodes and the connectivity between all nodes, subject to restrictions $R_1-R_3$ discussed earlier. To avoid
radical jumps in fitness from parent to offspring, structural mutations attempt to preserve the behavior of a network. For instance, new links are initialized with zero weight, leaving the behavior of the modified network unchanged. Similarly, hidden units are added to the network without any incident connections. Links must be added by future structural mutations to determine how to incorporate the new computational unit. Unfortunately, achieving this behavioral continuity between parent and child is not so simple when removing a hidden node or link. Consequently, the deletion of a node involves the complete removal of the node and all incident links with no further modification to compensate for the behavioral change. Similarly, deleting a link removes that parameter from the network.

The selection of which node to remove is uniform over the collection of hidden nodes. Addition or deletion of a link is slightly more complicated in that a parameter identifies the likelihood that the link will originate from an input node or terminate at an output node. Once the class of incident node is determined, an actual node is chosen uniformly from the class. Biasing the link selection process in this way is necessary when there is a large differential between the number of hidden nodes and the number of input or output nodes. This parameter was set to 0.2 in the experiments described in the next section.

Research in (Fogel, 1992a) and (Fogel, 1993) uses the heuristic of adding or deleting at most a single fully connected node per structural mutation. Therefore, it is possible for this method to become trapped at a structural local minima, although this is less probable than in nonevolutionary algorithms given that several topologies may be present in the population. In order to more effectively search the range of network architectures, GNARL uses a severity of mutation for each separate structural mutation. A unique user-defined interval specifying a range of modification is associated with each of
the four structural mutations. Given an interval of \([\Delta_{\text{min}}, \Delta_{\text{max}}]\) for a particular structural mutation, the number of modifications of this type made to an offspring is given by:

\[
\Delta_{\text{min}} + \left[ U [0, 1] \hat{f}(\eta) (\Delta_{\text{max}} - \Delta_{\text{min}}) \right]
\]  

(Eqn 6)

Thus the number of modifications varies uniformly over a shrinking interval based on the parent network's fitness. In the experiments below, the maximum number of nodes added or deleted was three while the maximum number of links added or deleted was five. The minimum number for each interval was always one.

4.4.2 Fitness of a Network

In evolving networks to perform a task, GNARL does not require an explicit target vector — all that is needed is the feedback given by the fitness function \(f\). But if such a vector is present, as in supervised learning, there are many ways of transforming it into a measure of fitness. For example, given a training set \(\{(x_1, y_1), (x_2, y_2), \ldots\}\), three possible measures of fitness for a network \(\eta\) are sum of square errors (Equation 7), sum of absolute errors (Equation 8), and sum of exponential absolute errors (Equation 9):

\[
\sum_i (y_i - \text{Out}(\eta, x_i))^2
\]  

(Eqn 7)

\[
\sum_i |y_i - \text{Out}(\eta, x_i)|
\]  

(Eqn 8)

\[
\sum_i e^{y_i - \text{Out}(\eta, x_i)}
\]  

(Eqn 9)

Furthermore, because GNARL explores the space of networks by mutation and selection, the choice of fitness function does not alter the mechanics of the algorithm. To show
Figure 26: An FSA that defines the enable-trigger task (Williams, 1990). The system is given a data stream of bit pairs \{ (a_1, b_1), (a_2, b_2), ... \}, and produces an output of 0's and 1's. To capture this system's input/output behavior, a connectionist network must learn to store state indefinitely.

GNARL's flexibility, each of these fitness functions will be demonstrated in the experiments below.

4.5 Results

In this section, GNARL is applied to several problems of interest. The goal in this section is to demonstrate the abilities of the algorithm on two very different problems.\(^3\) The various parameter values for the program are set as described above unless otherwise noted.

4.5.1 The Enable-Trigger Task

As an initial test, GNARL induced a solution for the enable-trigger task proposed by Williams (1990). Consider the finite state generator shown in Figure 26. At each time step the system receives two input bits, \((a, b)\), representing “enable” and “trigger” signals.

\(^3\) GNARL has also been applied to other problems, including language induction (Angeline, Saunders, and Pollack, 1994).
respectively. This system begins in state $S_1$, and switches to state $S_2$ only when enabled by $a=1$. The system remains in $S_2$ until it is triggered by $b=1$, at which point it outputs 1 and resets the state to $S_1$. So, for instance, on an input stream $\{(0, 0), (0, 1), (1, 1), (0, 1)\}$, the system will output $\{0, 0, 0, 1\}$ and end in $S_1$. This simple problem allows an indefinite amount of time to pass between the enable and the trigger inputs; thus no finite length sample of the output stream will indicate the current state of the system. This forces GNARL to develop networks that can preserve state information indefinitely.

The fitness function used in this experiment was the sum of exponential absolute errors (equation 9). Population size was 50 networks with the maximum number of hidden units restricted to six. A bias node was provided in each network in this initial experiment, ensuring that an activation value of 1 was always available. Note that this does not imply that each node had a nonzero bias; links to the bias node had to be acquired by structural mutation.

Training began with all two input strings of length two, shown in Table 1. After 118 generations (3000 network evaluations\(^4\)), GNARL evolved a network which solved this task for the strings in Table 1 within tolerance of 0.3 on the output units. The training set was then increased to include all 64 input strings of length three and evolution of the networks was allowed to continue. After an additional 422 generations, GNARL once again found a suitable network. At this point, the difficulty of the task was increased a final time by training on all 256 strings of length four. After another 225 generations (~20000 network evaluations total) GNARL once again found a network to solve this task.

---

\(^4\) Number of networks evaluated = $|\text{population}| + \text{generations} \times |\text{population}| \times 50\%$ of the population removed each generation, giving $50 + 118 \times 50 \times 0.5 = 3000$ network evaluations for this trial.
**Table 2: Initial training data for enable-trigger task of Wilson (1990).**

<table>
<thead>
<tr>
<th>Input</th>
<th>Target output</th>
</tr>
</thead>
<tbody>
<tr>
<td>{(0, 0), (0, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 0), (0, 1)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 0), (1, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 0), (1, 1)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 1), (0, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 1), (0, 1)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 1), (1, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(0, 1), (1, 1)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(1, 0), (0, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(1, 0), (0, 1)}</td>
<td>{0, 1}</td>
</tr>
<tr>
<td>{(1, 0), (1, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(1, 0), (1, 1)}</td>
<td>{0, 1}</td>
</tr>
<tr>
<td>{(1, 1), (0, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(1, 1), (0, 1)}</td>
<td>{0, 1}</td>
</tr>
<tr>
<td>{(1, 1), (1, 0)}</td>
<td>{0, 0}</td>
</tr>
<tr>
<td>{(1, 1), (1, 1)}</td>
<td>{0, 1}</td>
</tr>
</tbody>
</table>
Figure 27: Connectivity of two recurrent networks found in the enable-trigger experiment. (a) The best network of generation one. (b) The best network of generation 765. This network solves the task for all strings of length eight.

in Figure 27b. Note that there are two completely isolated nodes. Given the fitness function used in this experiment, the two isolated nodes do not effect the network's viability. To investigate the generalization of this network, it was tested over all 4096 unique strings of length six. The outputs were rounded off to the nearest integer, testing only the network's separation of the strings. The network performed correctly on 99.5% of this novel set, generating incorrect responses for only 20 strings.

Figure 28 shows the connectivity of the population member with the best fitness for each generation over the course of the run. Initially, the best network is sparsely-connected and remains sparsely-connected throughout most of the run. At about generation 400, the size and connectivity increases dramatically only to be overtaken by the relatively sparse architecture shown in Figure 27b on the final generation. Apparently, this more sparsely connected network evolved more quickly than the more full architectures that were best in earlier generations. The oscillations between different network architectures throughout the run reflects the development of such competing architectures in the population.
Figure 28: Different network topologies explored by GNARL during the first 540 generations on the enable-trigger problem. The presence of a link between node i and j at generation g is indicated by a dot at position (g, 10 * i + j) in the graph. Note that because node 3 is the output node, there are no connections from it throughout the run. The arrow designates the point of transition between the first two training sets.
4.5.2 The Tracker Task

GNARL was tested on a complex search and collection task – the Tracker task described in (Jefferson, et al., 1992), and further investigated in (Koza, 1992). In this problem, a simulated ant is placed on a two-dimensional toroidal grid that contains a trail of food. The ant traverses the grid, collecting any food it contacts along the way. The goal of the task is to discover an ant which collects the maximum number of pieces of food in a given time period. (Figure 29).

Following (Jefferson, et al., 1992), each ant is controlled by a network with two input nodes and four output nodes (Figure 30). The first input node denotes the presence of food in the square directly in front of the ant; the second denotes the absence of food in this same square, restricting the possible legal inputs to the network to (1, 0) or (0, 1). Each of the four output units corresponds to a unique action: move forward one step, turn left 90°, turn right 90°, or no-op. At each step, the action whose corresponding output node has maximum activation is performed. As in the original study (Jefferson, et al., 1992), no-op allows the ant to remain at a fixed position while activation flows along recurrent connections. Fitness is defined as the number of grid positions cleared within 200 time steps. The task is difficult because simple networks can perform surprisingly well; the network shown in Figure 30 collects 42 pieces of food before spinning endlessly at position A (in Figure 29), illustrating a very high local maximum in the search space.

The experiment used a population of 100 networks, each limited to at most nine hidden units, and did not provide a bias node. In the first run (2090 generations), GNARL found a network (Figure 31b) that clears 81 grid positions within the 200 time steps. When this ant is run for an additional 119 time steps, it successfully clears the entire trail. To
**Figure 29:** The ant problem. The trail is connected initially, but becomes progressively more difficult to follow. The underlying 2-d grid is toroidal, so that position "A" is the first break in the trail—it is simple to reach this point. Positions "B" and "C" indicate the only two positions along the trail where the ant discovered in run 1 behaves differently from the 5-state FSA of Jefferson, et al., (1992) (see Figure 32).
Figure 30: The semantics of the I/O units for the ant network. The first input node denotes the presence of food in the square directly in front of the ant; the second denotes the absence of food in this same square. This particular network finds 42 pieces of food before spinning endlessly in place at position P, illustrating a very deep local minimum in the search space.

understand how the network traverses the path of food, consider the simple FSA shown in Figure 32, hand-crafted by Jefferson, et al., (1992) as an approximate solution to the problem. This simple machine receives a score of 81 in the allotted 200 time steps, and clears the entire trail only five time steps faster than the network in Figure 31b. A step by step comparison indicates there is only a slight difference between the two. GNARL’s evolved network follows the general strategy embodied by this FSA at all but two places, marked as positions B and C in Figure 29. Here the evolved network makes a few additional moves, accounting for the slightly longer completion time.

Figure 33 illustrates the strategy the network uses to implement the FSA by showing the state of the output units of the network over three different sets. Each point is a triple of the form (move, right, left). Figure 33a shows the result of supplying to the network 200 “food” inputs – a fixed point that executes “Move.” Figure 33b shows the

5. Note however that the network’s behavior is not precisely captured by the FSA. Kolen (1994b, 1994c) shows that, in general, FSAs approximate networks only poorly. The next network I will describe makes this same point empirically.

6. No-op is not shown because it was never used in the final network.
Figure 31: The Tracker Task, first run. (a) The best network in the initial population. Nodes 0 & 1 are input, nodes 5-8 are output, and nodes 2-4 are hidden nodes. (b) Network induced by GNARL after 2090 generations. Forward links are dashed; bidirectional links & loops are solid. The light gray connection between nodes 8 and 13 is the sole backlink. This network clears the trail in 319 epochs. (c) Jefferson et al.'s fixed network structure for the Tracker task.
Figure 32: FSA hand-crafted for the Tracker task in Jefferson, et al., (1992). The large arrow indicates the initial state. This simple system implements the strategy "move forward if there is food in front of you, otherwise turn right four times, looking for food. If food is found while turning, pursue it, otherwise, move forward one step and repeat." This FSA traverses the entire trail in 314 steps, and gets a score of 81 in the allotted 200 time steps.

sequence of states reached when 200 "no food" signals are supplied to the network - a collection of points describing a limit cycle of length five that repeatedly executes the sequence "Right, Right, Right, Right, Move." These two attractors determine the response of the network to the task (Figure 33c, d); the additional points in Figure 33c are transients encountered as the network alternates between these attractors. The differences in the number of steps required to clear the trail between the FSA of Figure 32 and GNARL's network arise due to the state of the hidden units when transferring from the "food" attractor to the "no food" attractor.

However, not all evolved network behaviors are so simple as to approximate an FSA (Pollack, 1991). In a second run (1595 generations) GNARL induced a network that cleared 82 grid points within the 200 time steps. Figure 34 demonstrates the behavior of this network. Once again, the "food" attractor, shown in Figure 34a, is a single point in the
Figure 33: Limit behavior of the network that clears the trail in 319 steps. Graphs show the state of the output units Move, Right, Left. (a) Fixed point attractor that results for sequence of 500 "food" signals; (b) Limit cycle attractor that results when a sequence of 500 "no food" signals is given to network; (c) All states visited while traversing the trail; (d) The path of the ant on an empty grid. The Z axis represents time. Note that x is fixed, and y increases monotonically at a fixed rate. The large jumps in y position are artifacts of the toroidal grid.
Figure 34: Limit behavior of the network of the second run. Graphs show the state of the output units Move, Right, Left. (a) Fixed point attractor that results for sequence of 3500 “food” signals; (b) Limit cycle attractor that results when a sequence of 3500 “no food” signals is given to network; (c) All states visited while traversing the trail; (d) The path of the ant on an empty grid. The z axis represents time. The ant’s path is comprised of a set of “railroad tracks.” Along each track, tick marks represent back and forth movement. At the junctures between tracks, a more complicated movement occurs. There are no artifacts of the toroidal grid in this plot, all are actual movements (cf. Figure 33d).
space that always executes "Move." The "no food" behavior, however, is not an FSA; instead, it is a quasiperiodic trajectory of points shaped like a "D" in output space (Figure 34b). The placement of the "D" is in the "Move / Right" corner of the space and encodes a complex alternation between these two operations (see Figure 34d).

In contrast, research by Jefferson, et al., (1992) uses a genetic algorithm on a population of 65,536 bit strings with a direct encoding to evolve only the weights of a neural network with five hidden units to solve this task. The particular network architecture by Jefferson, et al., (1992) uses Boolean threshold logic for the hidden units and an identity activation function for the output units. The first GNARL network was discovered after evaluating a total of 104,600 networks while the second was found after evaluating 79,850. The experiment reported by Jefferson, et al., (1992) discovered a comparable network after about 17 generations. Given Jefferson, et al. (1992) used a population size of 65,536 and replaced 95% of the population each generation, the total number of network evaluations to acquire the equivalent network was 1,123,942. This is 10.74 and 14.07 times the number of networks evaluated by GNARL in the two runs. In spite of the differences between the two studies, this significant reduction in the number of evaluations provides empirical evidence that crossover may not be best suited to the evolution of networks.

4.6 Remarks

Given the above results, I now return to the network induction problem, reviewing it in retrospect.
4.6.1 The Network Induction, in Retrospect

Allowing the task to specify an appropriate architecture for its solution should, in principle, be the defining aspect of the complete network induction problem. By restricting the space of networks explored, constructive, destructive, and genetic algorithms only partially address the problem of topology acquisition. GNARL's architectural constraints $R_1-R_3$ similarly reduce the search space, but to a far less degree. Furthermore, none of these constraints is necessary, and their removal would affect only ease of implementation. In fact, no assumed features of GNARL's networks are essential for the algorithm's operation. GNARL could even use nondifferentiable activation functions, a constraint for backpropagation.

GNARL's minimal representational constraints would be meaningless if not complemented by appropriate search dynamics to traverse the space of networks. First, unlike constructive and destructive algorithms, GNARL permits a nonmonotonic search over the space of network topologies. Consider that in monotonic search algorithms, the questions of when and how to modify structure take on great significance because a premature topological change cannot be undone. In contrast, GNARL can revisit a particular architecture at any point, but for the architecture to be propagated it must confer an advantage over other competing topologies. Such a non-linear traversal of the space is imperative for acquiring appropriate solutions because the efficacy of the various architectures changes as the parametric values are modified.

GNARL allows multiple structural manipulations to a network within a single mutation. As discussed earlier, constructive and destructive algorithms define a unit of modification, e.g., "add a fully connected hidden node." Because such singular structural
modifications create a "one-unit structural horizon" beyond which no information is available, such algorithms may easily fixate on an architecture that is better than networks one modification step away, but worse than those two or more steps distant. In GNARL, several nodes and links can be added or deleted with each mutation, the range being determined by user-specified limits and the current ability of the network. This simultaneous modification of the structural and parametric modifications based on fitness allows the algorithm to discover appropriate networks quickly especially in comparison to evolutionary techniques that do not respect the uniqueness of distributed representations.

Finally, as in all evolutionary computation, GNARL maintains a population of structures during the search. This allows the algorithm to investigate several differing architectures in parallel while avoiding over-commitment to a particular network topology.

These search dynamics, combined with GNARL's minimal representational constraints make the algorithm extremely versatile. Of course, if topological constraints are known a priori, they should be incorporated into the search. *But these should be introduced as part of the task specification rather than being built into the search algorithm.* Because the only requirement on a fitness function $f$ is that $f: S \rightarrow \mathbb{R}$, diverse criteria can be used to rate a network's performance. For instance, the first two experiments described above evaluated networks based on a desired input/output mapping; the Tracker task experiment, however, considered overall network performance, not specific mappings. Other criteria could also be introduced, including specific structural constraints (e.g., minimal number of hidden units or links) as well as constraints on generalization. In some cases, strong task restrictions can even be implicit in simple fitness functions (Angeline and Pollack, 1993).
The dynamics of the algorithms guided by the task constraints represented in the fitness function allow GNARL to empirically determine an appropriate architecture. Over time, the continual cycle of test-prune-reproduce will constrain the population to only those architectures that have acquired the task most rapidly. Inappropriate networks will not be indefinitely competitive and will be removed from the population eventually.

Complete network induction must be approached with respect to the complex interaction between network topology, parametric values, and task performance. By fixing topology, gradient descent methods can be used to discover appropriate solutions. But the relationship between network structure and task performance is not well understood, and there is no "backpropagation" through the space of network architectures. Instead, the network induction problem is approached with heuristics that, as described above, often restrict the available architectures, the dynamics of the search mechanism, or both. Artificial architectural constraints (such as "feedforwardness") or overly constrained search mechanisms can impede the induction of entire classes of behaviors, while forced structural liberties (such as assumed full recurrence) may unnecessarily increase structural complexity or learning time. By relying on a simple stochastic process, GNARL strikes a middle ground between these two extremes, allowing the network's complexity and behavior to emerge in response to the demands of the task.

4.6.2 The Induction of Modules

In Chapter III, I mentioned an assumption common to subsumption and preemption; namely, the assumption of a particular behavioral decomposition. GNARL shows how to achieve a particular task without assuming such a decomposition, but the resulting networks (Figure 31b, for example), can hardly be called modular.
In Section 3.5.2, however, I pointed out that Angeline (Angeline, 1994; Angeline and Pollack, 1992; Angeline and Pollack, 1993) has done extensive work on the automatic induction of modules. His approach is similar to GNARL in that he relies on an evolutionary program to sample the space of structures. In contrast to GNARL (and other evolutionary programs, for that matter), Angeline simultaneously uses evolutionary search to explore various modularizations of structure. He has shown that this method allows modules to emerge from the task in a natural way.

4.6.3 GNARL and Communication

The results from the Tracker task (Section 4.5.2) show the range of agents which GNARL can induce. But the environment of this task is simple in two respects: it is completely deterministic, and it involves only a single agent. At this point, the ability of GNARL to find solutions for nondeterministic multi-agent tasks remains an open question. Evolving communication is just such a task, to which I turn in the next chapter.
CHAPTER V

THE EVOLUTION OF COMMUNICATION

[Problems in natural language understanding] stem from a picture of a program constructed of cooperating modules that "talk" to each other. While this may be a reasonable metaphor in some ways, anyone who has actually written such a program knows that "talking" is a very poor model of the communication. Yet many researchers...find English to be the ideal notation in which to encode messages. They are aware that message-passing channels are the most frustrating bottleneck through which intelligence must pass, so they wish their way into the solution: let the modules speak in human tongues! Let them use metaphor, allusion, hints, polite requests, pleading, flattery, bribes, and patriotic exhortations to their fellow modules!

- Drew McDermott¹

5.1 Introduction

McDermott's humorous warning, originally published in 1976, remains highly relevant today, especially to researchers in the study of adaptive behavior. While the work in this chapter does not approach metaphor, allusion, hints, etc., it does offer an approach towards communication which differs significantly from the state of affairs described by McDermott, where programs communicated by passing pointers and tables to each other. Instead, my agents evolve a scheme whereby one's squawking becomes meaningful to the others, as will be described below.

5.1.1 A Shift in Environments

Chapter IV showed how evolutionary search could allow a single agent to adapt to the environment of the Tracker task. This case is simple in two respects. First, the environment is deterministic; second, all changes to the environment are caused by the agent.

When multiple agents exist, however, the situation becomes more difficult. From the perspective of any one of the agents, the environment may change independently of that agent. Moreover, unlike the fixed agents in Pengi (Agre and Chapman, 1986; Section 2.3), all my agents evolve over time. Finally, I do not always assume perfect channels; noise within channels creates nondeterminism, raising the complexity of the environment even further.

5.1.2 Review of Related Work

As I stated in earlier, the role of communication in multi-agent systems remains one of the most important open issues in multi-agent system design (Brooks, 1991a; Arkin and Hobbs, 1993), and consequently several other researchers have explored this issue (Yanco and Stein, 1993; Werner and Dyer, 1992; MacLennan, 1992; Collins and Jefferson, 1991, 1992). But as I described in Chapter II, all this work shares an emphasis on discrete communication, with the assumption that the size of the language necessary to solve a task is known in advance. Furthermore, because each symbol in a discrete system is isolated, interpolation between messages is inhibited, making it more difficult to deal with noise. Finally, some studies make an architectural distinction between the agent sending the message and the recipient (Yanco and Stein, 1993; Werner and Dyer, 1992; and to some extend MacLennan, 1992, in the sense that at any given time, there is a privileged agent attempting to convey its local information to the others).
5.1.3 An Alternate Approach

My approach to understanding multi-agent communication differs significantly from all the work described above. Rather than assume the transmission of discrete signals between agents, I provide my agents with continuous channels capable of supporting a wide variety of communication schemes. Furthermore, I make no architectural distinctions between transmitter and receiver.

As will be described below, I model agents as connectionist networks. I supply each agent with a number of communications channels implemented by the addition of both input and output units for each channel. The output units initiate environmental signals which are perturbed by environmental noise and whose amplitude decays over distance. An agent does not receive input from other individuals, rather the agent’s input reflects the summation of all other agents’ output signals along that channel. Because I use real-valued activations, the agents communicate using real-valued vectors. Under GNARL, the agents coevolve a communication scheme over continuous channels which in order to be successful conveys task-specific information.

5.2 Communication with Continuous Symbols

This section describes my main experiments. First I introduce an extension of the Tracker task (Jefferson et al., 1992), which will serve as a substrate for my experiments. Next, I describe the method of communication my agents employ. Finally, I describe my experimental results.
5.2.1 The Tracker Task, Revisited

To study the evolution of communication in groups of agents, I extend the Tracker task in three ways (Figure 35):

- increasing the number of agents
- increasing the size of the grid to accommodate these agents
- moving all the food to a small area in the center of the environment

I assume that these modifications will shift the emphasis of the task from evolution of local internal state to evolution of distributed external state, i.e., communication. I concentrate the food within one area so that when an agent finds it and communicates, some food remains by the time other agents arrive. The size of the environment and the amount
of food it contains far exceed the capabilities of a single ant: in the limited time available an ant can neither search the entire space nor consume all the food therein. Thus (I assume) the task design ensures that the only method of complete success necessarily involves communication among the agents.

5.2.2 An Architecture for Communication

When faced with a task requiring communication, the architecture of Jefferson, et al., (Figure 30) will certainly fail; namely, because it in no way supports communication. To remedy this shortcoming, I add $n$ additional input and output units to the network of Figure 30, representing $n$ channels of communication. (Figure 36.) I maintain, from the original study, an implicit winner-take-all network on the (non-signal) outputs.

Output signals propagate throughout the environment, decaying in inverse proportion to squared distance.$^2$ Perception of these signals is governed by Equation 10.
The input signal to agent $a$ along the $i^{th}$ channel, $s_{IN}(a, i)$, is a summation of the signals of all other agents along this channel. $A$ is the set of agents, $s_{out}(b, i)$ is the $i^{th}$ output signal of agent $b$. The noise in the channel, $U[-u_i, u_i]$ is a uniform random number with range specific to the channel, and $\sigma$ is a linear threshold function, which bounds the signals in all channels to a user-specified range $[s_{min}, s_{max}]$. In the experiments below, $s_{min} = 0$ and $s_{max} = 40$.

$$s_{IN}(a, i) = \sum_{b \in A, b \neq a} \frac{\sigma(s_{OUT}(b, i) + U[-u_i, u_i])}{\text{distance}^2(a, b) \sigma(s_{OUT}(b, i) + U[-u_i, u_i])} \tag{Eqn 10}$$

Effectively, this equation creates a "sound-like" model of signal propagation, an example of which is shown in Figure 37.

For the studies reported in this chapter, all activations are continuous; only the hidden activation is squashed (with the standard sigmoid function). Fitness is measured by simply observing the total amount of food eaten by the group. Finally, all agents in an environment are homogeneous in that they share not only the architecture of Figure 36, but also common weights. As show below, however, their behaviors will be quite different depending upon each agent's perspective of its world, creating a heterogenous group at the behavioral level.

---

2. I assume that the signals propagate much faster than the agents react (as would a sound wave), so that effectively, at each discrete time step, an agent's output signals establish a wave front whose strength decays over distance.
Figure 37: The “sound-like” model of signal propagation created by Equation 10. Agent 1 outputs a relative signal strength of 1; agent 2 outputs a relative signal strength of 2; and agent 3 outputs a relative signal strength of 3. The topological map indicates how these signals interact, with the highest elevations corresponding to the strongest signals.

5.3 Results, Part I

I stated my thesis in Chapter I: task-specific communication schemes can indeed be evolved for cooperative multi-agent systems. Unfortunately, my first set of results does not support this claim, for the architecture of Figure 36 offers an easier solution for the modified Tracker task than communication, as we will see below. I present these results in detail for three reasons: first, these negative results reflect my experimental history in exploring the evolution of communication; second, they will motivate the architecture which does support my claim (Section 5.5); and finally, they illustrate the opportunistic nature of adaptive agent systems.
5.3.1 Experiment 1: The Curmudgeon Strategy

I begin with a very simple case: 2 agents, each with one hidden unit, capable of passing one real number between each other, with no noise ($u_0 = 0$, see Equation 10). Figure 38 shows a series of snapshots as the agents collect food. The radii of the circles correspond to the strength of communication. These agents do fairly well, scoring 196 on the task.

To investigate whether these agents had learned to communicate the presence of food, I plotted the status of each agent over the course of the run (Figures 39 and 40). These and similar figures describe all there is to know about a particular agent. The lowest graph represents the presence of food: when food is detected, the value spikes to one; otherwise it is zero. The next graph(s) represent the input signal(s) from the other agent(s), one graph per communication channel. The “hidden unit” graph(s) indicates the agent’s internal state. The “behavior” graph shows the agent’s behavior in terms of the output units of the architecture of Figure 36: move is 0; left is 1; right is 2; and noop is 3. Finally, the top graph(s) represent the output signal(s) of the agent, one graph for each communication channel.

At first glance, it appears that the agents have indeed learned to communicate the presence of food. The strength of an agent’s output signal is negatively correlated with the presence of food: when the agent sees food, its output signal is low; otherwise, it is high. Behaviorally, Figure 38b appears to show recruitment, i.e., the black agent attracting the white agent to the food.

---

3. The circles denote not signal range, but the radius at which signal strength (i.e., the summand in Equation 10) is one.
Figure 38: The "Curmudgeon" strategy (2 agents, 1 communication channel, no noise). Primitives are "Move," "Left," "Right," and "Noop." (a) t=1; (b) t=25; (c) t=50; (d) t=75; (e) t=100; (f) t=125; (g) t=150; (h) t=175; (i) t=200. See text for details.
Figure 39: Profile of curmudgeon 0 (200 time steps). The lowest graph is the food input: when food is detected, the value spikes to one; otherwise it is zero. The input signal reflects communication from the other agent. The value of the hidden unit reflects internal state. Behaviors are "Move," "Turn left," "Turn right," and "Noop," with values 0-3, respectively.
Figure 40: Profile of curmudgeon 1 (200 time steps). This agent's behavior is sensitive to the input signal from curmudgeon 0, but not in a way that qualifies as communication. See text for details.
In reality, however, these agents are *not* communicating the presence of food. The difference between Figures 38b and c shows that recruitment is not really occurring. In the former, the white agent appears to be moving towards the black agent, but in the latter, the white agent has made no progress towards the food. Instead, it has moved eastward in a path parallel to that of the black agent. Figure 38d is similarly surprising: here, the black agent has moved away from the food patch, and is traveling westward in a path parallel to that of the white agent.

In short, these agents have adopted what might be called “the curmudgeon strategy.” Each agent prefers to be alone – rather than attraction, what we see here is repulsion! When run on an empty grid, the agents move about avoiding each other. Food coerces agents to move closer together, but they still often maintain a separation, as shown above. In retrospect, one can see why this strategy was selected. By pushing each other away, the agents are able to explore a large area of their environment.

5.3.2 Experiment 2: The Diagonal Strategy

After several similar runs, I developed a hypothesis about why the agents refused to communicate the presence of food: it is far easier to use signals to implement a pseudo-random search strategy than for each agent to develop a good search strategy alone and communicate the results.

To test this hypothesis, I decided to endow each agent with a good initial search strategy, and let them evolve communication from there. I chose as my starting point a strategy discovered experimentally by GNARL for the normal Tracker task: the five-state
FSA of Figure 32. I built this FSA into each initial network, and then evolved solutions from there.

To clarify just how much this initial starting point helps, I first present the results of running just the FSA-network on the modified Tracker task (i.e., with no evolution). Figure 41 shows the result for the case of 3 agents. Because the black agent does not communicate the presence of food to the others, the white agent moves continually eastward, following the "search in a circle then move forward" strategy of the FSA. Similarly, the gray agent moves continually southward. These agents receive a score of 128.

Figure 42 shows the performance after evolutionary search for the three agent case. Fitness is now 195. Once again, to investigate whether these agents had learned to communicate the presence of food, I plotted the status of each agent over the course of the run (Figures 43-45); once again I learned the agents are not communicating. In the presence of food, each agent executes the "Move forward" strategy of the FSA. But in the absence of food, each agent ignores its input signal, and executes the strategy "Left, left, left, move, left, move," creating a diagonal walk through the search space. This is evident in the behavior profiles of agents 1 and 2, and even in the snapshots of the three agents. Note in Figure 42b that the black agent enters the food patch north of its initial position; it is moving northwest. Comparing Figures 42a-e shows that the white agent is moving southeast. Finally, the position of the gray agent throughout all the snapshots shows that it is simply moving southwest.

I repeated this experiment several times varying the number of agents (2, 3, and 5), and always obtained similar results. The "diagonal strategy" was the most pervasive, though a few other perturbations of the FSA's behavior were found.
Figure 41: Three FSA-initialized agents, before any evolution (one communication channel, no noise). Primitives are "Move," "Left," "Right," and "Noop." (a) $t=1$; (b) $t=25$; (c) $t=50$; (d) $t=75$; (e) $t=100$; (f) $t=125$; (g) $t=150$; (h) $t=175$; (i) $t=200$. 
Figure 42: Three FSA-initialized agents, after evolution (one communication channel, no noise). Primitives are "Move," "Left," "Right," and "Noop." (a) t=1; (b) t=25; (c) t=50; (d) t=75; (e) t=100; (f) t=125; (g) t=150; (h) t=175; (i) t=200.
Figure 43: Profile of the black agent (FSA-initialized, 200 time steps). Behaviors are "Move," "Turn left," "Turn right," and "Noop," with values 0-3, respectively. This agent has learned to correlate oscillation of its output signal with the absence of food.
Figure 44: Profile of the white agent (FSA-initialized, 200 time steps). The behavior of this agent depends only upon the presence of food, and not upon the varying input signal.
Figure 45: Profile of the gray agent (FSA-initialized, 200 time steps). Despite the large variation in its input signal, this agent has constant behavior. This is due to the fact that it never encounters food.
5.4 The Tracker Task, Rethought

This section examines why communication was not evolved in the experiments above, and proposes a slightly different direction of study.

5.4.1 Analysis of First Results

Why should the evolutionary algorithm favor strategies like “curmudgeon” and “diagonal” as opposed to solutions which involve the communication of food? The problem is twofold: the chosen primitives do not easily support communication, but they do easily support pseudo-random search.

First, following a communication signal is extremely difficult with the primitives “Move, Left, Right, Noop.” Because the agent only has access to the value of the signal at one particular grid position, to follow a gradient it would have to first turn four times, sampling and storing the signal value in four positions, then compare the stored results, turn to the appropriate direction, and finally move.

Second, the same “Move, Left, Right, Noop” primitives do easily support pseudo-random search. We have already seen that for a single agent with internal state can evolve a network which supports complex exploration of the search space (Figure 34, p. 97). With multiple interacting agents, such solutions are even easier to evolve.

In short, the reason for the lack of communication is that the presence of a simpler solution with the “Move, Left, Right, Noop” primitives: wander around the search space in a pseudo-random manner, then eat food when it is discovered. This type of solution was commonplace starting from either random networks (“curmudgeon” strategy, Figure 38) or from FSA-initialized networks (“diagonal” strategy, Figure 42).
5.4.2 A Shift in Architectures

The results above were more surprising than disappointing. As stated earlier, adaptive agent algorithms are inherently opportunistic, and they will always find the simplest solution to a task. My original claim was that task-specific communication schemes can indeed be evolved for cooperative multi-agent systems. The real problem is not that this claim is false; rather, the problem is that the "Move, Left, Right, Noop" primitives make the modified Tracker task too simple: solutions with pseudo-random walks of non-communicating agents abound.

But because the real concern is communication, pseudo-random solutions can be inhibited by simply shifting the task slightly; specifically, by changing the base architecture. To enable agents to exploit each other's signals, I give them the ability to follow communication gradients. (This capability is clearly within the realm of network-based agents, as shown by Addam in Chapter II.) To prevent the agents from alternating between Move, Left, and Right to effect a pseudo-random search, I force them to follow a particular search strategy; namely, that of the network of GNARL's first run (Figure 31, p. 94).

The new agent architecture, shown in Figure 36, allows a clear separation between complexity arising from communication, and complexity arising from clever activation of the output nodes. The \( n \) additional output units represent an agent's actions relative to the \( n \) communication channels. When the \( i^{th} \) "Follow gradient" node receives highest activation, the agent follows the gradient of communication channel \( i \).

These modifications, though not essential to the results, greatly facilitate their analysis. The food collection strategy of the FSA is indeed quite simple; if activated
Figure 46: An agent architecture which prevents pseudo-random solutions of Section 5.3. The "Move, Left, Right, and Noop" actions are condensed into "Follow FSA," one particular strategy found by GNARL. The "Follow gradient" nodes give the agent an Addam-like ability to respond to environmental signals.

repeatedly on a grid containing no food, the agent traverses its environment, turning in circles, but never veering from a straight line (cf. Figure 33d). Thus any agents moving non-linearly in the absence of food must be following a communication signal. Furthermore, because of the implicit winner-take-all network, it is easy to observe which communication signal the agent is pursuing by simply comparing activations across the output nodes.

5.5 Results, Part II

This section describes the results with the new agent architecture. In spite of the fact that these agents possess just two types of possible actions ("follow FSA" or "follow gradient"), the task is more difficult than with the "Move, Left, Right, Noop" primitives. The reason is that pseudo-random solutions are now disallowed, and that to search for food effectively.
the agents must learn to communicate. The difficulty here is in finding a way to do this over continuous channels.

5.5.1 Experiment 3: Constant Communication Schemes

I once again begin with a very simple case: 2 agents, each with one hidden unit, capable of passing one real number between each other, with no noise ($u_0 = 0$, see Equation 10). Figure 47a shows the initial environment. Without communication, each agent would follow the FSA, and the white agent would move in a straight line, finding no food.

With communication, however, the story is quite different. In 700 generations, GNARL discovered a pair of agents (from a population of 50) which had learned to communicate the presence of food. Figure 47b shows the case just as the black agent reaches the food. Figure 47c shows recruitment: the black agent's strong signal, due to the food, attracts the white agent. Figures 47d and e show both agents are emitting high signals while eating. Between Figures 47f and i, the black agent finishes his "half" of the food, and is recruited to help the white agent finish what is left.

Figures 48 and 49 show the agents' signals which allow them to produce this behavior. These and similar figures describe all there is to know about a particular agent. The "behavior graph" now represents the agent's action in terms of the architecture of Figure 36: zero indicates the agent is following the Jefferson FSA (Jefferson, et al., 1991); a value of $i$ indicates that the agent is pursuing the gradient of signal $i-1$.

The fact that only one communication channel was involved in the first experiment simplifies the interpretation of Figures 48 and 49. The black agent has learned to correlate the magnitude of its output signal with the presence of food. Of course, this correlation
Figure 47: Scenes of evolved communication, 2 agents, 1 communication channel, no noise. Primitives are "Follow FSA" or "Follow signal." (a) Initial positions: neither agent can sense food; (b) The black agent just reaches food, time is t=20; (c) Recruitment - first agent attracting the second, t=40; (d-g) Scenes at t=60, 80, 100, 140, respectively; (h-i) Recruitment again, though now in reverse, t=180 and 200, respectively.
Figure 48: Profile of the black agent (200 time steps). Behavior is either "Follow FSA" (low) or "Follow Signal" (high). This agent has learned to correlate the magnitude of its output signal with the presence of food.
Figure 49: Profile of the white agent (200 time steps). The point where recruitment occurs (Figure 47c) corresponds to the first spike in the behavior profile of this agent. When this agent reaches food (between Figures 47c and d), the behavior reverts to following the FSA.
would be meaningless without a suitable response from the white agent: when the white agent "hears" a large input signal, it follows the signal to find food.

I chose this case as a demonstration for several reasons. First, snapshots easily capture the evolved communication scheme: larger circles imply a higher signal. Second, the evolved language is fairly intuitive: each agent "yells" when it finds food by increasing the strength of its output signal; upon "hearing" such a signal, the second agent follows it to the source of food. I have also observed other implementations of the same behavior, e.g., "Yell constantly when you’re searching for food, but then grow quiet when eating." In this case, agents learn to respond to silence. But such constant signalling behavior by no means exhausts the possible means of communication.

5.5.2 Experiment 4: Oscillatory Communication Schemes

In contrast to the cases described above, the next example shows how oscillatory communication schemes may evolve. For this experiment, I used the same food distribution, increased the number of agents to three, and retained a single hidden unit for each agent. To investigate how the agents would respond to noise, I gave them two communication channels, the first clear \((u_{I}=0)\), the second noisy \((u_{I}=10)\).

Figure 50a shows the initial environment. The circles reflect the strength of signal 0. I omit signal 1, transmitted along a noisy channel, because it is not used by the agents (more on this below). After just one time step, the signals along channel 0 have shrunk to their size in Figure 50b. In the absence of food, signals in this channel oscillate between these two extreme values.
Figure 50: Scenes of evolved communication, 3 agents, 1 communication channel, no noise. Primitives are "Follow FSA" or "Follow signal." (a) Initial condition, circles denote signal 0; (b) After one time step, signal 0 has shrunk to its minimum value. It oscillates between the two extremes when no food is present; (c-i) t=25, 50, ..., 300.
Figure 51: Agent paths for the 3-agent case. Dots indicate path of agents (food and signals have been removed for clarity). (a) Agent 0 recruiting the others. After food has been consumed, agents 1 and 2 stay together, but agent 0 strikes out on a different path.
Because overall behavior is difficult to discern from just snapshots, Figure 51 abstracts just the agents' paths from Figure 50. Figure 51a shows recruitment by agent 0; Figure 51b shows that recruitment is not permanent: when the food has been consumed, agent 0 strikes out on its own.

Figures 52-54 show the signals by which this behavior is accomplished. Figure 52 gives the profile of the black agent over the run. Note how its output signal 0 oscillates in the absence of food. Figure 53 shows the profile of the white agent throughout the run. The lack of oscillation in the black agent’s output is enough to turn the white agent towards the food. (The 5 spikes in the behavioral profile indicate “Follow signal 0” behavior.)

The gray agent, however, is slightly different (Figure 54). Note the oscillation in its behavior, as it alternates between following the gradient of signal 0 and following the FSA. At first glance, this seems incorrect, because the inputs to agents 1 and 2 look identical, but their output behaviors are very different. The problem, however, is simply one of scale. Figure 55 zooms in on the first 50 time steps of the signal 0 input to agents 1 and 2. It is the phase difference between these two signals which is responsible for the difference in the agents’ behaviors.

From the agent profiles (Figures 52-54), it appears that the evolved agents are relying solely on channel 0, the clear channel. To test this, I blocked the agents’ signals by shunting the channel with various constant values. In all cases, removal of channel 0 drastically reduced fitness, yet the removal of channel 1 failed to hamper the search behavior of the agents, confirming my expectations that the agents had learned to ignore the noisy channel.
Figure 52: Profile of the black agent (300 time steps). This agent has learned to correlate oscillation of its output signal 0 with the presence of food.
Figure 53: Profile of the white agent (300 time steps). The five spikes in behavior indicate points where the agent follows signal 0, as can be seen in Figure 50. Because the agent perceives no food during this time, the resulting behavior occurs due to the agent's input signals.
Figure 54: Profile of the gray agent (300 time steps). Although its initial inputs (food & signals) look identical to that of the white agent, this agent's initial behavior oscillates between "Follow food" and "Follow signal." The difference is resolved in Figure 55.
Figure 55: Magnified view of the first input signal of agents 1 and 2 (50 time steps). The white agent's input begins oscillating between .03 and .04. The gray agent's input begins oscillating between .06 and 0. It is not the magnitude, but the difference in phase which is responsible for the agents' different behaviors.
5.5.3 The Evolution of Communication

The previous subsections focused on how the evolved systems of signals and responses allowed the agents to communicate the presence of food. This section, in contrast, focuses on an orthogonal question: how the system of communication evolves.

Section 5.5.1 described an evolved communication scheme in which an agent $A_0$ signalled the presence of food by "yelling," and agent $A_f$ responded by to the louder signal by moving towards the food. As stated earlier, this solution was discovered in 700 generations. To determine how it came about, I compared $A_0$ and $A_f$ with their parents in generation 699, and then their grandparents, etc., tracing their ancestry all the way back to randomly-generated agents in the initial generation. My first step in analyzing this lineage was to determine if evolution proceeded smoothly or by phase transition (Huberman and Hogg, 1987; Pollack, 1991). To do this, I compared gross characteristics of the agents over the course of the run: fitness, number of links in the networks, and total absolute value of the network weights (Figure 56). I discovered a phase transition before which fitness was low and gross characteristics varied widely, and after which fitness was high and gross characteristics were fairly stable.

I next moved from gross characteristics to specific agent features; specifically, I began examining the networks which comprised the agents. Because the agents were fairly stable towards the end of the run, I plotted only through generation 475 (the location of the arrow in Figure 56). Each network consisted of a maximum of 20 links in a 5x4 matrix ($w_{00}$ - $w_{43}$). This result is split across Figures 57 and 58. Once again, I observed a phase transition in the weight values.
Figure 56: Gross characteristics of the ancestors of the agents which demonstrated a constant communication scheme (Section 5.5.1). This figure shows generations 0 - 700. (The arrow denotes generation 475, used later.)
Figure 57: Evolution of network weights $w_{00} - w_{21}$, generations 0-475. Non-links are shown as a zero weight.
Figure 58: Evolution of network weights $w_{22} - w_{43}$, generations 0-475. Non-links are shown as a zero weight.
Unfortunately, just seeing the weight changes in the networks over time is of little help in understanding how the varying structure effects an increase in fitness. I proceeded with a functional analysis as follows. The key to the solution in the agents of generation 700 is that they have learned to communicate the presence of food. In particular, if neither agent has discovered the food patch, both agents search; if one agent has discovered the patch, it communicates this knowledge to the other. Figure 59, where the agents are taken from the final solution at generation 700, illustrates these two critical points in the task. To determine how the agents evolve functionally, I compared how they behave in these two situations. Specifically, I compared four features: the output signal of the black agent before it encounters food, the response of the white agent to this signal, the output signal of the black agent after it has discovered the food patch, and the response of the white agent to this signal. Furthermore, I graphed the absolute difference in values between the output signals of the black agent in the food/no-food cases.

Figure 60 shows the results. Let the black agent be 0; the white agent be 1; the food signal of agent 0 be \( S_f \), and the no-food signal of agent 0 be \( S_n \). Behaviorally, the critical aspect of the figure is the response agent 1 to \( S_f \). The phase transition in fitness corresponds exactly to a phase transition in this behavior.

Why does it take 475 generations for these phase transitions to occur? From the middle graph, one can observe a non-linear increase in \( |S_f - S_n| \), representing a distinction in the food/no-food cases by agent 0. Interestingly, however, the phase transition in fitness is not forthcoming despite many large spikes in this value, since two conditions must be met for the jump in fitness:
Figure 59: Snapshots of the agents in their environment at t=10 and t=30 (cf. Figure 47). The agents come from the final solution at generation 700. The particular time steps, chosen arbitrarily, simply capture the situation both before and after the black agent finds the food.
Figure 60: Functional difference of the agents over time (generations 0 - 475). Two events precede the phase transition in fitness: a distinction in the food/no-food signals by agent 0, and a recognition of this distinction by agent 1.
C₀ – agent 0 must effect a distinction between Sᵣ and Sᵣ.  
C₁ – agent 1 must respond to this distinction appropriately.

Consequently, search is hindered by two factors. First, C₀ must logically precede C₁, and second, the simple fitness function used for this task (number of pieces of food consumed) offers no reward for satisfying C₀ if C₁ is violated.

5.6 Remarks

I began with very few assumptions about the nature of communication, essentially stripping away the information-theory veneer that has made previous systems easy to understand. First I replaced the engineer with evolutionary search. Second, I eliminated discrete events and allowed the agents to modify channels with continuous values. These assumptions did not prevent solutions to the modified Tracker problem; in fact some novel approaches were discovered. I was able to evolve agents which demonstrated such task-specific behaviors as recruitment.

In the next chapter, I will discuss the implications of these results.
CHAPTER VI

REMARKS

Suppose, for example, that we adopt the intentional stance toward bees, and note with wonder that they seem to know that dead bees are a hygiene problem in a hive; when a bee dies its sisters recognize that it has died, and believing that dead bees are a health hazard and wanting, rationally enough, to avoid health hazards, they decide they must remove the dead bee immediately. Thereupon they do just that. Now if that fancy an intentional story were confirmed, the bee-system designer would be faced with an enormously difficult job. Happily for the designer (if sadly for bee romantics), it turns out that a much lower-order explanation suffices: dead bees secrete oleic acid; the smell of oleic acid turns on the “remove it” subroutine in the other bees; put a dab of oleic acid on a live, healthy bee, and it will be dragged, kicking and screaming, out of the hive.

– Daniel C. Dennett¹

6.1 Introduction

Up until this point, a subtle inconsistency has pervaded the description of my work. I have focused on the “evolution of communication,” but I have described my agents as evolving a “structural coupling” between their internal dynamics and the dynamics of their environment. But is this really communication? In this chapter, I summarize my work, and approach the nature of communication from a different perspective.

6.2 Dissertation Review

Let me first briefly review the ground I have covered. I began by introducing the problem of communication: given a set of agents and a task, how does one determine a communication scheme which aids the agents in their performance of the task? My first approach was to use Brooks' behavior-based design as an agent representation. The result, described in Chapter III, was Addam, a connectionist agent whose modules interacted to solve a task. With Addam, I proposed a particular training algorithm to determine appropriate modules; namely, using scenarios and a modified version of backpropagation. In terms of Winograd and Flores (1986), Addam showed how a behavioral decomposition could be used to attain a structural coupling between the agent in its environment.

Before endowing Addam-like agents with the ability to communicate, however, I became troubled by two particular aspects of the work. As with Brooks' approach to behavior-based design, Addam assumed an *a priori* behavioral decomposition, and an *a priori* agent structure. My next step was to try to eliminate these assumptions.

To do so, I adopted a somewhat simpler environment, the Tracker task (Jefferson, et al., 1991). Chapter IV proposed GNARL, a search technique motivated by evolution. As an alternative to backpropagation, GNARL required very little information about the task: no behavioral decomposition, no training set, and no *a priori* network structure. GNARL required only a fitness function to differentiate good solutions from poor solutions, and used evolutionary search over both weight values and network structure.

Given the success of GNARL in simple environments, I tried to adapt it to the more difficult task of evolving communication. Unlike the deterministic, single-agent environment of the original Tracker task, evolving communication over noisy channels
involves both a group of agents and non-determinism. Chapter V showed that evolutionary search still performed well in this situation. After resolving a problem with primitives, a variety of communication schemes were evolved which allowed the agents to perform well on the task. Although these schemes were evolved with regard only to overall task performance, each can be viewed in retrospect as providing two separate capacities: the ability to generate signals, and the ability to respond. An agent's signals lack any meaning whatsoever when removed from the context of the other agents. In terms of Winograd and Flores (1986), the agents have evolved a common structural coupling, i.e., the interrelated structures of the agents allowed interlocking patterns of behavior. But even more can be said about the agents' signals and responses.

6.3 Beyond Structural Coupling

Structural coupling provides a unifying metaphor for the way in which agents adapt to their environment. This relationship is present in the design of Brooks' agents (Figure 6, p. 24), the interplay of behavioral modules of Maes (Figure 7, p. 27) or Mataric (Figure 10, p. 34), and, less graphically, in all the works discussed in Chapter II. Structural coupling is adaptation, but with a slightly different emphasis. Saying that "an agent adapts to its environment" implies that the environment is static; "structural coupling" emphasizes that the environment, too, is constantly changing, and real adaptation occurs when an agent modifies its behavior to achieve its goals.

The term is particularly useful because, unlike other similar terms ("learning," "training," "adaptation," "evolution," etc.), structural coupling has very few preconceptions associated with it, but the novelty of the term should not be used to hide the fact that other metaphors apply. In particular, the omnipresent search metaphor of AI is
appropriate: structural coupling is the goal; evolution provides a means of search. In this light, one might think that labeling the evolved system of signals "communication" is merely creative naming (McDermott, 1981). But I claim that my agents really are communicating, in the sense that I describe below.

6.3.1 Levels of Explanation

In Chapter III, I showed how various levels of explanation (Chandrasekaran and Josephson, 1993) could be used to construct an agent with multiple interacting behaviors. Can "communication" be interpreted as another level of description?

In his discussion of mind, Dennett treats the vocabulary of beliefs, desires, and the like as a level of explanation used to describe the mental world, and labels this level the intentional stance. Adopting such a stance allows one to obtain a "better" description of a system, where "better" is realized in terms of predictive power or generality (Dennett, 1987, p. 139).

So what gains in predictive power or generalization accrue by assuming that our agents are indeed communicating? In repeating the experiments, I showed how each time a set of agents evolved a communication scheme which somehow helped perform the task. Sometimes they evolved the "scream at food" communication scheme; other times they evolved "be quiet when eating" scheme; still other times they evolved "oscillate until finding food" scheme. Although these schemes appear quite different, an underlying commonality binds them together: namely, in the types of information the agents transmit, receive, and use. In all cases, the agents have learned to communicate to each other the presence or absence of food, the most critical object in their fitness environment.
Interpreting the signals as communicating information allows not only this generalization, but also the prediction that any evolved communication scheme will convey similar information, regardless of its surface appearance.

6.3.2 The Communicative Stance

I claimed earlier that the agents do indeed communicate. Appealing to information processing almost substantiates my claim, but someone still might argue that calling the signals communication is still just my fanciful imagination. To reply to this criticism, I appeal to an analogy by Dennett. Consider the distraction display of birds. When a predator approaches a bird's nest, the mother bird will feign a broken wing, leading the predator away from the nest's helpless chicks. Adopting an intentional stance towards the mother bird, we can hypothesize her feelings upon seeing a predator:

I'm a low-nesting bird, whose chicks are not protectable against a predator who discovers them. This approaching predator can be expected soon to discover them unless I distract it; it could be distracted by its desire to catch and eat me, but only if it thought there was a reasonable chance of its actually catching me (it's no dummy); it would contract just that belief if I gave it evidence that I couldn't fly anymore; I could do that by feigning a broken wing, etc. (Dennett, 1987, p. 258).

Alternatively, however, the bird's behavior might be a simple stimulus-response mechanism:

Here comes a predator; all of a sudden I feel this tremendous urge to do that silly broken-wing dance. I wonder why? (Dennett, 1987, p. 258)

For its low-level nature, Dennett terms this latter explanation a "killjoy," and goes on to say:

But suppose it turned out that the killjoy interpretation was closest to the truth; the bird has a dumb tropism of sorts and that's all. Would we
thereupon discard the label “deceptive” for the behavior? Yes and no. We
would no longer credit the individual bird with a rationale of deception, but
the rationale won’t just go away. It is too obvious that the raison d’être of
this instinctual behavior is its deceptive power. That’s why it evolved. If we
want to know why this strange dance came to be provokable on just these
casions, its power to deceive predators will have to be distilled from all
the myriad of other facts, known and unknown and unknowable, in the
long ancestry of the species. But who appreciated this power, who
recognized this rationale, if not the bird or its individual ancestors? Who
else but Mother Nature herself? That is to say: nobody. Evolution by
natural selection “chose” this design for this “reason” (Dennett, 1987, p.
259).

This, finally, is the reason my agents are truly communicating. Initially, the agents
form a cacophony, each outputting signals in its own particular jargon, and each responding
to signals in its own particular way. Over time, structure emerges from this babbling as the
agents learn to convey particular information about their environment. But without the
“communicative stance,” there is no good explanation as to why all the various
communication schemes appear similar at the information processing level. Why do the
agents consistently learn to convey particular information to each other? The answer lies in
the functionality of that information for the task. The communication schemes appear
similar at the information processing level because the evolutionary model consistently
selects agents whose systems of signals and responses convey information which aids in
task performance. Other systems of signals and responses were not selected because they
lacked this functionality. Adopting the communication stance allows one to summarize this
by saying “the agents are communicating in the sense of conveying to one another task-
specific information.”
6.4 Beyond the Communicative Stance?

I began this dissertation with the claim "Task-specific communication schemes can be evolved for cooperative multi-agent systems." This claim has now been substantiated, at least for the case of the multi-agent Tracker task. In this section, I move on to more speculative thoughts on the implications of my results.

6.4.1 Relativized Semantics

In discussing the nature of grammatical operations, I have restricted myself to syntactic and phonological examples, avoiding questions of semantic interpretation. If a grammar is to characterize the full linguistic competence of the speaker-hearer, it must comprise rules of semantic interpretation as well, but little is known of any depth regarding this aspect of grammar (Chomsky, 1968, p. 49).

Over 25 years after Chomsky's observation, the boundary between syntax and semantics remains murky, in large part because semantics itself is an exceedingly difficult concept. Within AI, progress has been made on the syntactic end of the dichotomy (e.g., Winograd, 1983, Language as a Cognitive Process: Volume I: Syntax), but Volume II does not appear to be forthcoming anytime soon. In this section, I propose a new approach to the study of meaning.

Consider the agents in experiment 3 (Section 5.5.1). They learned to generate signals of varying magnitude to indicate the presence of food. From the perspective of one of these agents, the signal "High, high, high..." means that food is present, while the signal "Low, low, low" means no food has been found. In other words, the agents have developed a language with meaning relative to the task domain; consequently, I propose that these

2. Indeed, it is unlikely to appear at all, as Winograd’s views have shifted significantly. (See Winograd and Flores, 1986).
signals possess a semantics, which I call relativized semantics. Furthermore, I propose the following:

Language study should shift its focus from syntax to semantics. Investigating the emergence of relativized semantics within restricted environments may offer insights into the semantics of natural language.

Language has long been defined as "a particular relationship between sound and meaning" (Chomsky, 1968, p. 15), but research focuses on syntax. Turning this situation around by granting semantics a position of primacy offers the following benefits:

- Intuitive appeal of tackling the harder problem first. It seems unlikely (at least to me) that the solution to semantics will arise as an extension of the study of syntax. Putting semantics first explicitly recognizes this fact.

- Resolution of linguistic and cognitive competence. One problem with studying linguistic competence (Chomsky, 1965) in isolation is that ignores the related issue of cognitive competence, i.e., the fact that "a child has to have something to say before he expresses it" (Herriot, 1987). Studying relativized semantics of simulated agents within restricted domains offers a way to integrate these two preconditions for language development.

- Insights into evolution of semantics. Vygotsky (1962) pointed out that word meanings evolve over time, and that social interactions are responsible for language development. My approach offers the option of looking at the evolution of not only the agents (e.g., Figures 57 and 58), but also the syntax (e.g., Figures 48 - 49) and semantics (Figure 60), as I did above.

Other work in AI stresses the primacy of semantics over syntax (Schank, 1975), but does not address the evolution of semantics. Chomsky goes even further, explicitly turning away from evolution as a means of studying language:

The examples of animal communication that have been examined to date do share many of the properties of human gestural systems, and it might be reasonable to explore the possibility of direct connection in this case. But
human language, it appears, is based on entirely different principles. This, I think is an important point, often overlooked by those who approach human language as a natural, biological phenomenon; in particular, it seems rather pointless, for these reasons, to speculate about the evolution of human language from simpler systems - perhaps as absurd as it would be to speculate about the "evolution" of atoms from clouds of elementary particles.

As far as we know, possession of human language is associated with a specific type of mental organization, not simply a high degree of intelligence (Chomsky, 1968, p. 62).

Even granting that human language is discontinuous with that of other primates does not justify turning away from computational models of language evolution as a source of linguistic insight.

6.4.2 The Role of Syntax

The way the syntax of communication evolves is particularly illuminating, especially when contrasted with the discrete communication systems of Section 2.6. Consider the inputs to the white agents in experiment 3 (Section 5.5.1) and experiment 4 (Section 5.5.2), shown together in Figure 61. At no time is the input stream partitioned, normalized, or recognized; it simply modulates the behavior of the agent. But the agents of the two experiments develop radically different languages: the first is sensitive to the input signal's magnitude; the second is sensitive to the signal's phase.

If we label the constant and oscillatory communication schemes "languages" $L_0$ and $L_1$, respectively, one may well wonder if there exist corresponding grammars which describe these languages. The answer is yes and no. First consider $L_0$. Based on observation of Figures 48 and 49, when an agent wants to communicate the presence of food, it outputs a "food phrase," consisting of a single high signal. Similarly, to communicate the absence of food, it outputs a "no-food" phrase, consisting of a single low signal. The rules of
Figure 61: Comparison of input streams across experiment 3 (Section 5.5.1) and experiment 4 (Section 5.5.2) for the evolved white agent, t=0-50.

Grammar $G_0$ (Figure 62) capture this behavior. The case for $L_1$ (Figures 52 - 54) is similar, although now “sentences” are of length 2, as shown by $G_1$ (Figure 62).

But how well do $G_0$ and $G_1$ really capture the communicative behavior of the agents (Figures 47 and 50)? Ignoring the issue of continuous signals, $G_0$ seems to capture $L_0$ relatively well. But $G_1$ fails to capture $L_1$ in the end of the run, where the white and gray agents head southward “muttering to each other” (Figures 50h, 50i, 51b, 53, and 54). In this case, the grammar fails to capture an important communicative behavior of the agents.

Two resolutions are possible. First, I may have been too cavalier in devising $G_1$. Perhaps the grammar is too simplistic, and a more complicated one would produce $L_1$ exactly. I have no doubt that such a grammar exists, but I reject this possibility because the “deviant” behaviors of the agents described above do not seem to be deviant at all – such
couplings of signals are commonplace, and accounting for all the special cases in the grammar would defeat the goal of capturing the language concisely.

The second resolution is that actual communicative behavior of the agents simply approximates that of the ideal grammar $G_1$, as in Chomsky's competence/performance distinction (Chomsky, 1965). I reject this option as well, since in my situation, I see no compelling reason to grant the grammar a position of primacy over the actual behaviors of the agents.

It is appealing to attempt to dismiss syntax of natural languages similarly, but unfortunately this is not possible. The same syntax which limits the potential structure of a natural language is also responsible for the wealth of structures in the language. In fact, in contrast to traditional approaches to language study, the most difficult problem with relativized semantics will be finding something like syntax which allows compositionality (Chandrasekaran, 1990) within the evolved languages, in a way which leads to the generative capacity (Fodor and Pylyshyn, 1988; Pollack, 1990). But without further study,

---

**Figure 62:** Two grammars, written to capture the language of the agents in experiments 3 (Section 5.5.1) and 4 (Section 5.5.2), respectively. $S$=sentence, $FP$=food-phrase, $NP$=nofood-phrase, $V$=value.
semantics remains a promissory note which keeps accruing interest, and the debt will never be paid by continuing to focus on syntax alone.

6.5 Conclusions

The previous section was highly speculative, but I wish to end on firmer ground. In this work, I looked for commonality among evolved signals and responses, adopted the stance that the agents were communicating, and arrived at information processing. The destination itself is familiar enough; from my perspective, the interest lies in the route by which I arrived. I eliminated Brooks' behavioral decomposition; I eliminated connectionists' assumption of fixed architecture. I assumed only an evolutionary search algorithm operating over network structures, and arrived at communication.

Much work still needs to be done; in particular, there are many side roads from the path I took. How will evolution of communication be affected by further varying the number of agents, the width of the communication channel, the amount of noise in the channels, or the task itself? How can the fitness function be altered in a way that gives positive feedback for progress towards communication, but in a way that does not solve the task? How can the agents develop a richer syntax on top of their relativized semantics, so that something approaching generative capacity might be achieved? What relationship (if any?) does the evolution of relativized semantics have to the evolution of natural semantics?

Despite the number of open questions, the results so far are promising. In criticizing the state of affairs in 1976, McDermott describes communication between computer programs:
At the current stage of research, it is ridiculous to focus on anything but raw communication of information... Packaging and encoding of the information are usually already done. Ambiguity is avoidable. Even brevity is unimportant (at least for speed), since a huge structure can be transmitted by passing an internal name or pointer to it shared by sender and receiver (McDermott, 1981, p. 151).

While a set of agents evolving a way to signal the presence of food is a far cry from natural language, the evolved communication – and I really mean communication in the sense of Dennett – is also a far cry from raw communication of information, where programs communicate by passing pointers. And I feel it is a step in the right direction towards a true understanding of the evolution of communication.
LIST OF REFERENCES


