Machine Learning 3: 161–192, 1988
© 1988 Kluwer Academic Publishers – Manufactured in The Netherlands

Classifier Systems that Learn Internal World Models

LASHON B. BOOKER (BOOKER@NRL-AIC.ARPA)
Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC 20375-5000, U.S.A.

(Received: March 11, 1988)
(Revised: July 22, 1988)

Keywords: Classifier systems, genetic algorithms, internal models, instinctive behavior

Abstract. Most classifier systems learn a collection of stimulus-response rules, each of which directly acts on the problem-solving environment and accrues strength proportional to the overt reward expected from the behavioral sequences in which the rule participates. GOFER is an example of a classifier system that builds an internal model of its environment, using rules to represent objects, goals, and relationships. The model is used to direct behavior, and learning is triggered whenever the model proves to be an inadequate basis for generating behavior in a given situation. This means that overt external rewards are not necessarily the only or the most useful source of feedback for inductive change. GOFER is tested in a simple two-dimensional world where it learns to locate food and avoid noxious stimulation.

1. Introduction

> People talk fondly of computer programs that will start with some fundamentals and acquire all the knowledge needed by some natural sequence of learning, experiencing the environment in which it must function. Very little effort gets spent studying what it would take to accomplish this, perhaps because there is implicit realization that the task is harder than it might seem (Norman, 1981, p. 284).

As artificial intelligence research has focused on real-world problems, one of the most important issues to emerge has been the need to flexibly represent and utilize a large repertoire of knowledge about the problem domain. The kind of knowledge required includes more than specific expertise about the problem-solving task. It also includes background knowledge about the overall task environment – the sort of general assumptions, facts, and methods usually associated with ordinary common sense. In a complex task domain, a well-organized body of such facts and methods can provide an important context for deciding which problem-solving assumptions are reasonable or assessing which new facts are important. This source of power is especially important to machine learning research (Carbonell, Michalski, & Mitchell, 1983). Task-specific knowledge helps to focus the application of inductive inference methods by constraining the choices about what to learn and when to learn. A broad base of general knowledge provides a framework in which new information can be understood, making it easier to integrate that information with what is already known.

An obvious question for researchers in machine learning is how to bootstrap enough knowledge into the system to realize these advantages. Some have argued for long-term rote efforts to construct large knowledge bases having encyclopedic scope (Lenat, Prakash, & Sheperd, 1986). Others stress the critical role of empirical knowledge derived from a system's experience with its environment (Simon, 1983). Both methods of acquiring knowledge about the environment are likely to be important. In either case, one of the fundamental research problems for machine learning is understanding how to organize, construct, and modify representations of the task environment (Scott, 1983; Holland, Holyoak, Nisbett, & Thagard, 1986; Michalski, 1986). This paper describes machine learning research that uses *genetic algorithms* (Holland, 1975) and *classifier systems* (Holland, 1976) to acquire empirical knowledge about an environment.

Constructing representations of an environment is particularly difficult when the correspondence between representation and reality cannot be taken for granted. Most real-world environments can be very uncooperative from the standpoint of providing salient, timely, complete, or unambiguous information. A learning system trying to cope with such an environment can be faced with several challenges:

- Categorizing situations given uncertain and perpetually novel input;
- Predicting future states of a complex environment on the basis of incomplete knowledge;
- Determining how a situation is relevant to the attainment of ill-defined goals given only sparse payoff or reinforcement;
- Deciding on a course of action when the requirements for behavior are continual and perhaps even real-time.

These difficulties impose important constraints on the kinds of learning strategies that can be successful.

For example, consider how a concept-learning task is affected by these conditions. Whenever the stream of input data is complex and the important concepts to be learned are not specified in advance, a learning system must decide which inputs to group into categories as well as how each category is defined. These decisions require a pragmatic, incremental approach to learning that differs in important ways from most traditional concept-learning methods (Lebowitz, 1987). Similar considerations arise when learning tasks are closely coupled with goal-directed problem-solving activity (Holland et al., 1986). Uncertainty surrounding inputs and goals makes categorization difficult, and it complicates both the association of actions with categories and the discovery of action sequences required to attain goals. Under these circumstances, a learning system must actively generate opportunities for inductive change by using the verification or falsification of its own predictions as a feedback signal (Holland, 1986).

The best way to cope with environmental complexity and uncertainty is to maintain a *model* of the environment that summarizes what to expect and suggests appropriate actions. As Craik (1943, p. 61) put it, a system having an internal model of external reality

> ...is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way react in a much fuller, safer, and more competent manner to the emergencies which face it.

This kind of model, together with feedback from the environment that allows continual adjustment of expectations to reality, subsumes a fundamental aspect of commonsense knowledge (Simon, 1983). Moreover, there are good reasons to believe that a model of the environment provides important constraints on the kinds of categories a system can acquire (Murphy & Medin, 1985), making the model a primary source of inductive bias. Since learning is the basic tool for building and maintaining such a model, it is clear that models of the environment are an important topic for machine learning research.

What are appropriate task domains for investigating the machine learning issues raised here? Applications related to mobile robots and autonomous vehicles are obvious candidates. Although there has been limited emphasis on learning in much of the practical work on mechanical devices, computational models of simple organisms and other autonomous systems in simulated worlds are free to make extensive use of learning techniques. In these domains a model is important not only as a source of inductive bias that directs and constrains learning; it is also important in helping to manage continuous problem-solving behavior and interaction with the environment. For this reason, learning must be flexibly integrated with the other mechanisms that help the system function successfully. In particular, learning must operate on the structure of knowledge representations, as well as the way in which those representations are used. These two aspects of information processing, in the context of purposeful behavior in an environment, fall squarely under the umbrella of cognitive science. Techniques for learning internal models of an environment can therefore be profitably studied in conjunction with cognitive models of functioning in that environment.

There have been several research efforts involving machine learning techniques and cognitive models of simple organisms. Doran (1968) describes a simple automaton that uses rote learning to model every possible state transition in its simple environment, and then uses that model to find its way back to a known location. Findler and Allan (1973) designed an organism that learns a world map of a dynamic, three-dimensional environment by rote. This map includes object attributes and information about goal attainment, as well as a basic summary of state transitions. Holland and Reitman (1978) demonstrate that a rule-based cognitive system can use genetic algorithms to learn the stimulus-response associations needed to obtain rewards in a simple maze. Sutton and Pinette (1985) show how a connectionist network can learn the stimulus-stimulus associations needed in a model by incrementally improving internal predictions of future states of the world.

The Holland and Reitman (1978) work has led to more extensive research along the same lines. Their rule-based cognitive model was the first experiment in the research on *classifier systems* (Holland, 1976). This paradigm is well suited for studying the acquisition of internal models because classifier systems can dynamically construct and modify rule-based representations. For example, Booker (1982) used a classifier system to model an organism that builds nontrivial cognitive structures based on experience and learns to respond appropriately to both attractive and aversive stimuli. Wilson (1985, 1987) has focused on a specialized set of cognitive tasks that he calls the *animat* problem: learning multiple disjunctive concepts incrementally under payoff. His animat was a classifier system that learned to find food and navigate around obstacles in a two-dimensional world. Although the classifier-system framework has broad implications for understanding learning and mental models in organisms and machines (Holland et al., 1986), the scope of experiments conducted so far has been relatively small.

This paper describes a particular classifier-system model of cognitive behavior, illustrating the current state of the art and pointing out the machine learning issues requiring further study. The goal of this research is to design machine learning systems capable of building functional "cognitive" models of realistic environments. The models are cognitive in the sense that they must include representations of categories and their associations with each other and with overt behavior. The models are functional in that the system must continually use its model to select an action. The next section contains a brief overview of classifier systems. In subsequent sections we give more details about what it means to learn an internal model, and we describe experiments demonstrating the potential of classifier systems to accomplish this goal. Booker (1982) presents these results in more detail.

2. Overview of classifier systems

A *classifier system* is a parallel, message-passing, rule-based system designed to permit nontrivial modifications and reorganizations of its knowledge as it performs a task. In the simplest version, all messages are fixed-length binary strings. The set of messages to be processed at any given moment is stored on a *message list.* Classifier systems process these messages using a finite population of *classifiers.* Every classifier is a fixed-length rule having the structure

$$t_1, t_2, \ldots, t_n \Longrightarrow m.$$

Each t_i is a fixed-length string called a *taxon* in an alphabet {0, 1, #}, which serves as an activating condition. A classifier can be activated only if each of its taxa matches a message on the message list, where a match requires that the 0's and 1's in the taxon be identical to the values at corresponding positions of the message. The # symbol is a "don't care" place holder that allows general conditions. Once a classifier is activated, it generates the message m and places it on the message list.

2.1 Basic organization

The typical classifier system is organized into three interacting subsystems (Holland, 1986): a *performance* system, a *credit assignment* system, and a *rule discovery* system. The performance system is responsible for interacting with the problem-solving environment and generating behavior. It consists of the following basic execution cycle for activating rules:

(1) Place messages from the input interface on the current message list;
(2) Compare all messages to all conditions and conduct a competition among relevant classifiers to determine which ones will become active;
(3) For each active classifier, generate one message for the new message list;
(4) Replace the current message list with the new message list;
(5) Process the current message list through the output interface to produce system output;
(6) Return to step 1.

Note that the performance system is designed to activate several classifiers in parallel. Classifier systems can use groups of rules to represent complex concepts, constraints, and problem-solving behaviors. Because individual rules must compete to become active, classifier systems have the flexibility to construct and modify the representation of a problem as problem solving proceeds.

The credit assignment system is responsible for evaluating each classifier and assessing how useful it has been in producing successful problem-solving behavior. This is a difficult problem in a system that uses many rules over several execution cycles to produce that behavior. It is made even more difficult when overt feedback from the environment is a rare event. The only realistic alternative for classifier systems is to evaluate performance for behavioral sequences in terms local to the classifiers that were involved.

The mechanism most often used for this purpose is the *bucket brigade* algorithm (Holland, 1985). Every classifier is assigned a quantity called *strength* that summarizes its overall usefulness to the system. Classifiers that are relevant to a message *bid* a small fraction of their strength to become active. The competition for becoming active is resolved probabilistically based on the size of these bids. An active classifier then pays its bid to the classifier that posted the message on the previous cycle. Each classifier thus participates in a transaction where it pays and receives strength. This repeated strength adjustment eventually leads to bids that predict the amount of strength a classifier expects to receive when it is activated. Classifiers active when payoff is available directly from the environment receive their increase in strength directly from the payoff. In this way, all rules activated during a behavioral sequence leading to external payoff eventually accrue strengths that predict the size of that payoff. This shuffling of strength between active classifiers on successive cycles is an effective way to adjust predictions about eventual goal attainment. When predictions do not involve external payoff, more direct prediction-based revision methods can be used.

The rule discovery system is responsible for generating plausible new classifiers that might yield better problem-solving performance. The mechanism used most often for this purpose is a *genetic algorithm* (Holland, 1975). A genetic algorithm is a general-purpose search procedure that uses sample-based induction (Holland et al., 1986) to conduct the search. The basic requirement for the procedure is that the elements of the search space be strings constructed of components or building blocks and that the elements can be ranked in terms of a solution-relevant preference. The algorithm draws an initial sample of elements from the space, then repeatedly selects and recombines building blocks from the current sample to construct elements of the next sample. The new elements are constructed using genetic operators such as crossover, inversion, and mutation. In classifier systems, the preference criterion for selection is based on strength and new classifiers replace low-strength classifiers in the current population.

A more detailed discussion of classifier systems, genetic algorithms, and how they relate to more conventional AI and machine learning methodologies can be found in Booker, Goldberg, and Holland (in press).

2.2 Implementation issues

This rather broad characterization of classifier systems leaves several questions unanswered about their implementation. Because the performance, credit assignment, and rule discovery systems are so tightly coupled, implementation decisions cannot be made arbitrarily. Systematic criteria for making these decisions, such as those available for basic genetic algorithms (Grefenstette, 1986), are the subject of ongoing research in classifier systems. However, guidance is currently available from an examination of successful architectures. We briefly review some of these implementation issues and how they have been successfully handled.

In looking at the basic execution cycle of a classifier system, the most obvious issue involves when rule discovery should be triggered. More subtle questions relate to the factors necessary for computing a bid, and the mechanisms needed to retain useful concepts when classifiers are constantly being added and deleted. Below we summarize many of the important decisions required for implementation.

Resource management. Because classifier systems work with a fixed-sized population of classifiers, the size and use of the limited classifier memory is an implementation issue. There must be some assurance that the system has enough memory to solve its problem. The following heuristic has proven to be a useful way to determine how much memory is required. First estimate the number N of rules or concepts needed to solve the problem. Then multiply N by the number C of classifiers a standard genetic algorithm (Grefenstette, 1986) would need to discover any one of these concepts in isolation. The result (CN) is a working estimate of the total number of classifiers the system should be given. Since the typical small problem requires a few hundred classifiers, data structures like digital search trees or discrimination nets are often used to efficiently determine which classifiers compete to become active.

It is also important to allocate the available memory so that efforts to discover one concept do not interfere with concomitant efforts for other concepts. The standard genetic algorithm searches for the single concept having the highest overall strength, but several modifications have been proposed to let it discover multiple concepts simultaneously for use in a classifier system. When the number of concepts is known in advance, a classifier system can be designed to have separate populations for each concept (Holland & Reitman, 1978). Alternatively, one can have a single population with predetermined partitions and restrictions on learning operators (Goldberg, 1983). If these precompiled solutions are not possible, dynamic memory management can be achieved by implementing simple mechanisms analogous to the speciation and niche competition found in biological populations (Booker, 1985; Wilson, 1985, 1987).

Triggering conditions. The primary concern about when to trigger rule discovery is ensuring that classifiers are fully evaluated before their strengths are used to select the building blocks for recombination. All successful implementations of classifier systems have made sure that the interval between rule discovery events is greater than the expected time required to evaluate a classifier. Rule discovery is usually triggered to coincide with payoff events, though in some systems an additional background rate of rule generation is used to help make the discovery process more robust (Booker, 1982; Wilson, 1985). It is important to note that the frequency of rule deletion is tied to the frequency of rule discovery. A high deletion rate can bias the system against rules that are activated and receive payoff less often than the average rule (Wilson, 1987).

Unrecognized situations. The basic execution cycle presumes that the system always has at least one classifier that is relevant to a message. Although there are some simple situations where this can be assured by starting with an initial population of very general classifiers (Holland & Reitman, 1978), in most cases provisions must be made for situations in which no rule is applicable. One approach is to insert the unrecognized message into the taxon of a new classifier, perhaps with a few #'s added, and to generate the rest of the classifier randomly (Holland, 1976; Wilson, 1985). An alternative is to identify rules that are partially relevant and use the rule discovery process to find building blocks for the desired new rule (Booker, 1982). The relative merit of these two approaches has not yet been determined. However, it is clear that a classifier system cannot succeed unless it learns something from unrecognized situations.

3. Learning viewed as model building

Learning in most implementations of classifier systems depends heavily on overt external "rewards" that indicate which inputs are goal-related and which behaviors are effective. This emphasis on external reward as the driving force for learning is very similar to the stance of the stimulus-response (S-R) learning tradition in psychology (Bower & Hilgard, 1981). The basic epistemological assumption is that all the system's knowledge is summarized by the strengths associated with given responses in given situations. S-R learning problems fit easily into the classifier-system framework because the rule discovery system builds associations between situations and responses, and because the credit assignment system refines strengths so that they can be used to select the correct action. The problem with a strict interpretation of this stance for learning in general is that it will lead classifier-system researchers into the same dilemmas that plagued the S-R psychologists: it will be difficult to account for latent learning that occurs without external reward, hypothesis-testing behavior, and any other "cognitive" behavior that is easily explained by positing explicit internal representations of goals.

An alternative formulation of learning processes is in terms of knowledge, goals, and purposes rather than collections of stimulus-response pairs. This emphasis follows the tradition of cognitive learning theories in psychology (Bower & Hilgard, 1981). The important epistemological assumption here is that the system maintains an internal model of the environment and how to function in it. Such a model describes predictive and associative relations between events as well as connections between events and overt responses. From the cognitive point of view, learning focuses on model building instead of strengthening rewarded S-R pairs.

The driving force for learning these internal representations is derived primarily from the system's routine functioning and problem solving. Learning is triggered whenever the system's model proves to be an incomplete, inconsistent, or otherwise inadequate basis for generating behavior in a given situation. Overt external rewards provide useful guidance about which events are important and where model-building efforts should be focused. However, such rewards are not necessarily the only or the most useful source of feedback for inductive change. As Holland et al. (1986) point out, a system that models its environment can use the outcome of model-based predictions about events and outcomes to guide induction. In this way, learning focuses more on avoiding surprise at events in the environment than on explicitly optimizing system rewards.

Samuel's (1959) early learning research on the game of checkers is an instructive example of the model-building approach. The only overt reward for checkers is a single bit of information (win or lose) available at the end of the game. Rather that trying to devise a clever and informative reward function or a complex credit assignment scheme, Samuel chose to model the problem-solving environment (i.e., the opponent). His learning system used the outcome of model-based predictions as feedback to improve the model and thus make it a more effective tool for choosing good moves.

Before classifier systems or any other machine learning paradigm can adopt the model-building approach to learning, the possibilities for induction must be sufficiently constrained. The most readily available sources of constraint are the environment and the learning system's problem-solving activity. When inductive mechanisms are tightly coupled with problem solving, learning takes place in a pragmatic context that delineates which inductions are important and reasonable (Holland et al., 1986; Laird, Rosenbloom, & Newell, 1986). A problem-solving environment is a helpful source of constraints to the extent that it is coherently organized. Coherent environments are characterized by discernible regularities, structured categories, and consistent laws governing state transitions. This organization provides a frame of reference in which it is practical to formulate and test inductive hypotheses. Regularities of this kind are characteristic of natural environments, and are routinely exploited by organisms as they learn about the world (Kaplan, 1982).

One cannot take for granted this kind of synergetic relationship tying together learning, problem solving, and the environment. It must be carefully incorporated into the specification of a machine learning problem. Many experiments in machine learning select an isolated problem to solve, together with a few example solutions or a helpful critic to provide feedback. This arrangement often does not provide a repertoire of experiences that is rich and complex enough to require dynamic construction, testing, and refinement of internal models. A key issue is how easily the information available in the environment can be extracted and exploited to uncover goal-relevant regularities. Information from a problem-solving environment must inevitably be presented to a machine learning system as a symbolic expression. However, if the structure of these symbols and expressions is not correlated with the structure of the objects they designate, the information in the environment may not be easily accessible.

Many learning systems use high-powered interpretive processes to determine which symbols are relevant in a context and how the symbols are related to each other. The difficulty of making such interpretations in a complex environment has led to doubts about the ability of artificial systems to function in realistic task domains (Dreyfus, 1972). The complexity of the mechanisms required to exploit environmental information can be substantially reduced by paying careful attention to the representation of the environment. As Boden (1977, p. 15) argues, when "... the likeness between sign and significate is rich and systematic, ... it can be exploited in using the symbolic representation as an aid to intelligent thinking about the thing symbolized." A systematic relationship between the structure of environmental symbols and the objects they designate can give a learning system a crucial advantage in coping with a complex and uncertain problem-solving environment.

4. A testbed for studying model building

Identifying regularities in structures and their relationships is the most fundamental kind of inference organisms must make in the real world (Bruner, 1957). Therefore, it seems reasonable to consider simulations of simple organisms as a way to study how machine learning systems can build and use internal models of their environment. A very basic problem-solving task for any organism is to locate food while avoiding dangerous or noxious stimulation. In the remainder of this section, we describe a class of simulated environments for experimenting with this task, together with a hypothetical organism and the input-output primitives needed to function in these environments. In subsequent sections, we show how a classifier system can use these primitives to learn to accomplish the task by building an internal model.

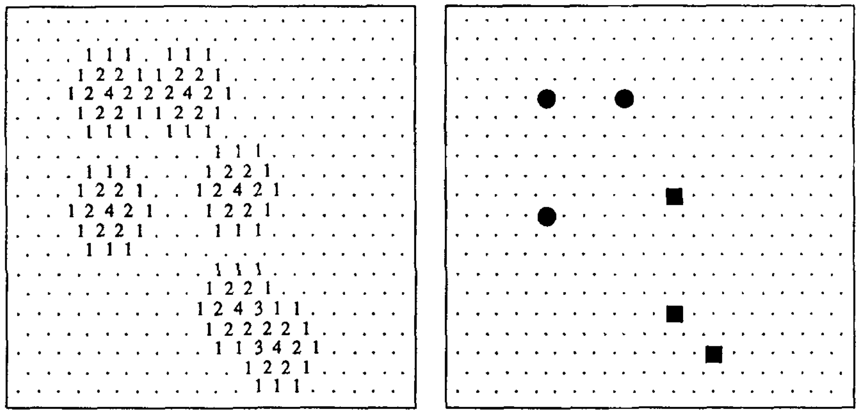

Figure 1. A typical distribution of objects and signal intensities in an environment. Each object (a square or circle), is located at the center of its stimulus aura and intensity falls off 30% per unit distance. The dots indicate loci where no signals are available.

4.1 Design of an artificial world

The proposed locomotion task requires an environment having two kinds of information: cues that orient behavior with respect to the relevant spatial patterns in the environment, and cues that convey information about object categories. Keeping in mind that exploitable regularities are the most important property from the standpoint of learning, a very simple artificial environment can be designed to have useful information of this kind.

The first decision that must be made regarding spatial information concerns the number of spatial dimensions. Locating objects in space requires knowledge about their direction and distance. It is easy to directly manipulate direction and distance parameters in two dimensions. Designing a three-dimensional world requires attention to details about surfaces and volumes that complicate localization in ways not relevant to the simple notion of locus needed here. Accordingly, we will limit our attention to environments having two dimensions. The environment is defined as a discrete hexagonal grid that is mapped onto a torus so that there are no edges. Stimulus signals from an object are available in a symmetric area around the object's location. The intensity of each signal is a scalar quantity that is highest at the object location and that falls off with distance uniformly in all directions (see Figure 1). A simplifying assumption is that all objects are sources of stimulus energy in a uniform medium. Signals from different kinds of objects do not interfere with each other, and signals from similar objects that can be detected at the same location combine additively. Thus, a straightforward intensity gradient is available to help orient behavior relative to an object's location.

The identity of a stimulus can be conveyed by associating a pattern category with each object. Hayes-Roth (1973) describes a "schematic" approach to characterizing pattern categories that compactly specifies multiple disjunctive concept structures. In the simplest case, this approach assumes that each pattern category can be defined by a single structural prototype or *characteristic*. Each such characteristic is a schema designating a set of features values required for category membership. Unspecified values are assumed to be irrelevant for determining membership. Disjunctive categories are handled by specifying one characteristic for each disjunct. Pattern generators based on the schematic approach generate exemplars by assigning the mandatory combinations given by one of the pattern characteristics and producing irrelevant feature values probabilistically. In this way, each exemplar of a category manifests at least one of the defining characteristics.

We can define categories having binary signals that are suitable as input for a classifier system as follows. Each characteristic is a string in the alphabet {1, 0, *}, where the * is a place holder for irrelevant features. A characteristic is a template for generating binary strings in the sense that the 1 and 0 indicate mandatory values and the * indicates values to be generated at random. Thus, the characteristic 1*0* generates the four strings 1000, 1001, 1100, and 1101. Stimulus signals for disjunctive object categories are generated by randomly selecting one characteristic to use as a template. Note that a new stimulus signal is generated every time the system samples the environment for input. This way of designating stimulus signals enhances the potential for environmental uncertainty, because the information about an object available at any given location is constantly changing. Moreover, it allows us to challenge the learning system with some of the most difficult categorization problems found in the machine learning literature, such as the Boolean 'multiplexer' concept (Wilson, 1987).

4.2 Design of a hypothetical organism

Having specified a class of environments affording the localization and identification of objects, we can now consider the kind of problem-solving behaviors that can exploit these environments. Here again, it is helpful to begin by looking at the way organisms accomplish these tasks in the real world. Tinbergen's (1951) model[1] of instinctive behavior in animals speaks directly to these issues. The simplest kind of instinctive mechanism relies on the environment to select and activate appropriate behavior. Tinbergen uses the term *innate releasing mechanism* to describe a "hard-wired" pathway between a perceptual unit that detects some significant stimulus or event and the behavioral routine that generates the appropriate response. The presence of the stimulus directly elicits or "releases" the behavior in question. An organism that relies exclusively on innate releasing mechanisms must wait for the environment to provide an action mandate. In a complex and uncertain environment, this would most likely be a long and dangerous wait. Not only might there be no stimulus sufficient

[1]We consider the principles of the model's organization here without endorsing Tinbergen's mechanisms, which require the accumulation and "draining away" of various impulses.

to activate a releasing mechanism, but several pathways might be activated at the same time, requiring the organism to make some kind of control decision.

In order to choose one pathway over another, an organism must be given criteria for deciding which inputs are most important in a given situation, and it must have the structural apparatus to selectively process and respond to the chosen stimuli. The most basic kind of control factors involved in modulating response selection in animals are called *motivational* factors. These factors depend on metabolic conditions, such as hormone concentrations and food or water deprivation levels. Their effect is to determine a *central motive state* that changes the relative effectiveness of stimuli to elicit behavior (Bindra, 1978). Tinbergen's model emphasizes that there are important differences in the roles of the motivational factors and releasing stimuli. A motivational factor selects for some behavioral goal by priming or otherwise increasing the readiness of appropriate motor complexes. Once a set of alternative behaviors has been primed, the current input configuration releases the one that is most appropriate. In this model, the priming is necessary for the releasing action to be effective.[2] This allows an organism to process information "off line" without immediately producing overt behavior, an important capability in any environment where there are very complex associations between actions and the state changes they produce.

Additional machinery is required to handle situations where no behavior is released or several actions are available for achieving a particular goal. Tinbergen proposes a hierarchically organized collection of *instinctive centers* as a useful way of managing these complexities of instinctive behavior. An instinctive center is a simple mechanism or agent that has a releasing stimulus and an associated action, just like an innate releasing mechanism. However, it has two potential sources of control inputs: the current motivational state and other instinctive centers higher in the hierarchy (see Figure 2). Centers at the same hierarchical level designate behaviors associated with different subgoals. Centers associated with the same subgoal at different levels are connected, so that generic behaviors are at the higher levels and more specific or stereotyped responses are at the lower levels.

The lowest-level agents are innate releasing mechanisms that may or may not have control inputs. Activation of an instinctive center, via the combined effect of facilitating control signals and a releasing stimulus, leads to priming of all directly subordinate centers. If none of these subordinate centers becomes active, the center releases its own behavior. Typically this will be some kind of search or exploratory behavior that strives to find a stimulus that will release one of the subordinate centers. The exploratory behavior associated with an active center is therefore a default response when control cannot be passed to a more specialized behavior. If more than one subordinate center tries to become active, priority is given to the center that is lowest in the hierarchy. If several centers within a hierarchical level try to become active, the conflict is resolved through a winner-take-all competition among the centers based on

[2]Behaviors related to certain defensive or aversive reactions must be effectively "ungated" so that the releasing stimulus can interrupt processing and reliably elicit a response. It is as if there is an ongoing motivation to deal with these stimuli.

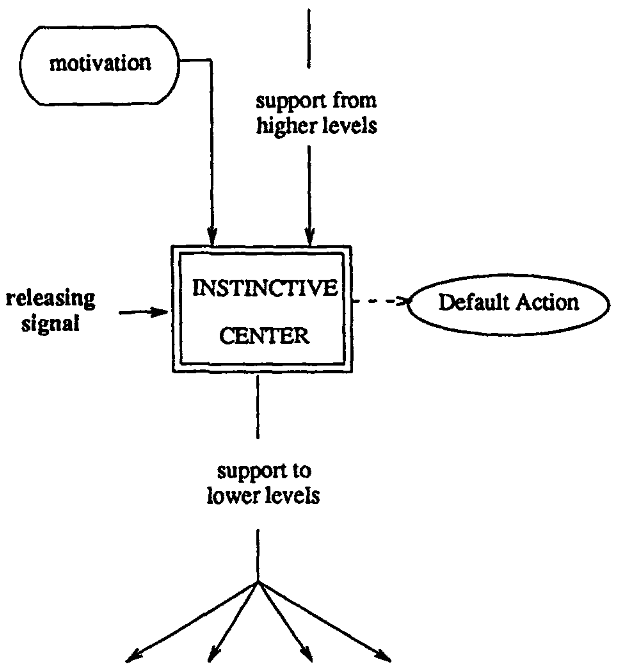

Figure 2. An instinctive center, the basic element in Tinbergen's (1951) model of the organization of instinctive behavior. Centers are linked together in a hierarchy. A typical center accrues support from motivational factors and from other centers higher in the hierarchy.

strength of activation. Thus, Tinbergen's representation for the organization of a major instinct is a complete hierarchy. The highest center in the hierarchy, which represents the overall goal, may or may not require a releasing stimulus to be activated.

We can use this framework as the basis for an organism that finds objects in our artificial world. First we specify the primitive motor actions:

- TURN – changes the direction of motion 60° to the right or left.[3]
- MOVE – takes the organism one step directly ahead.
- CONSUME – lets the organism nourish itself when in contact with a food resource.

The sensory interface is equally primitive. The organism is given detectors that can pick up only the signals directly ahead, to the right, and to the left. This restricted "retina" gives the organism a well-defined orientation and focuses its sensors on objects in its path. The organism also has a detector indicating

[3]In a hexagonal grid, the direction to neighboring points is always a multiple of 60°.

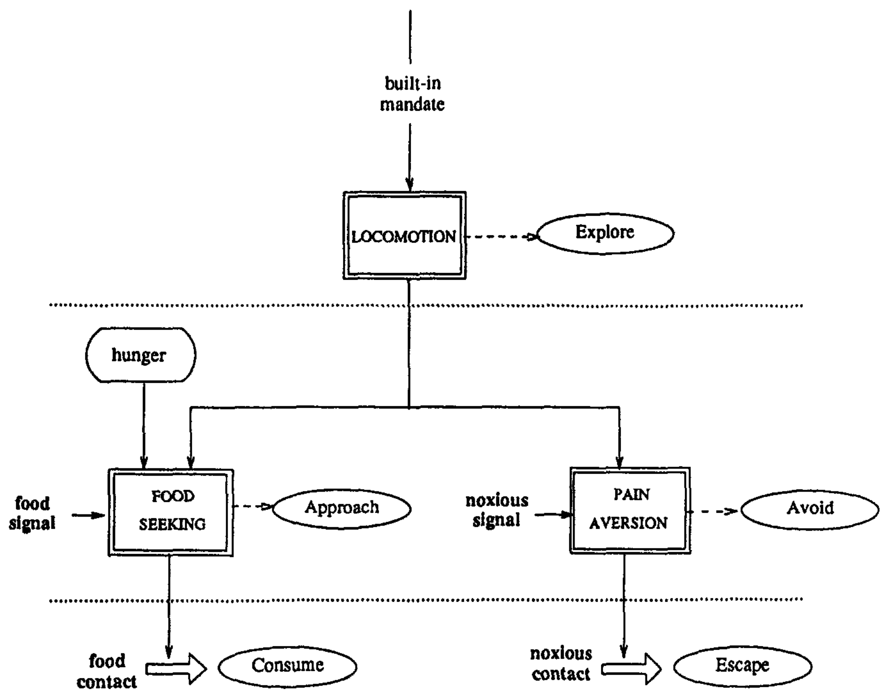

Figure 3. The control structure for the simple organism, shown here as an "instinct" following Tinbergen's (1951) model of instinctive behavior.

contact with objects, along with a motivational "need" for food that increases every time step but is satisfied when food is consumed.

These few primitives can be combined to produce goal-directed behavior. The necessary motor routines are derived from the primitive actions:

- ESCAPE – a random TURN, followed by a MOVE.
- EXPLORE – a random TURN, followed by a random number of MOVES.
- APPROACH – TURN to center the strongest intensity on the retina, then MOVE.
- AVOID – TURN away from the most stimulated part of the retina, then MOVE.

Figure 3 shows how one can organize these behaviors into a "locomotion instinct" for the hypothetical organism. The instinctive behavior is organized into three levels. At the highest level is a center responsible for overall control of locomotion. In our organism this center is always active. Whenever none of the lower-level components is active, the default motor response is EXPLORE. The middle level contains two centers subject to motivational control, only one of which can be active at any given time. The food-seeking center, when active, causes the organism to APPROACH food until contact is made and consumption is possible. The amount of facilitation for food seeking depends on the level of need for food. The pain-aversion center, which always has enough facilitation to be released by a noxious signal, causes the organism to AVOID noxious objects. At the bottom level are the two stereotyped responses CONSUME and ESCAPE. CONSUME requires facilitation from the food-seeking center and contact with food to become active. ESCAPE is released automatically by contact with a noxious object.

One can easily show that this organism finds food and avoids noxious objects. (Booker, 1982). As a simplification, the hunger level is implemented with a counter that is incremented every time step and reset to zero by CONSUME. In an environment containing 400 distinct locations, four food objects, and four noxious objects (see Figure 4), the organism never goes more that 80 steps without locating food once it is motivated to do so. By contrast, if one assumes that any deprivation interval exceeding 100 steps is lethal, a simpler organism using only EXPLORE to find food usually lives about 191 time steps. The goal-directed processing of stimulus signals is therefore a crucial advantage for surviving in this environment.

5. Implementing the organism as a classifier system

The hypothetical instinctive organism just described has been implemented using a specialized classifier system called GOFER. Classifiers learned from experiencing the environment are used as releasing mechanisms to control the food-seeking and pain-aversion instinctive centers. The classifier system thus must learn how to run about and "go for" what it needs, an ability that was programmed into the instinctive model. We assume that the organism has no *a priori* knowledge about which stimulus patterns are important, nor about why a stimulus might be relevant. Only contact with appetitive and aversive objects has any built-in significance for behavior. Contact with these objects is reinforcing in that the system receives an unambiguous signal that representations for those events should be learned. Activation of the internal representations of these events in other situations then becomes the criterion for releasing food-seeking or pain-aversion behavior. This section discusses GOFER's classifier-system architecture and its operating principles in more detail.

5.1 Functional constraints

The simulated environment places several demands on the organism that influence the way we have implemented the classifier system. First, GOFER must be capable of identifying and selectively processing the most relevant signals from the environment. In order to learn which stimulus patterns and associations are significant, it must cope with a bootstrapping problem. Clearly, it will not do to wait for external reinforcement to indicate relevance. Not only is such reinforcement rare in an uncertain environment, but regularly attaining reinforcement is an achievement that itself requires a considerable amount of learning. Similarly, it is not helpful to rely exclusively on explicit goals or

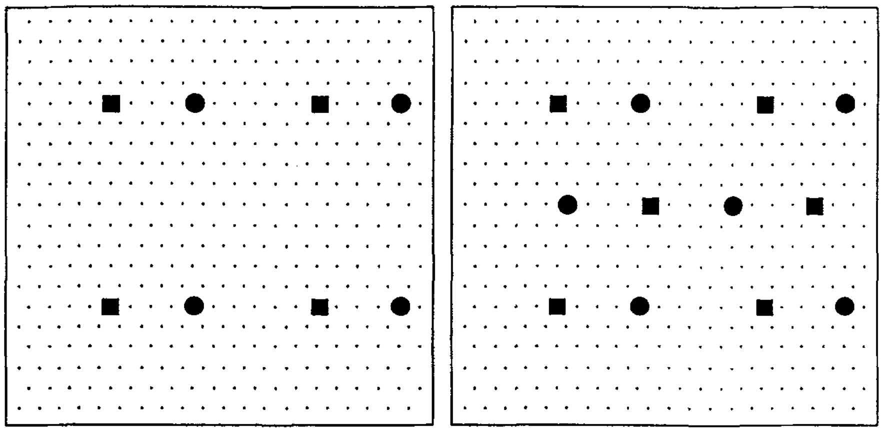

Figure 4. Two environments for experimenting with the locomotion task. The food objects (squares) and noxious objects (circles) all have intensity auras like those shown in Figure 1. The locomotion task is easy to solve in the environment on the right because EXPLORE is an adequate strategy for finding food. Surviving in the environment on the left is more challenging because it requires goal-directed processing of stimulus signals.

purposes to indicate relevance. Goal-directed selection presupposes some prior knowledge about the content or distal source of the stimuli. Even a default goal like "curiosity" cannot help unless it specifies exactly what it is the system is supposed to be curious about.

The bottom line is the need for an assortment of attention mechanisms that ensures the system gets a broad brush overview of what it needs to know. Attention in natural systems is a very complex issue (Norman, 1976) and we will not attempt to discuss its many facets in detail. It suffices to say that, at the sensory interface, selecting signals having distinctive physical properties is a useful heuristic for focusing attention. The only physical property of signals in the simulated environment having functional significance is the signal intensity. The higher the relative intensity within a stimulus aura, the closer the organism is to a given object. In this sense, intensity is a rough comparative measure of the importance of input signals.

GOFER must also have some way to selectively influence its information-processing activity using criteria other than sensory attention. In the instinctive organism, selective control at all levels is provided by motivational factors that selectively prime instinctive centers. Once the relevant internal representations have been learned by the classifier system, the same kind of control must be exercised. This cannot happen unless the system has a way of selectively activating the concepts relevant to the current goal. The need for this is clear once we consider environments in which more than one object can be detected at the same time. If each object is relevant to a different goal, the system must make sure that the object at the focus of attention is the one relevant to the current goal. Otherwise, the internal representation of the situation may be incoherent in the sense that the activated object and goal concepts are incompatible. Such dilemmas can be avoided by having the current motivational state facilitate the activation of all object and goal representations relevant to that state. This will help to bias the competition among alternative representations toward a coherent outcome.[4]

5.2 Basic elements of GOFER

These considerations lead to a classifier system design that differs in two important ways from the standard architecture. The first difference involves making a clearer distinction between the message patterns that activate a classifier and the *tags* that identify a classifier to the rest of the system. Tags are usually defined as a set of bits incorporated into the condition part of a classifier to provide a simple addressing capability. To send a message directly to a classifier with condition 0011##...##, for example, it suffices to prefix the message with the binary string 0011. In this sense the tag is an address that can be used to directly couple two classifiers by having the tag of one classifier match certain bits in the messages generated by another. More generally, tags of this kind are used to restrict the set of messages to which a classifier attends. We will refer to these as *coupling tags*. The drawback with using coupling tags as identifiers in this manner is that there is no mechanism to select all classifiers having a similar tag regardless of which messages are available. This capability is desirable if tags are given a functional significance, encoding the contexts, topics, goals, or computations relevant to a classifier.

In order to let representations be retrieved by either messages (data-driven attention) or motivations (model-driven attention), we extend the definition of a classifier to include *control tags*:

$$t_1, t_2, \ldots, t_n; g_2, \ldots, g_k \Longrightarrow m.$$

Each control tag g_i is a fixed-length binary string. Whereas the input taxa function like the releasing stimuli in Tinbergen's model, control tags are like the control inputs. When a control tag is similar to the binary designation of a current goal, motivation, or problem-solving context, the classifier in question receives extra *support* in the competition to become active. Support is the term Holland et al. (1986) use to describe the degree to which aspects of the current situation indicate that a classifier is actually relevant. Computationally, the binary designations of goals are like special messages that are always checked against the control tags of relevant classifiers. For instance, in the organism model there are two possible goals: avoid noxious objects, which is active by default, and the seek-food goal, which is considered when the organism is hungry. Support from the current goal biases activity toward goal-relevant classifiers and processes, thereby helping to achieve a more coherent flow of activity in the system. Because control tags are binary strings,

[4]It is important that classifier systems are capable of activating representations in unanticipated, and even "incoherent," ways. Dynamically constructing new combinations of existing rules and rule clusters is a powerful way to handle novel situations.

another advantage can be realized if control tags have the same length as messages. The control tags from active classifiers in one part of the system can serve as messages for classifiers in another part of the system. This gives classifier systems the important ability to recognize and act on the basis of their own internal patterns of activity.

The second difference between the GOFER architecture and standard classifier systems is that all messages have an associated intensity. Message intensities make it possible to modulate the degree of support a classifier receives. The intensities of messages from the sensory interface establish a rough priority among sensory inputs regarding how much relative influence they will have in activating a representation. Active classifiers produce messages with an intensity proportional to the size of their bid and the intensity of the messages for which they bid. This is the kind of support one classifier can give to another, thus helping the system generate coherent sequences of coupled classifiers. Support derived from control tags also has an associated intensity, which is based on the importance of the goal and the proximity of the control tag to a binary string identifying the goal.

In order to keep track of the net effect of these influences, each classifier is given a new parameter called *excitation*. Excitation is a temporary measure of how competitive a classifier will be on a given execution cycle. An increment to excitation is computed every time a classifier is relevant to a message, based on the specificity of the input taxon to the message and the total degree of support received. The more messages on the message list for which a classifier is relevant, the more excitation it accumulates. Classifiers compete probabilistically for activation based on their total excitation. In this way, each classifier is treated like a tentative hypothesis about how to satisfy some subset of the prevailing information-processing constraints. The combined influence of all these various biases on competition among classifiers yields a representation that is well suited to the system's current situation and problem-solving state.

5.3 The GOFER architecture

Control tags, message intensity, and excitation are all important aspects of GOFER's ability to model goals, object categories, and their relationships. GOFER uses *respondent conditioning* (Keller, 1954) to learn those relationships. Contact with a food resource or noxious object, as a primary reinforcer, elicits a built-in internal response. This response is the activation of a motivational state that coarsely identifies the stimulus as good or bad, providing support to help activate all relevant representations. Under the principle of respondent conditioning, pairing a neutral stimulus with the eliciting stimulus eventually causes the previously neutral stimulus to elicit the same response. In this case, the binary input message strings transmitted by an object are neutral *a priori*. GOFER must be conditioned so that the input messages detected at the moment of contact elicit the same coarse identification, or affect code, as actual contact. More important, once the system has learned to recognize categories of stimulus patterns, the same ability can be generalized to all signals characterizing an object. This makes it possible to learn the correct releasing stimuli for the food-seeking and pain-aversion instinctive centers.

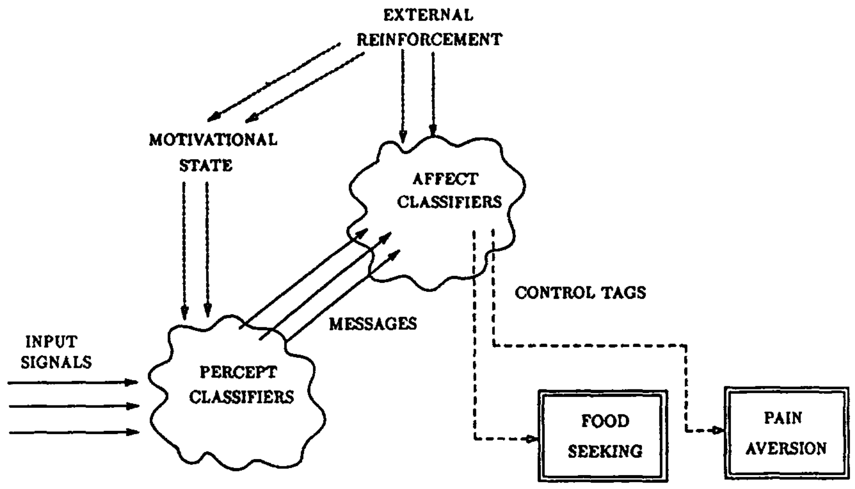

Figure 5. A classifier system that uses two populations of classifiers. The percept population represents object categories and the affect population represents reinforcing events. Affect classifiers are activated by messages from the percept population. The control tags of active affect classifiers serve as releasing signals for food-seeking and pain-aversion behaviors.

The classifier system designed to accomplish this is shown in Figure 5. There are two populations of classifiers, with the system processing messages through each population before producing an overt response. All classifiers have a single input taxon and a single control tag. As a simplification, each population has a separate message list to avoid having to learn coupling tags that route messages in the desired manner. The first population stores perceptual representations of the stimulus patterns encountered in the environment. These *percept* classifiers receive messages directly from the sensory interface and their control tags accrue support from the current motivational state. Percept classifiers thus represent object categories and their potential relevance to goal attainment. The population of *affect* classifiers receives the messages transmitted by active percept classifiers. Control tags for affect classifiers only get support from the motivational states induced by overt external reinforcement. For this reason, affect classifiers are used as internal representations of events involving reinforcement. Coupling a percept classifier with an affect classifier is the way GOFER models the prediction that detecting an object will ultimately lead to reinforcement. Respondent conditioning is achieved when the learning mechanisms discover useful percept and affect classifiers and couple them correctly.

The classifier system activity is tied to the motor control hierarchy using the control tags of active affect classifiers. The coupled percept/affect pairs take the place of the releasers for the food-seeking and pain-aversion instinctive centers shown in Figure 3. Every food-related control tag in the affect population is interpreted as a food signal. Pain-related tags are likewise interpreted as noxious signals. The two instinctive centers compete to be released based on the total excitation accrued by their respective affect classifiers. Holland et al. (1986) point out that a useful empirical model of the environment will include predictive relationships as well as atemporal associations. GOFER uses internal messages to implement predictive links and control tags to implement atemporal links.

5.4 Operating principles of the system

The overall execution cycle for GOFER proceeds as follows:

(1) Place messages from the sensory interface on the current input message list;
(2) Compare all messages to all conditions in the percept population and determine which classifiers will become active;
(3) For each active percept classifier, generate one message for the internal message list;
(4) Compare all internal messages to all conditions in the affect population and determine which classifiers will become active;
(5) Process the control tags of active affect classifiers to produce the system output;
(6) Empty both message lists, reset all excitation levels, and return to step 1.

As noted previously, the competition determining which classifiers are activated on a cycle is based on the amount of excitation each one accumulates.

In GOFER's problem-solving environment only the bare minimum of external reinforcement is provided: a binary indication of good or bad. The utility of a classifier is therefore defined in terms of internal, model-based criteria - its effectiveness in the model for categorizing and predicting environmental states - rather than in terms of expected overt rewards. In order to adjust strength to reflect these criteria, three factors must be taken into account: specificity, support, and *message impact.* Specificity and support have been mentioned previously in the discussion of excitation levels. Message impact refers to the amount of influence a classifier's message has in activating other classifiers coupled to it. This impact can be assessed by looking at the average specificity and support of classifiers activated in response to a message. The product of these three factors is used to compute a classifier's internal payoff, and strength is adjusted as a recency-weighted average of previous payoffs. The link between strength and goal attainment is achieved by treating external reinforcement as a very strong source of support.

It is worth emphasizing again that a classifier's relevance to goal attainment is indicated by its role in the internal model, not merely by its level of strength. This is an important point that distinguishes the strength adjustments in GOFER from the typical bucket-brigade credit assignment scheme. Learning mechanisms are used to discover high-strength rules that will improve the effectiveness of the internal model. Improvements in the model lead to more frequent goal attainment because the system comes equipped with a problem solver that can use the model effectively.

Here again, Samuel's (1959) checker player provides an instructive example. The min-max problem-solving strategy gave this program the rudimentary capability to play the game of checkers. The quality of play was determined by the amount of knowledge the program could infer about its environment, which in this case consisted of knowledge about how the opponent evaluated board positions. Performance improvements could therefore be made simply by learning more about the environment, without explicitly assigning credit for winning or losing any particular game.

In an analogous way, the motor control hierarchy gives our organism the basic capability to perform its locomotion task. The level of performance depends on how accurately the releasers characterize goal-relevant situations. Accordingly, the more the system learns about the situations it will encounter in the environment, the more effective it will be at the locomotion task. Ethologists (e.g., Gould, 1982) hypothesize that releasers are the primary mechanism through which instincts direct organisms regarding what to learn and how to use the knowledge they acquire. This is also somewhat reminiscent of the way SOAR (Laird et al., 1986) uses weak methods to provide a rudimentary ability to solve problems that is enhanced by domain-specific knowledge. The difference is that GOFER generates new rules by recombining building blocks instead of compiling (chunking) existing rules.

5.5 Implementation decisions

We noted earlier that implementing a classifier system involves making choices about how to manage resources, handle unrecognized events, and trigger inductive mechanisms. Here we discuss the choices that were made for GOFER.

Resource management. Since the number of concepts needed to model the problem-solving environment is not known by the system in advance, a mechanism is needed to dynamically manage the limited classifier memory. GOFER uses a mechanism called *payoff sharing.* All classifiers relevant to a given situation are treated as a group that is collectively responsible for how well the situation is modeled. Strength in a group is adjusted in a manner similar to the bucket-brigade computation: every classifier pays out a fixed fraction of strength as a fee for being included in the group; then an internally generated reward is distributed among classifiers in the group based on the three effectiveness factors cited above. The larger the group, the less reward each individual classifier gets. For groups of a given size, the most effective classifiers get more reward than the others. A crowding pressure on groups is exerted by choosing existing classifiers to be replaced by new ones in inverse proportion to their strength. In this way, the number of classifiers in a group dynamically adjusts itself based on the relative strength of other groups. Note that payoff sharing schemes are also effective when rewards are based more directly on external reinforcement (Wilson, 1987). GOFER uses a genetic algorithm modified with a *restricted mating policy* to learn multiple concepts in a single population of classifiers. In standard implementations of genetic algorithms, the search for new classifiers is unconstrained in the sense that any two classifiers have a non-zero probability of being paired under the crossover operator. This means that classifiers representing different categories can be combined to produce new classifiers not likely to be useful for categorization. For example, consider the pattern characteristics 111111*** and 000000***, designating two distinct object categories. The crossover operator will combine classifiers matching these categories to produce new classifiers matching characteristics like 111000***, which do not correspond to either object category. This difficulty can be avoided by restricting the genetic algorithm to recombine only those classifiers competing for the same message.

Unrecognized situations. Organisms in the real world may not get a second chance to do the right thing in an unfamiliar situation. Therefore, it is important that GOFER be designed to manage novel situations in a responsible way. In particular, it seems ill-advised to make arbitrary assumptions about goal relevance and appropriate behaviors when constructing rules to handle these situations. One way to avoid ad hoc constructions is to rely heavily on relevant past experience. If no classifiers match a message, the best the system can do is to find existing classifiers that are partially relevant and hope that what worked in the past will be at least somewhat appropriate in the current case. Using partially relevant rules also gives the inductive mechanisms a head start in identifying useful building blocks for constructing a new classifier that handles the situation adequately.

Accordingly, GOFER uses a *partial match* criterion to determine relevance instead of treating the match between a classifier and a message as an all-or-none event. Very briefly, the usual match criterion requires that the features in the message correspond exactly with the 0's and 1's encoded in the classifier condition. Specificity is given by a match score that simply counts the number of 0's and 1's in a matched condition. GOFER computes a partial match by counting the number of matched 0's and 1's and gives partial credit for each #, since they match by default. In order to avoid having too many misleading rules deemed relevant because of a partial match, the execution cycle of the classifier system is augmented with a preliminary competition. This competition, based on specificity and strength, determines which of the partially matched classifiers will be allowed to accumulate excitation and participate in the main competition to become active. Note that partial matching and the preliminary competition are only used in unrecognized situations. Booker (1985) provides more details about partial matching, restricted mating, and payoff sharing.

Triggering conditions. The clearest opportunity to trigger rule discovery is during a reinforcement event. External reinforcement leads to high-intensity support for classifiers having the correct control tag, so that support and message impact become dominant factors in strength adjustment and the competition to become active. This support provides direct information about the goal-relevance of all active concepts. The genetic algorithm is always triggered during these events, in order to take advantage of this unambiguous feedback. Modification of classifiers also occurs at a slow background rate independent of any external reinforcement. This is a default learning strategy that assures representations for the most frequently encountered situations will eventually be constructed. All other factors being equal, the primary evaluative criterion during these background learning episodes is the specificity of match with the current set of messages.

6. Experiments with GOFER

We are now ready to determine if GOFER is capable of learning how to behave in the simulated environment. Recall that the learning task confronting an organism is twofold. First, it must discover the category structure of the environment by deciding which inputs to group into categories and it must make pragmatic generalizations (Lebowitz, 1987) about how to define each category. Second, the system must learn how each category is significant with respect to its goals of finding food and avoiding noxious stimuli.

In order to demonstrate that GOFER behaves as expected, several organisms were tested in the simple environment shown in Figure 4. This environment contains twelve uniformly distributed objects, six of each type, so that at least one stimulus signal can be detected at nearly every location. Food resources emit signals having the characteristic 1111111111******. Noxious objects emit signals having the characteristic 0000000000******. In order to prevent organisms from dying while they learn, we assume that an organism has its need for food provided during a training interval of fixed length. During this time, the only responses an organism can make are to CONSUME a needed food resource when in contact with it, ESCAPE from contact with a noxious object, or EXPLORE. Each organism has 200 percept classifiers and 200 affect classifiers generated initially at random. The system innately interprets the control tag 000...000 as the code for the food-seeking goal and 111...111 as the code for pain aversion. Selection of one of these goals is determined by a winner-take-all competition among the active affect classifiers. However, no overt food-seeking or pain-aversion behavior is released.

Five organisms were each simulated for 10,000 execution cycles or trials. The organisms were evaluated according to how often they recommended behavior that, if allowed to become overt, would be inappropriate given the system's motivational state and the current input configuration. An error was recorded each time this occurred. Two sources of these errors were observed in the training simulations. Either the releasing signal generated was incorrect due to an errant coupling between percept and affect (a "failure of association") or, when faced with simultaneous stimulation from both kinds of objects, the concept that became active was not the one most relevant to the current motivational state (a "failure of attention"). Figure 6 shows the average cumulative error for five organisms simulated over 10,000 execution cycles or encounters with external stimuli. Although the average behavior of the organisms quickly improves above that expected by chance, the organisms continue to make errors at a relatively steady rate.

An examination of the classifiers learned by the errant organisms revealed a common symptom. The affect classifiers all had similar control tags and their input taxa matched the messages transmitted by both kinds of percept classifiers. This explains why the error rate failed to decrease. After a certain point, the internal model generated the same releasing signal in all circum-

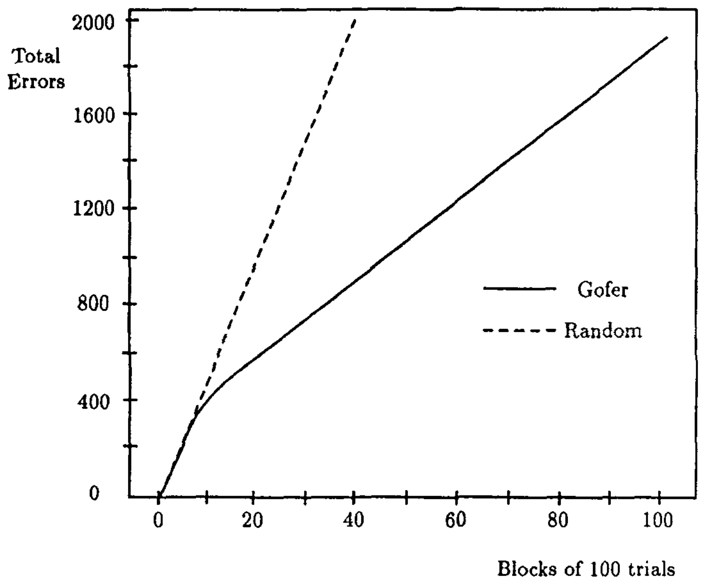

Figure 6. Average cumulative error for initial training simulations of five organisms.

stances. Interestingly, that signal was for pain aversion, which is most often the correct action given that food resources are automatically replenished. All of the affect classifiers with food-seeking control tags had been deleted.

Why did this happen? Problems involving convergence of a genetic algorithm to an undesirable solution usually arise when the selective pressures have been incorrectly specified. The internal models learned by these organisms indicate a marked lack of pressure toward any meaningful coupling between percept and affect classifiers. New classifiers are generated by the genetic algorithm at a background rate of once every ten cycles to learn the category structure of the environment. The genetic algorithm is also triggered whenever the organism receives external reinforcement so that the goal-relevance of all classifiers can be learned. This occurs at an average rate of approximately once every 19 cycles. The genetic algorithm is therefore triggered most frequently in non-reinforcing situations. In the simulated environment, an organism is most often at a locus where signals from both kinds of objects are available simultaneously. Applying the genetic algorithm to the affect population in that situation generates a selective pressure for affect classifiers that match messages from both kinds of percept classifiers. This pressure is exacerbated when these situations occur more frequently than the unambiguous situations involving reinforcement. Once the affect population fails to discriminate between the two kinds of messages, the system has no way to maintain two different control tags. The most frequently correct code for pain aversion eventually predominates over the alternative.

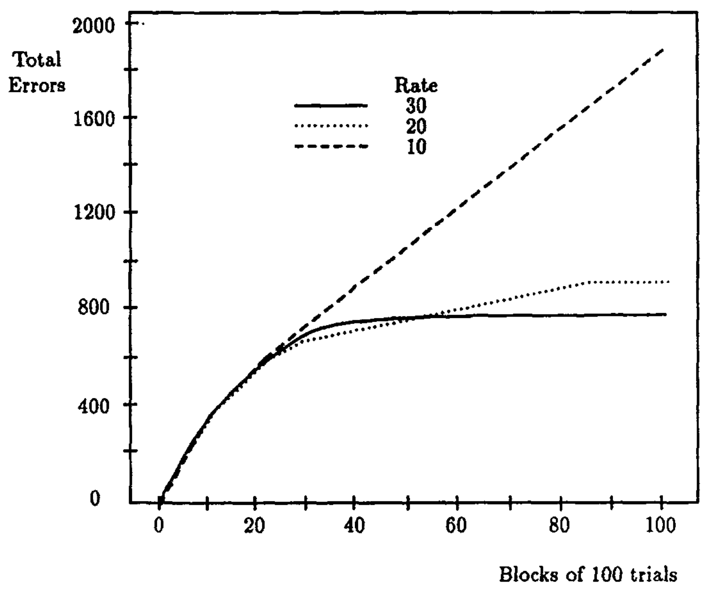

Figure 7. The effect of different background rates for the genetic algorithm on overall performance.

Based on this analysis, decreasing the background rate at which the genetic algorithm is triggered should solve the problem. This would make external reinforcement the determining factor in learning control tags, as originally intended. Two alternative rates were examined: once every 20 cycles, which is roughly equal to the external reinforcement rate, and once every 30 cycles, which makes external reinforcement the dominant factor. Each increase resulted in a marked improvement in performance, as shown in Figure 7. The rate of once every 30 cycles is clearly superior, giving a quick and steady reduction in errors. Over the last 1000 trials of the interval of observation, the organisms had an average error rate of one every 25 cycles. Two of the organisms made no errors at all during this period.

The few errors being made at this point were all failures of attention, related to the way competition works in the system. One of the organisms chose to represent food signals disjunctively using the two taxa 1#11##111#####1# and 1#11##111#####0# instead of the more compact 111111111######. This is a perfectly reasonable solution to the categorization problem because it always gives the correct answer in the simulated environment. However, when more than one signal is available from an object, the total excitation is distributed over two groups of classifiers instead of one. This dissipation of activity sometimes makes each group less excited than the competing concept, even though their combined excitation is much greater. The other more frequent source of error was that the stochastic competition does not always result in the most excited classifiers becoming active. When the levels of excitation for competing classifiers are relatively close, the less excited alternatives win nearly as often as the most excited one.

Neither of these two sources of error seem significant from the standpoint of the GOFER's overall functioning. Indeed, stochastic competition is the way classifier systems test new hypotheses that ultimately lead to better performance. The best demonstration that this flexibility does not interfere with the performance of the task is to show that a trained organism is capable of surviving on its own. Accordingly, the organism that made the *most* errors in the training simulation was tested with its food-seeking and pain-aversion behaviors under the control of the affect classifiers. The environment used for this test was the difficult one shown in Figure 4, which the instinctive organism handled successfully. This test is a worst-case estimate of how much the so-called errors interfere with accomplishing the task. Unless the organism makes the correct decisions about when to APPROACH and AVOID, its hunger level will exceed threshold and it will die.

The organism was simulated for over 700,000 cycles to be sure it had ample opportunity to make a tragic set of errors. The average hunger level over this interval was almost identical to that observed for the instinctive organism. Errors tended to occur in clusters separated by long periods of perfect performance. The organism's behavior indicated that, when errors occurred, the classifiers related to pain aversion were gradually being deleted from the population. However, a brief period of encounters with noxious stimuli promptly restored them to their deserved proportions. Quite simply, as noxious signals are successfully avoided, there is no opportunity for the genetic algorithm to make new copies of these classifiers. The continued reproduction of classifiers related to food seeking increases their proportion in the population at the expense of the pain-aversion classifiers. This "forgetting" problem can be alleviated by protecting high-strength classifiers from deletion.

The simulations done so far indicate that the organism can learn when to release behavior that was completely preprogrammed in the instinctive organism, but the system is capable of much more interesting learning. If knowledge about the environment changes in any significant way, an individual instinctive organism has no way of changing its behavior so as to remain viable. GOFER generates a useful and flexible internal model as a result of its experience with the environment. This internal structure, and the inherent learning capabilities of the system, give the organism a meaningful advantage whenever knowledge must be revised to account for new experiences. One way to demonstrate this advantage is by the transfer of learning from one situation to another.

To demonstrate the beneficial effects of prior structure on learning in new situations, we revised the environment. Objects were distributed as before, except that now food resources generated signals with the different characteristic `*****11111******` and noxious objects generated signals with the characteristic `*****00000******`. This environment is a generalization of the one used in the previous simulations, because this set of signals includes the original set of signals as a proper subset. An organism trained in the original environment has many classifiers that are still useful in the new environment, but addi-

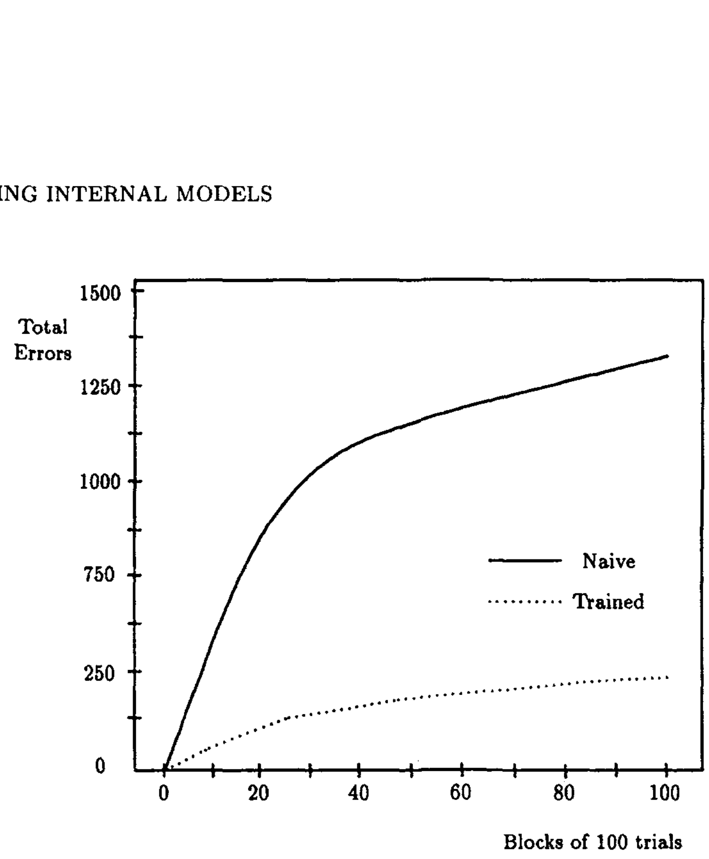

Figure 8. Positive transfer of previous experience to a new environment.

tional classifiers must be learned. Sets of signals such as 1111100000****** are totally ambiguous to a trained organism. Fortunately, the population of classifiers in a trained organism contains many building blocks well suited for constructing new classifiers to resolve the added uncertainty. The results of a simulation testing for positive transfer of experience confirm the assertion that trained organisms can learn the new environment more readily than a naive organism. Figure 8 shows that the organisms trained in the original environment make fewer total errors and learn much faster than their naive counterparts.

7. Conclusions

Because classifier systems are rule-based systems, learning in this framework is most often viewed as the acquisition of procedural knowledge. This characterization is compatible with stimulus-response theories of learning that emphasize the way in which external reinforcement guides the development of useful response sequences. Indeed, most classifier systems learn a collection of S-R rules, each of which directly acts on the environment and accrues strength proportional to the overt reward expected from the behavioral sequences in which the rule participates (Holland & Reitman, 1978; Goldberg, 1983; Wilson, 1985). Cognitive theories of learning, on the other hand, assert that learning involves the acquisition of an internal model containing knowledge about the environment and how to function in it. The emphasis in our research has been on explicit internal representations of events, expectations, and goals, as well as how these relate to each other. Representations are learned under the guidance of general principles of association, such as contiguity, instead of solely on the basis of reinforcement. GOFER demonstrates how classifier systems can be used to study these issues in cognitive learning.

The system learns to produce goal-seeking behavioral sequences without learning any stimulus-response rules. Instead, what GOFER learns are rules that represent objects, goals, and associations between them. Sequences of behavior are generated by an organized collection of action routines designed to effectively use the system's knowledge about its environment. The design effort in this approach to machine learning research is therefore centered more on a knowledge-based decomposition of the task than on finding an informative reward function or learning critic. Because goal relevance can be determined from associative relationships in GOFER's internal model of the environment, there is no need to rely exclusively on a single parameter like strength to summarize a classifier's utility. This greatly reduces the burden on credit allocation schemes to generate numbers that meaningfully evaluate rules involved in long entangled sequences of behavior. Goal relevance can be *retrieved* from associations in the model instead of being computed as a number and stored with each classifier.

GOFER introduces several new ideas about how to design and use classifier systems. It is the first classifier system to learn to use internal messages and message intensities to solve a problem. Internal messages are the cornerstone of any effort to view classifier systems as a general framework for studying learning and problem solving (Holland et al., 1986). GOFER also demonstrates the importance of payoff sharing and restricted mating as mechanisms for dynamically and automatically allocating the space in a single population of classifiers to learn several concepts in parallel. Finally, the system shows how partial matching can be used effectively to learn in unrecognized situations, and how control tags can help direct problem-solving activity in classifier systems.

The simple model of instinctive behavior we have used is sophisticated enough to control behaviors more elaborate than idealized locomotion. The principle of hierarchically controlled behaviors, with one level elaborating and subsuming another, has already been used to flexibly and robustly control mobile robots (Brooks, 1986). The ease with which this architecture can be extended to accommodate new behaviors without disturbing existing capabilities is an important advantage. An obvious role for learning in such systems is to refine the releasing criteria for response pathways so that actions are triggered only in the right circumstances. The control exercised by motivational factors can be generalized to include factors not tied so closely to external reinforcement, making GOFER more relevant to other kinds of problem-solving behavior. Several kinds of criteria that do not depend on external reinforcement can be used to control behavior and trigger learning. Examples include the perceived *novelty* of inputs, the outcome of *predictions*, "...well-ordered *preferences*, sound *plans of action*, in short all the favorite tools of the cognitive psychologist" (Dennett, 1978, p. 80). Another way GOFER can be extended is to view the instinctive centers as primitive problem-solving agents. The searching behavior associated with an instinctive center is a simple version of a *difference engine* (Minsky, 1986). These agents act to reduce the difference between the current situation and some desired situation. Reducing differences is a powerful way of characterizing the essential aspects of purposeful behavior (Ernst & Newell, 1969).

An important question about GOFER is how well its representation and problem-solving capabilities scale up to more complex problems. The model-building achievements of the system are relatively modest, involving only learned associations between objects and the goals to which they are related. A more general model of the environment would also include a broad spectrum of relationships among the objects themselves. It must still be demonstrated that GOFER's capabilities let it use internal messages to model more than one kind of association. In particular, it remains to be shown that nontrivial *default hierarchies* (Holland et al., 1986), based on subordinate-superordinate relations among categories, can arise and persist through the mechanisms currently in use. The appealing explanatory power of classifier systems is still far ahead of our knowledge about how to build them.

Acknowledgements

The experiments described in this paper were done as part of the author's Ph.D. dissertation research at the University of Michigan. The work was supported by the Ford Foundation, the IBM Corporation, the Rackham School of Graduate Studies, and Grant MCS78-26016 from the National Science Foundation.

I would like to thank David Goldberg and Pat Langley for their helpful comments and thorough work as editors. Special thanks go to Stewart Wilson and Stephen Kaplan for their insights and detailed comments on an earlier draft of this paper.

References

Bindra, D. (1978). How adaptive behavior is produced: A perceptual-motivational alternative to response-reinforcement. *The Behavioral and Brain Sciences*, *1*, 41–91.

Boden, M. (1977). *Artificial intelligence and natural man.* New York: Basic Books.

Booker, L. B. (1982). *Intelligent behavior as an adaptation to the task environment.* Doctoral dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor.

Booker, L. B. (1985). Improving the performance of genetic algorithms in classifier systems. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 80–92). Pittsburgh, PA: Lawrence Erlbaum.

Booker, L. B., Goldberg, D. E., & Holland, J. H. (in press). Classifier systems and genetic algorithms. *Artificial Intelligence.*

Bower, G. H., & Hilgard, E. R. (1981). *Theories of learning.* Englewood Cliffs, NJ: Prentice-Hall.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, 1,* 14–23.

Bruner, J. S. (1957). Going beyond the information given. In J. S. Bruner (Ed.), *Contemporary approaches to cognition.* Boston: Harvard University Press.

Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Craik, K. J. W. (1943). *The nature of explanation.* New York: Cambridge University Press.

Dennett, D. (1978). *Brainstorms.* Cambridge, MA: MIT Press.

Doran, J. E. (1968). Experiments with a pleasure-seeking automaton. In D. Michie (Ed.), *Machine intelligence* (Vol. 3). New York: American Elsevier.

Dreyfus, H. L. (1972). *What computers can't do: A critique of artificial reason.* New York: Harper & Row.

Ernst, G., & Newell, A. (1969). *GPS: A case study in generality and problem solving.* New York: Academic Press.

Findler, N., & Allan, A. (1973). Studies on the behavior of an organism in a hostile environment. *Journal of the Institution of Computer Science, 4,* 58–69.

Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and rule learning.* Doctoral dissertation, Department of Civil Engineering, University of Michigan, Ann Arbor.

Gould, J. L. (1982). *Ethology: The mechanisms and evolution of behavior.* New York: W. W. Norton.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, 16,* 122–128.

Hayes-Roth, F. (1973). A structural approach to pattern learning and the acquisition of classificatory power. *Proceedings of the First International Joint Conference on Pattern Recognition* (pp. 343–355). Washington, DC.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Holland, J. H. (1976). Adaptation. In R. Rosen and F. M. Snell (Eds.), *Progress in theoretical biology* (Vol. 4). New York: Academic Press.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 1–7). Pittsburgh, PA: Lawrence Erlbaum.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.) *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: MIT Press.

Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* New York: Academic Press.

Kaplan, S. (1982). Perception of an uncertain environment. In S. Kaplan & R. Kaplan (Eds.), *Humanscape: Environments for people.* Ann Arbor, MI: Ulrich's Books.

Keller, F. S. (1954). *Learning: Reinforcement theory.* New York: Random House.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, *1*, 11–46.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, *2*, 103–138.

Lenat, D., Prakash, M., & Sheperd, M. (1986). CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, *6*, 65–85.

Michalski, R. S. (1986). Understanding the nature of learning: Issues and research directions. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.) *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Minsky, M. (1986). *The society of mind.* New York: Simon and Schuster.

Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, *92*, 289–316.

Norman, D. A. (1976). *Memory and attention* (2nd ed.). New York: Wiley.

Norman, D. A. (1981). Twelve issues for cognitive science. In D. A. Norman (Ed.), *Perspectives on cognitive science.* Norwood, NJ: Ablex.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, *3*, 210–229.

Scott, P. D. (1983). Learning: The construction of a posteriori knowledge structures. *Proceedings of the National Conference on Artificial Intelligence* (pp. 359–363). Washington, DC: Morgan Kaufmann.

Simon, H. A. (1983). Search and reasoning in problem solving. *Artificial Intelligence*, *21*, 7–29.

Sutton, R. S., & Pinette, B. (1985). The learning of world models by connectionist networks. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 54–64). Irvine, CA: Lawrence Erlbaum.

Tinbergen, T. (1951). *The study of instinct.* New York: Oxford University Press.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 16–23). Pittsburgh, PA: Lawrence Erlbaum.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning*, *2*, 199–228.