Artificial Life and the Animat Approach to Artificial Intelligence

Jean-Arcady Meyer

What is Artificial Life?

Artificial Life (AL) is a novel scientific pursuit which aims at studying man-made systems exhibiting behaviors characteristic of natural living systems. "AL complements the traditional biological sciences concerned with the analysis of living organisms by attempting to synthesize life-like behaviors within computers or other artificial media. By extending the empirical foundation upon which biology is based beyond the carbon chain life that has evolved on Earth, AL can contribute to theoretical biology by locating life-as-we-know-it within the larger picture of life-as-it-could-be " (Langton, 1989) . In other words, AL views life as a property of the organization of matter, rather than a property of the matter which is so organized. In still other words, whereas biology has largely concerned itself with the material basis of life, AL is concerned with the formal basis of life.

At the core of the AL research program is the concept of emergent properties . These are properties exhibited by a collection of interacting organisms whose global behavior cannot be reduced to a simple aggregate of the individual contributions of these organisms. In other words, emergent behavior is said to exist when the whole is greater than the sum of its parts. Life, for example, is an emergent property resulting from interactions at a certain level of integration which cannot be explained simply in terms of the sum of the properties of elements interacting at lower levels (organs, tissues, cells, membranes, molecules, and so forth). Because traditional methods in biology are typically reductionist, and seek most often to decompose a system into its constituent sub-systems and then study these subsystems in isolation from one another according to a top-down approach, it is difficult for these methods to reveal and explain the emergent properties characteristic of living organisms. On the other hand, AL is resolutely bottom-up in its approach. Starting with a collection of organisms exhibiting behaviors which are simple and well understood, AL aims to synthesize more complex systems in which organisms interact in ways which are non-linear and give rise to lifelike emergent properties. Such a methodology reveals what sort of elementary properties and what sort of interactions are necessary and sufficient for the appearance of particular emergent properties. It also permits a very high degree of control and reproducibility, and may therefore prove itself to be a useful complement to traditional biological approaches (Taylor, 1992).

Research in AL has recently given rise to a number of applications (Langton, 1989; Langton et al., 1992; Levy, 1992), notably to computer viruses, to biomorphs and ontogenetically realistic processes, to autocatalytic nets, to cellular automata, and to artificial nucleotides.

One particularly active area of Artificial Life is concerned with the conception and construction of animats (Wilson, 1985) -- that is, of animals simulated by computers or by actual robots -- whose rules of behavior are inspired by those of animals (Meyer and Wilson, 1991; Meyer et al., 1993). This research has a twofold objective. In the short term, it aims to understand and reproduce the mechanisms which enable animals to adapt and survive in changing environments. In the longer term, it aims to contribute to our understanding of human intelligence, an objective reinforcing that of AI Artificial Intelligence.

After highlighting the similarities and differences between the animat approach and standard approaches in AI, this paper describes a number of related efforts in the two fields.

The animat approach to AI

- Objective and methods of standard AI approaches

The majority of research in AI considers the simulation of the most elaborate faculties of the human brain -- the resolution of problems, the understanding of natural language, and the ability to reason logically, for example. These simulations take the form of computer programs implementing computational approaches to human cognition. They generally involve the use of "physical symbol tokens", in other words, various objects or physical patterns (like, for example, the set of magnetic moments of the various particles constituting the memory of a computer) which represent something other than themselves (for example, a set of objects in an environment, concepts, desires, emotions, and so forth), though syntactic rules operating on the physical characteristics of these symbols. The motto of AI in its standard form is that such a "symbolic physical system" is necessary and sufficient to produce intelligent behavior (Newell and Simon, 1976).

It is characteristic of simulations in standard approaches to AI to use data carefully selected by the programmer and to consider restricted application domains whose heterogeneity and predictability are controlled as rigorously as possible.

- Objectives and methods of the animat approach

With the aim of explaining the essence of faculties peculiar to humans by examining the simplest hereditary adaptive systems of animals, the animat approach is based on the conception or construction of simulated animals or robots capable of "surviving" in unpredictable environments presenting a greater or lesser degree of danger. Like certain animals (Roitblat, 1987), these animats prove themselves capable of actively searching for essential information and of choosing behaviors which permit them to react beneficially with their environment. Moreover, they are often able to improve their adaptive faculties thanks to learning performed by individuals or to evolutionary processes occurring across many successive generations. In these areas, the animat approach is inspired in part by the most recent work on the cognitive behavior of animals (Roitblat and Meyer, 1994), in part by computational tools with a basis in natural phenomena -- such as Neural Nets and Genetic Algorithms.

The resulting simulations do not require any symbolic manipulation to produce adaptive or intelligent behavior. They demonstrate that such manipulation is perhaps sufficient to produce such behavior but is not, in any case, necessary. On the contrary, the simulations implemented under the auspices of the animat approach seek, according to the aims of AL described above, to cause the properties of adaptation and intelligent behavior to appear in the form of emergent functionalities (Steels, 1991) issuing from interactions between simple behavioral mechanisms. Rather than immersing programs which exhibit a narrow competence of a high order in environments simplied to the extreme, the animat approach aims to model organisms which are simple, but complete, interacting in environments as realistic as possible, in which the organisms can nourish themselves, escape predators, etc. (Dennett 1978).

The motto here is that it is possible to touch on issues of human intelligence according to a bottom-up approach which, originating in minimal architectures and simple environments, aims progressively to increase the complexity of these architectures and environments. If we take care to add to these architectures only those features which are necessary to the primary goals of perception, categorization, and the pursuit of autonomously generated tasks, it will become possible to resolve increasingly complex problems of survival without loosing the capacity to resolve the simplest. In the long term, we might hope to come to understand by what evolutionary mechanisms the adaptive capabilities of bacteria gave birth to human intelligence and why it took so much more time to learn to survive and master the environment than to manipulate symbols. As Brooks put it (1990): "This suggests that problem solving behavior, language, expert knowledge and application, and reason, are all rather simple once the essence of being and reacting are available."

- How the two approaches complement one another

It therefore appears, as Wilson has stressed (1991), that "Standard AI is basically competence-oriented, modeling specific human abilities, often quite advanced ones. However, while many AI programs exhibit impressive performance, their relevance to the understanding of natural intelligence is, in several respects, limited. In addressing isolated competences, AI systems ignore the fact that real creatures are always situated in sensory environments and experience varying degrees of need satisfaction. Furthermore, the systems attach less importance to such basic natural abilities as perception, categorization, and adaptation than they do to algorithmic processes like search and exact reasoning. This leads to problems connecting the arbitrary symbols used in internal reasoning with external physical stimuli ("symbol grounding (Harnad, 1990)), and to "brittleness" (Holland, 1986), the tendency for AI systems to fail utterly in domains that differ even slightly from the domain for which they were programmed."

The animat approach, on the other hand, places importance on the characteristics neglected by standard AI approaches. It is interested explicitly in the interactions between an animat and its environment and particularly stresses the aptitude of the animat to survive in unexpected environmental circumstances. Centered around the study of behavior rooted in the real and the robust, research on the adaptive behavior of animats necessarily avoids the hazards of standard AI approaches and improves our knowledge in those domains where this last has failed notoriously, while addressing the resolution of problems of perception, of categorization, and of sensory-motor control (Brooks, 1991; Maes, 1992; Roitblat, 1994).

What is adaptive behavior ?

In a changing, unpredictable, and more or less threatening environment, the behavior of an animal is adaptive so long as the behavior allows the animal to survive. Under the same conditions, the behavior of a robot is considered to be adaptive so long as the robot can continue to perform the functions for which it was built. The survival of an animal is intimately involved with its physiological state and the successful operation of a robot depends upon its mechanical condition.

Under these circumstances, it is obvious that one can associate with an animat a certain number of state variables upon which its survival or successful operation depends, and that each of these state variables has a specific range within which the animat's continued survival or operation is preserved. Ashby referred to such variables long ago as essential variables . Their ranges describe a zone of viability inside the given state space, allowing the animat to be referenced at any instant by a point within this zone (Figure 1). Under the influence of environmental or behavioral variations affecting the animat, the corresponding reference point moves and may at times approach the limits of the viability zone. In this case, the animat's behavior can be called adaptive so long as it avoids transgressing the boundary of viability (Ashby, 1952; Sibly and McFarland, 1976).

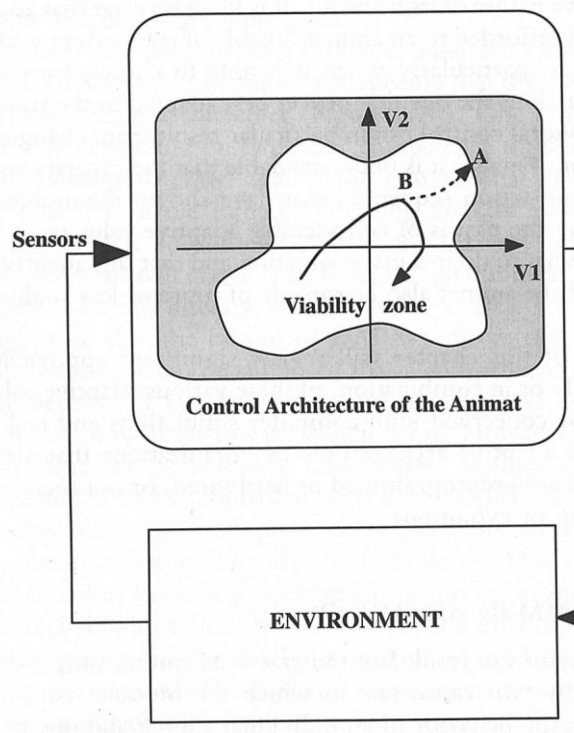

FIGURE 1 Viability zone associated with two essential variables, V1 and V2. The adaptive behavior of the animat is evidenced by the corrective action it takes at point B to avoid a movement outside of the viability zone at point A.

Such behavior can be generated by means of several different or complementary abilities and architectures. For example, the laws governing the animat's operation may rely upon various homeostatic mechanisms thanks to which, if the reference point alluded to earlier moves away from an adapted point of equilibrium -- adapted because it is suitably located within the viability zone -- this process tends to return it to its original position, thereby decreasing the risk that it will pass outside the limits of the zone. Other ways in which to lower this risk involve the use of high-quality sensory organs or motor apparatus that allows the animat to detect as early as possible that it is approaching these limits and/or to move away from them quickly and effectively. In this line of reasoning, it is obvious that the equivalent of a nervous system is mandatory in order to connect the animat's perceptions with its actions and that reflex circuits activated as quickly as possible increase the adaptive nature of its behavior. It is likewise clear that additional adaptive potential is afforded to an animat capable of responding with more than simple reflexes, particularly when it is able to choose from among several possible reactions the one that proves best adapted to the situation at hand. Such a behavioral control can in particular result from changes in the animat's motivation brought on by the situation. Lastly, it is understandable that the capacity to memorize the perception/action sequences that have shown themselves to be useful or harmful in the past is of considerable adaptive value to an animat obliged to decide what to do in a given situation, and that this adaptive value is enhanced should the animat also be capable of more or less sophisticated forms of planning.

The remainder of this article will review significant approaches that make use, separately or in combination, of these various adaptive solutions. This review will be concerned with computer simulations and real robots and will deal with innate -- because they are programmed or wired in -- and acquired -- because they result from learning of evolution -- adaptive solutions.

Preprogrammed behaviors

The fact that adaptive or intelligent behavior can result from interactions among simple modules can perhaps be illustrated by two cases: that in which the modules compete to control the behavior of an individual animal and that in which the modules constitute all of the elementary animats living and interacting in some societal context.

- Individual intelligence

Many animats exhibit adaptive behaviors because they were purposely programmed or cabled this way. For instance, work by Brooks is based on the construction of real robots whose sizes, morphologies, and missions vary (Flynn and Brooks, 1988), but which are all controlled by the same subsumption architecture (Brooks, 1986). Essentially, this architecture consists in superimposing layers of networks of finite-state machines, augmented with various timers and registers. Each layer connects sensors to actuators and implements a control system which achieves a certain level of competence. Higher-level layers can subsume the roles of lower levels by suppressing their outputs. However, lower levels continue to function as higher-level layers are added and higher levels of competence are achieved. The result is a robust and flexible robot-control system needing much less computation than in more traditional approaches. For example, this architecture allows the robot Genghis to chase infrared sources over rough terrain (Brooks, 1989). Likewise, it permits Squirt -"the world's largest one-cubic-inch robot" -- to act as a "bug", hiding in dark corners and venturing out in the direction of noises only after the noises are long gone. Connell (1990) demonstrates how it is possible to account for the behavior of the coastal snail Littorina by supposing that this behavior is controlled by a subsumption architecture (Figure 2). One can indeed regard this behavior as depending on two basic competence modules: UP, which tells the snail always to crawl against the gravitational gradient, and DARK, which tells it to avoid light by crawling directly away from the source. However, DARK subsumes UP, i.e. if a very bright light source is present, the snail will crawl away from it even if it means going downward. On Figure 2B, this interaction is shown by a circle with an arrow entering it, suggesting that the output of DARK replaces the output of UP. In a similar manner, it can be supposed that a competence BRIGHT subsumes DARK because, if one turns the snail upside down, instead of avoiding light, it will now head toward bright areas. However, because this light-seeking behavior occurs only underwater, another competence module must be added to the control architecture: DARKER, which takes precedence over all the other light-sensitive behaviors when the snail is out of water. Finally, a last module, STOP, halts the snail when it encounters a dry surface and thus keeps it from wandering too far inland. Fraenkel (1980) explains how this collection of competence modules and their interactions aids the snail in its pursuit of food and how it allows it to arrive at the region of maximum algae concentration, even if it has to negotiate major obstacles along the way (Figure 2A). Other arguments supporting the biological realism of the subsumption architecture -- or, more precisely, the logic of decentralized control it implies -- are to be found in Altman and Kien (1989). Indeed, this logic mobilizes no central control module responsible for analyzing the environmental situation at each instant, then for deciding to activate one specific behavior module or another. On the contrary, each competence module in Brooks' robots reacts in parallel to the environmental characteristics concerning it, and the collective behavior is an emergent property resulting from the interactions between instantaneous behaviors of the modules.

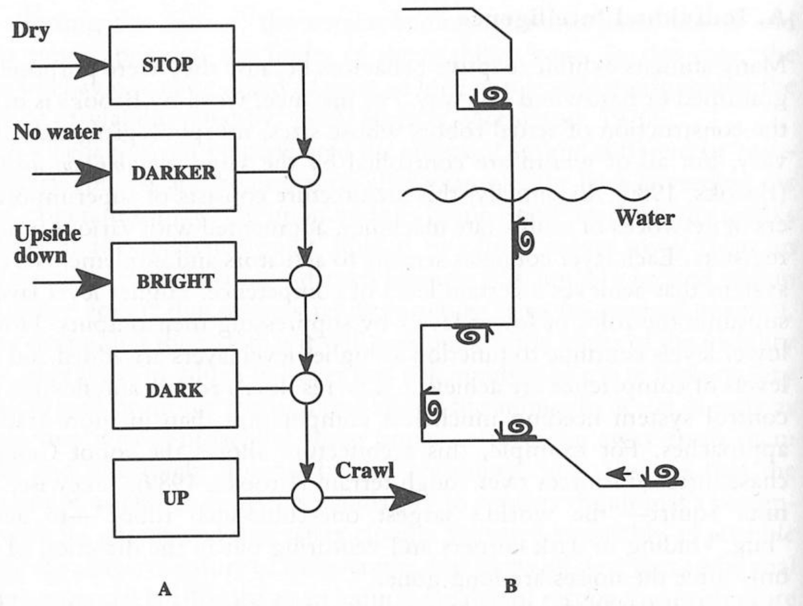

FIGURE 2 (A) The subsumption architecture underlying Littorina's behavior. (B) Littorina's adaptive crawling behavior. (Adapted from Connell, 1990, Figure 1.3, p. 8.)

- Swarm intelligence

A simple generalization of the previous remark applies to another biological metaphor underlying much work on animats, that of insect society or of swarm intelligence. Here, the idea is to use a colony of animats with individual behaviors that may be quite simple but, due to the fact that they communicate with each other -- either directly, or by means of the environment -- can exhibit a collective behavior that is relatively complicated and adaptive. In particular, this collective behavior can be maintained even in the event of the dysfunction of one or more individual animats. Brooks et al. (1990) describe, for example, simple individual behaviors that must be able to be exhibited by robots capable of forming a colony that could be sent to the moon for the purpose of building a permanent base. Thus, the interactions and individual behaviors on Table 1A would enable the colony to select an open surface, whereas the interactions and individual behaviors in Table 1B would take care of the leveling of the ground and ensuring that the soil was deposited in a few piles at the borders of the construction site. Similar individual behaviors can be designed to achieve other global tasks cooperatively, such as pushing a rock or a broken robot out of the way, digging out trenches for the habitation units, covering the habitation units with soil from the piles, etc.

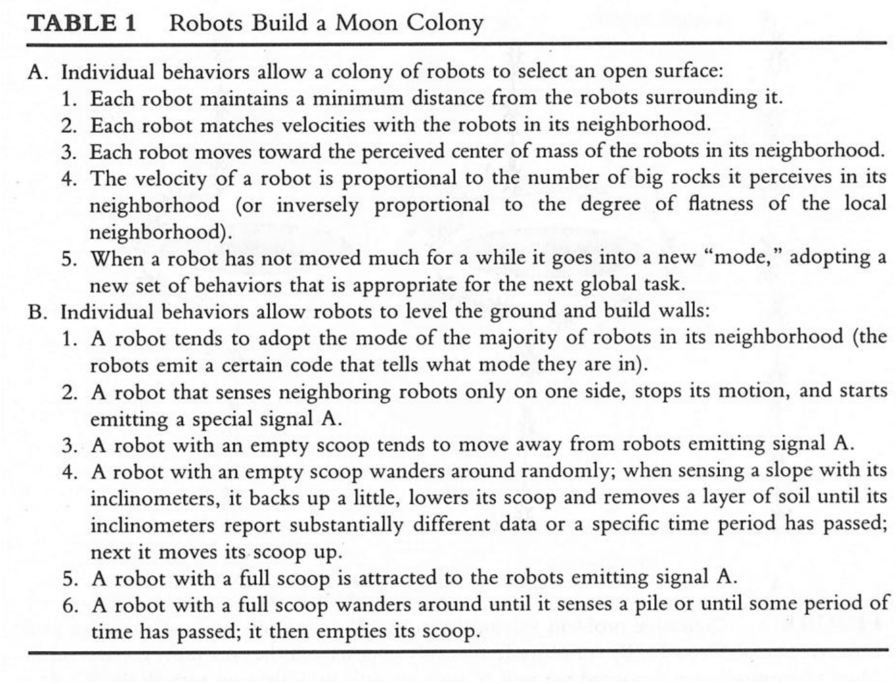

TABLE 1 Robots Build a Moon Colony

A. Individual behaviors allow a colony of robots to select an open surface:
1. Each robot maintains a minimum distance from the robots surrounding it.
2. Each robot matches velocities with the robots in its neighborhood.
3. Each robot moves toward the perceived center of mass of the robots in its neighborhood.
4. The velocity of a robot is proportional to the number of big rocks it perceives in its neighborhood (or inversely proportional to the degree of flatness of the local neighborhood).
5. When a robot has not moved much for a while it goes into a new "mode," adopting a new set of behaviors that is appropriate for the next global task.

B. Individual behaviors allow robots to level the ground and build walls:
1. A robot tends to adopt the mode of the majority of robots in its neighborhood (the robots emit a certain code that tells what mode they are in).
2. A robot that senses neighboring robots only on one side, stops its motion, and starts emitting a special signal A.
3. A robot with an empty scoop tends to move away from robots emitting signal A.
4. A robot with an empty scoop wanders around randomly; when sensing a slope with its inclinometers, it backs up a little, lowers its scoop and removes a layer of soil until its inclinometers report substantially different data or a specific time period has passed; next it moves its scoop up.
5. A robot with a full scoop is attracted to the robots emitting signal A.
6. A robot with a full scoop wanders around until it senses a pile or until some period of time has passed; it then empties its scoop.

Another application of the swarm intelligence metaphor is described by Colorni et al. (1992) and relies on the observation that each ant lays down traces of pheromone on the paths it travels and that these traces incite the other ants, that otherwise would wander randomly, to follow this same path. These new ants in turn lay down pheromone traces on the path in question and reinforce its attractive force. These individual behaviors generate an interesting collective property: that of making it possible to identify the shortest path around an obstacle (Figure 3). It indeed appears that, given equal individual speeds of locomotion, a larger number of ants covers path BCD than path BHD per unit time. The attractive force of path BCD is accordingly more strongly reinforced than that of path BHD, with the result that, little by little, the ants will follow only the shortest path. This property is exploited in a program that seeks the optimum solution to the Traveling Salesman Problem. This problem is actually solved collectively by an colony of ants that are turned loose on the network of towns and mark the paths they explore.

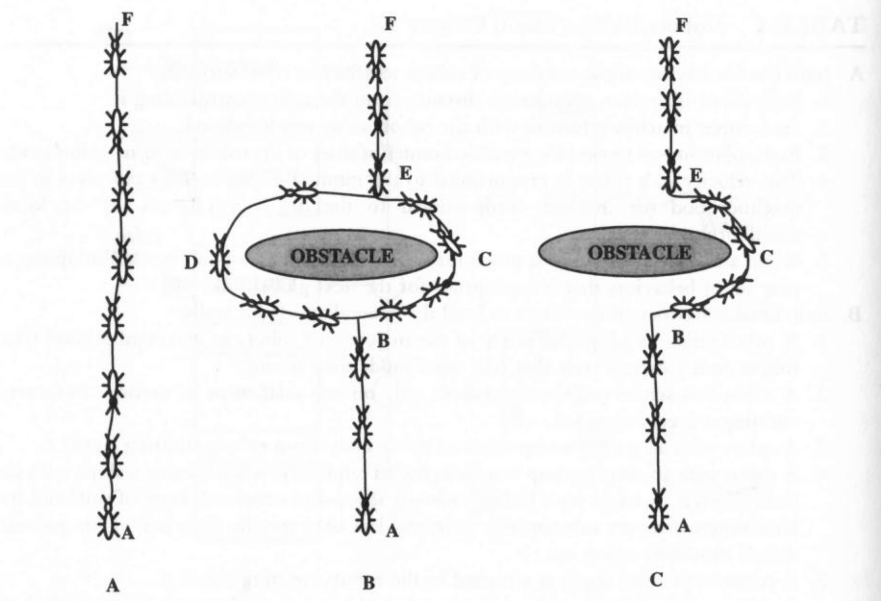

FIGURE 3 Collective problem solving by ants (A) Some ants are walking on a path between points A and F (B) An obstacle suddenly appears and the ants must get around it. More pheromones are deposited per unit of time on path BCE than on path BDE (C) At a steady state, the ants follow the shortest path (Adapted from Colorni et al , 1992, Figure 1, p 135)

Learned Behaviors

Numerous studies address the way in which an animat can improve the adaptive character of its behavior to the extent that it experiences new situations in its environment. From this perspective, two types of learning can be claimed to be biologically realistic: unsupervised learning -- which permits an animat to learn, to memorize, and to reconstruct a pattern by associating the various parts of this pattern with one another -- and learning by reinforcement, which permits an animat to learn to recognize and to favor those behaviors which yield rewards rather than those which yield disincentives.

- Learning in Neural Networks

This type of learning usually involves artificial neurons each one characterized by a set of input signals X_1, X_2, … X_n and an output signal, OUT (Figure 4). The inputs, collectively referred as vector , correspond to the signals into the synapses of the neurons. Each signal is multiplied by an associated weight W_1, W_2,… W_n, and corresponds to the synaptic strength of each connection. It is applied to a summation block which, roughly speaking, simulates the biological cell body, and adds all of the weighted inputs algebraically, producing the NET output: NET = X.W.

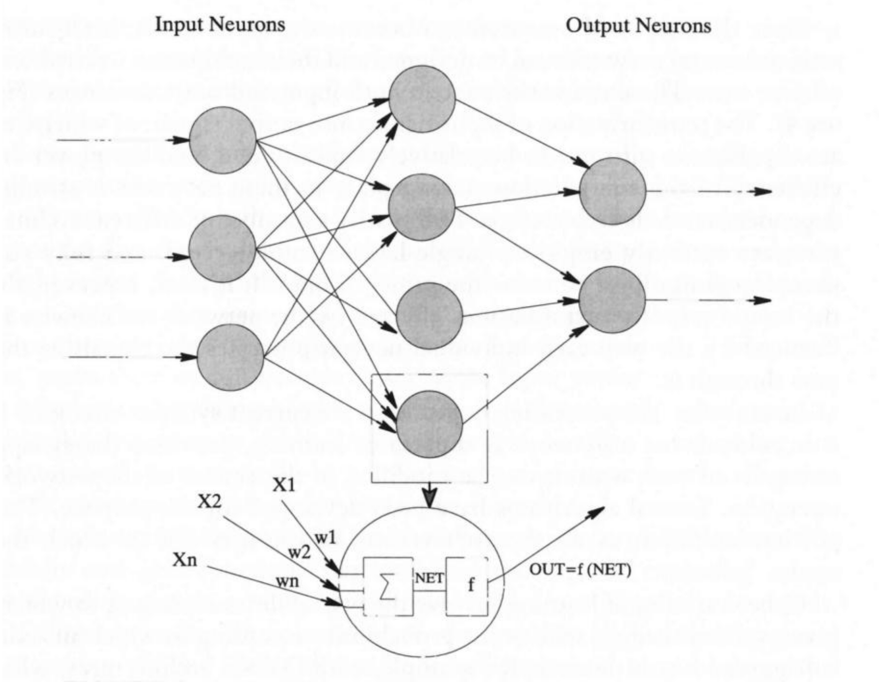

FIGURE 4 An artificial neural network with input and output neurons

The NET signal is usually further processed by an activation function F to produce the neuron's output signal OUT. According to the case considered, the activation function may be:

a linear function: OUT = K . NET, K = constant;

a function with a threshold: OUT = 1 if NET > T; otherwise OUT = 0, T = constant;

any sort of non-linear function, such as
the logistic function: OUT = 1 / (1 + e $^{-NET}$)
or the hyperbolic tangent function: OUT = tanh(NET).

Naturally, more complex operational laws have been proposed by many authors in the interest of enhancing the biological realism of such artificial neurons.

Insofar as the output of a neuron can become the input to another neuron, artificial neural networks can be designed and their architectures varied with relative ease. These networks contain both input and output neurons (Figure 4). The transformation of input signals into output signals of which they are capable can turn out to be relatively sophisticated, and the power and efficiency of the computations taking place in these networks is strongly dependent upon their architecture. This is why a number of different architectures are currently employed: single-layered, multi-layered and fully connected organizations, to name the principal ones. It is clear, however, that the input/output transformations effected by the network are likewise influenced by the way each individual neuron processes the signals as they pass through it.

Insomuch as the processing depends upon the current synaptic strengths W, it is evident that a network is capable of learning, provided the synaptic strengths of each neuron can be modified in the course of the network's operation. Several algorithms have been developed for the purpose. Their efficiencies differ, as do the architectures of the network to which they apply.

Other varieties of learning involve the probabilities according to which a given system changes state or the probabilities according to which an action is triggered -- as is the case, for example, with DYNA architectures, which will be discussed later.

- Unsupervised Learning

An example of unsupervised learning is provided by the work of Nehmzow and Smithers (1991) who use a Kohonen Net (Kohonen, 1988) to allow various robots to distinguish and recognize landmarks in their environment.

A Kohonen Net is composed of a certain number of artificial neurons, each of which receives the same input vector **i**. The output $\mathbf{o}_j$ of the neuron j is determined by the scalar product of the input vector **i** and the weight vector **w** of neuron j and is given by $\mathbf{o}_j = \mathbf{i}.\mathbf{w}_j$

At initialization, the weight vectors are chosen at random and are all distinct. It follows that when a certain entry vector is transmitted to the network, there must exist a neuron whose output is stronger than those of the other neurons. The output from the network is then given as a simple binary +1 response from the winner and there is no output from any other neuron. In effect, the winning neuron represents the category that the input pattern belongs to.

Unlike most other neural networks, in a Kohonen network only the winning neuron and its physical neighbors (within a distance from the winner which is designed by the network designer) modify the weights of their connections; the remaining neurons experience no learning. The training law used by the network is $\mathbf{\Delta W} = \boldsymbol{\eta}.\ (\mathbf{i} - \mathbf{w}_j)$, where is a learning gain that determines the amount of change experienced by the elements of vector .

Since both the input and the weight vectors are usually normalized to a unit magnitude, each of them points to a position on a unit hypersphere. Therefore, the winning neuron is the one with its weight vector closest to the input vector on this hypersphere and the result of a training pass is to nudge its weight vector closer to the input vector, with the size of the nudge depending on the learning gain. The winning neuron's physical neighbors also adjust their weights by applying the same learning equation as above. Thus, they too move their weight vectors closer to the input vector.

After several training passes, not only does the network give typically dissimilar responses to dissimilar input vectors, but also develops regions which respond most strongly to particular types of input vectors. In other words, the network implements a kind of primitive map.

The robots of Nehmzow and Smithers posses a control architecture which exhibits emergent functionalities useful for movement and for exploring an unknown environment -- such as avoiding obstacles, escaping from cul-de-sacs, or following walls. These robots are also capable of distinguishing and recognizing various landmarks in their environments, notably those which appear in the form of convex or concave corners. For example, when a robot explores a territory bordered by a wall by following the wall, a convex corner is detected if the time the robot needs to turn towards the wall exceeds a certain threshold time. Likewise, a concave corner is detected if the time it takes the robot to get away from a detected obstacle exceeds the threshold time.

The control architecture used to map the environment is a Kohonen net implemented in the form of a 50 neuron ring (Figure 5A), and the neighborhood within which the weight vectors are updated is ±2 neurons.

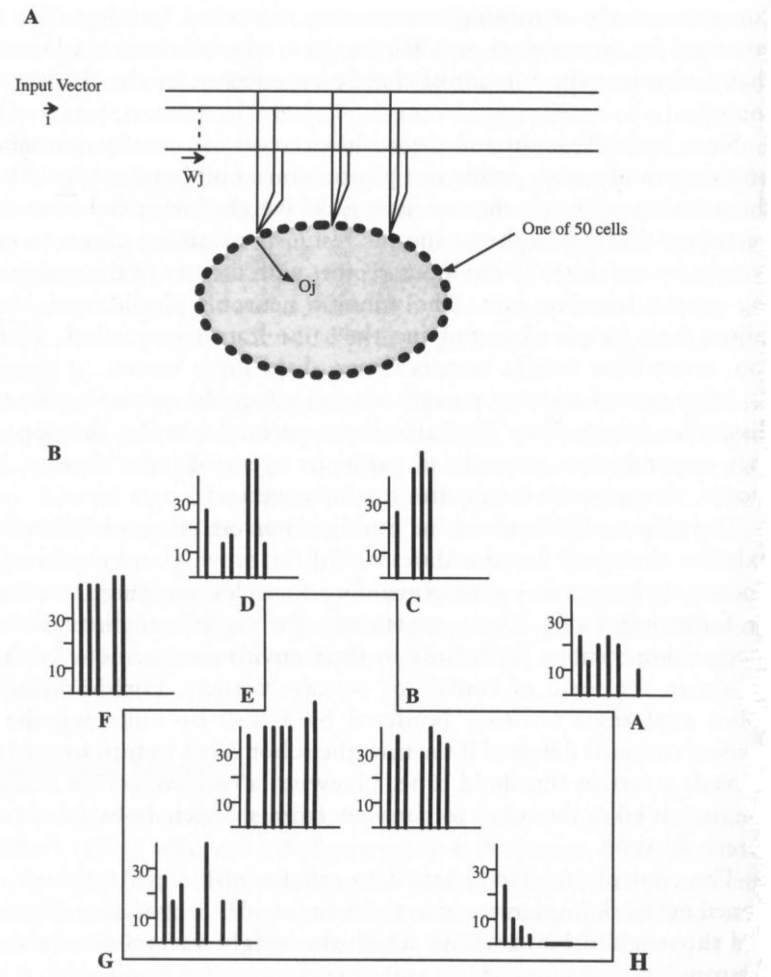

FIGURE 5 (A) The ring of 50 cells used by Nehmzow and Smithers as a self-organizing network. (B) The recognition of corner H when using information about the previous sensor readings Histograms show the absolute difference between the memorized pattern of cells' activation at every corner and the new pattern obtained at corner H Such differences are smaller at point H, indicating that the corner is recognized. (Adapted from Nehmzow and Smithers, 1991, Figures 3 and 7, pp 154 and 156)

Typically, the input vector given to the network describes whether the actual landmark is a convex or concave corner and whether the previous encountered landmarks were convex or concave corners. Odometry information about the distances between landmarks is also entered into the network. Given such information, it appears that the unsupervised learning procedure just described leads a given cluster of neurons to be more active than other neurons when the robot is in a particular place, where it receives a specific input signal, and leads some other cluster of neurons to be the most active in another place where the input signal is different. Thus, the robot is capable of categorizing its environment, i.e., to distinguish, for example, landmarks A to H of figure 5B. Moreover, once the corresponding learning is accomplished, the robot is capable of moving about the environment and recognizing at any given moment where it is situated. It is possible that such an aptitude has a certain amount in common with the capacity of bees to use landmarks to guide their way and locate food sources (Cartwright and Collett, 1983).

- Reinforcement learning

The typical scenario in reinforcement learning is one in which the animal receives a special signal from the environment -- called a reward, although it may be either an incentive or a disincentive -- and must learn to coordinate its actions so as to maximize its cumulative reward over time. In other words, reinforcement learning is a kind of learning by trial and error from performance feedback. Among the numerous architectures which allow the implementation of learning by reinforcement, the DYNA architecture proposed by Sutton (1991) is original in introducing an internal world model and allowing for mental activities generally considered to be "cognitive", such as reasoning and planning.

A DYNA architecture relies essentially on four interacting components (Figure 6):

- the real world that changes state in relation with the animat's behavior and that distributes incentives and disincentives:

- the world model that the animat elaborates for itself and that is intended to mimic the one-step input/output behavior of the real world;

- the policy function relied on by the animat to determine what action to engage in response to each possible state of the world;

- a primitive reinforcement learner that improves the policy function over time.

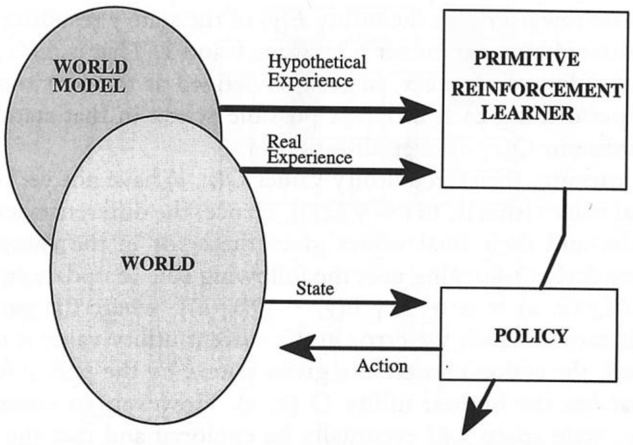

FIGURE 6 The four components of DYNA architectures.

In a particular implementation of DYNA architectures, called DYNA-Q, the reinforcement learner makes use of the Q-learning algorithm (Watkins, 1989). Such algorithm uses a utility function Q(x,a) which represents the utility of performing action a in state x . This utility is defined as the sum of the immediate reward r plus the utility E(y) of the state y resulting from the action discounted by a parameter γ between 0 and 1. That is, $Q(x,a) = r + \gamma E(y)$. The utility of a state x, in turn, is defined as the maximum of the utilities of doing each different possible action in that state. That is, E(x) = maximum Q(x,a) over all actions a.

During learning, the stored utility values Q(x,a) have not yet converged to their final values (that is, to $r + \gamma E(y)$). Hence, the difference between the stored values and their final values gives the error in the current stored values. Therefore, Q-learning uses the following rule to update stored utility values: $\Delta Q(x,a) = \alpha (r + \gamma e(y) – Q(x,a)$. The parameter α controls the rate at which the error in the current utility value is corrected.

In general, the action chosen in a given state x by the policy function is the one that has the highest utility Q(x,a). However, to ensure that all states in the state space will eventually be explored and that the utility of every pair (x,a) will be assessed, some randomness is actually introduced in the choices of the policy function.

The novel aspect of DYNA architectures is their inclusion of an internal world model along with mechanisms for learning it. The world model is a function that describes which actions are produced in a real world and which rewards are obtained in response to each possible action of the animat. Consequently, every time an action is performed by the animat and a reward is obtained, the state of the real world changes and the corresponding triple (action : reward : state change) is recorded in the world model. In this context, the Q-learning procedure described previously can be applied to real experiences in the real world as well as to hypothetical experiences in the world model. Such hypothetical experiences, which call on the metaphor of "planning as performing thought experiments," can be interleaved with experiences in the real world.

The results obtained by Sutton with a DYNA architecture demonstrate that it is possible for an animat to learn to reach a goal as directly as possible in an environment encumbered with obstacles. For instance, in the case of Figure 7, the animat's task is to navigate through the maze from the starting state S to the goal State G. In each state there are four possible actions: UP, DOWN, LEFT, and RIGHT, which change the state accordingly, except where the corresponding movement would take the animat into an obstacle—in which case the state is not changed. Reward is 0 for all transitions, except for those leading into the goal state, for which it is +1.

Each time the animat reaches the goal state, thus ending a learning trial, it is placed back at S and starts a new trial. Therefore, the animat's objective is to learn a policy that improves over trials and that ultimately enables it to obtain the maximum rate of reward over time. Incidentally, such a policy will correspond to the discovery of the shortest path leading from S to G.

In this context, although learning from real experiences updates the utili ty function according to the actual consequences of each action, planning updates this same utility function for simulated transitions chosen from the world model. Several such updates can be performed between each actual step taken in the real world. However, it should be noted that the world model, on the contrary, is updated only for actual transitions from state to state.

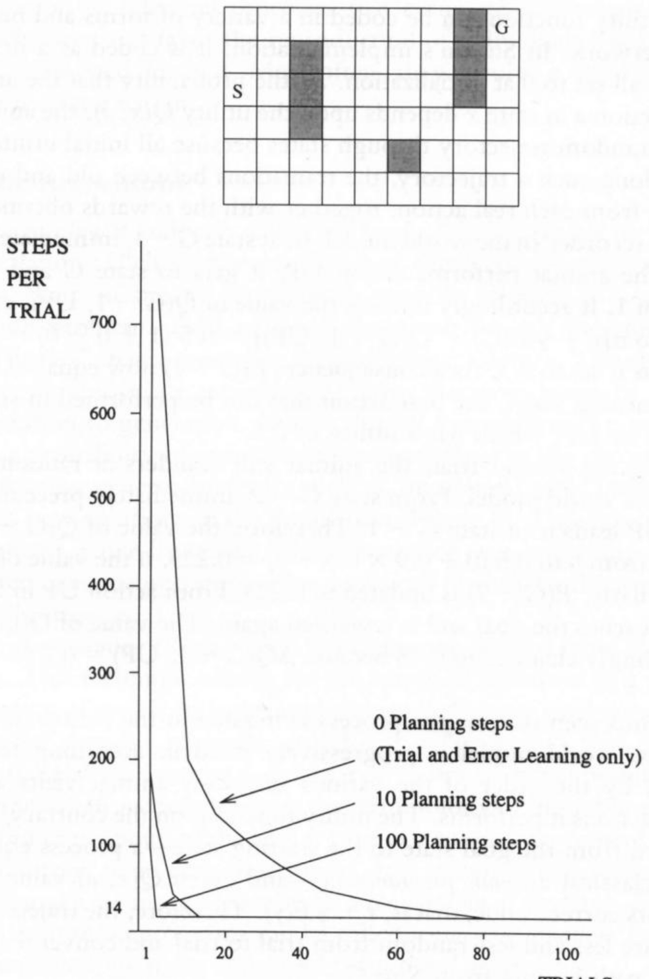

FIGURE 7 An example of the learning curves obtained in a specific navigation task with DYNA architectures The more hypothetical experiences (*k* planning steps) using the world model, the faster an optimal path was found These data were averaged over 100 runs (Adapted from Sutton, 1991, Figure 4, p 292)

The utility function can be coded in a variety of forms and notably as a neural network. In Sutton's implementation, it is coded as a matrix with elements all set to 0 at initialization. As the probability that the animat will choose action *a* in State *x* depends upon the Utility *Q(x, a),* the animat's first trial is a random trajectory through states because all initial utilities are the same. Along such a trajectory, the transitions between old and new states resulting from each real action, together with the rewards obtained in each state, are recorded in the world model. If, at state G - 1, immediately preceding G, the animat performs action UP, it gets to state G and receives a reward of 1. It accordingly updates the value of Q(G -1, UP), changing it from 0 to α[r + *y E(G)* - Q(G - 1, UP)] = 0.5 (1 + 0 - 0) = 0.5, if the value of a is set to 0.5. As a consequence, *E(G* - 1) now equals 0.5 because, at this learning stage, the best action that can be performed in State G - 1 seems to be UP, which has a utility of 0.5.

During the second trial, the animat still wanders at random and still updates its world model. From State G - 2, immediately preceding G - 1, action UP leads it to state G - 1. Therefore, the value of Q(G - 2, UP) is updated from 0 to 0.5 (0 + 0.9 x 0.5 - 0) = 0.225, if the value of 7 is set to 0.9. Similarly, *E(G* - 2) is updated to 0.225. From action UP in G - 1, the animat reaches the goal and is rewarded again. The value of Q(G - 1, UP) is accordingly changed to 0.75 because ΔQ(G - 1, UP) = 0.5 (1 + 0 - 0.5) = 0.25.

It is thus seen that, as the process is iterated in the course of successive trials, the world model is progressively filled in according to an order imposed by the order of the various states the animat visits and of the various actions it performs. The utility function, on the contrary, is updated backward from the goal state to the starting state—a process equivalent to that of classical *dynamic programming*—and every *Q(x, a)* value converges toward its correct value; that is, *r + y E(y).* Therefore, the trajectories of the animat are less and less random from trial to trial and converge toward the shortest path leading from *S* to G.

Hypothetical experiences allow for the same adjustments of utility estimates as real experiments. However, although the latter can, for example, adjust Q(G - 2, UP) only when the animat moves from G - 2 to G - 1, the former can make such an adjustment even when the animat is situated somewhere else. Therefore, planning allows for a quicker back-propagation of utility estimates and accelerates learning, as evidenced by the learning curves on Figure 7.

Evolved behaviors

Following the work of Holland (1975), a number of research efforts have addressed the simulation of processes which "improve" the behavior of individuals in a population from generation to generation. These efforts involve the implementation of selection processes which eliminate individuals with ill-adapted behaviors and favor the reproduction of individuals displaying behaviors which are well-adapted. Most often, they involve a classical Genetic Algorithm or some variant.

- Genetic Algorithms

A *genetic algorithm* (Goldberg, 1980; Holland, 1975) manages in parallel some population of "chromosomes" each of which codes -- generally in the form of a chain of binary symbols -- a possible solution to a particular optimization problem. Each of these chromosomes can therefore be assigned a "fitness" that assesses the corresponding solution. The application of the genetic algorithm accordingly causes this population to "evolve" from generation to generation. It does this by maintaining, for each chromosome, a probability of reproduction proportional to the chromosomes fitness and by using genetic operators such as "mutation" and "crossover" to give rise to new solutions in the population (Figure 8). This type of evolutionary process causes chromosomes of ever-increasing fitness to be generated until the optimum value is reached, or sufficiently nearly so for all practical purposes.

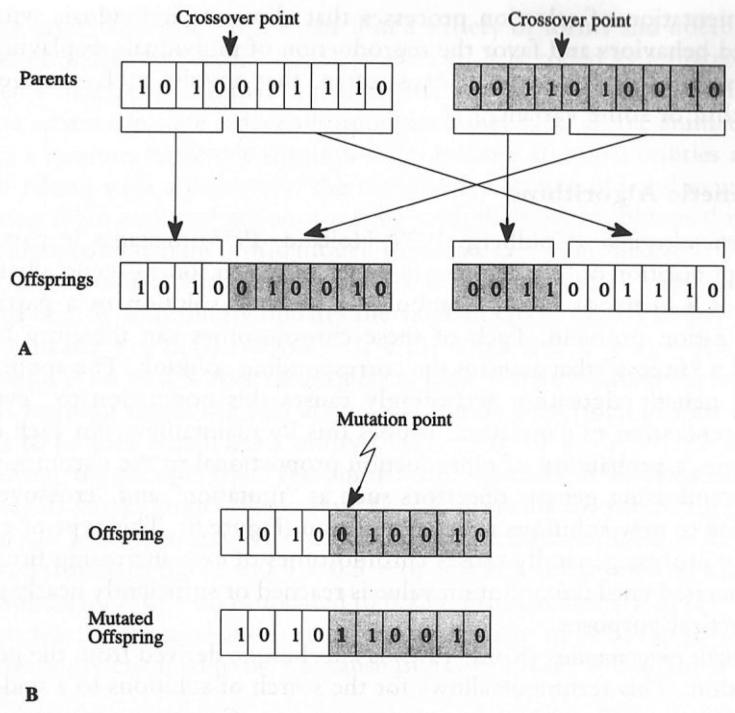

FIGURE 8 How new solutions are discovered by the genetic algorithm: (A) Role of the crossover operator. A crossover point is chosen at random within the chromosomes of the parents. Before that point an offspring inherits the genetic material of one parent. After that point it inherits the genetic material of the other parent. (B) Role of the mutation operator. A mutation point is chosen at random within the chromosome of an offspring and the binary value at that point is swapped.

Genetic programming (Koza, 1990) is a technique derived from the genetic algorithm. This technique allows for the search of solutions to a wide and diverse variety problems in computer science. Genetic programming can essentially be thought of as the search for a "program" which permits a computer to produce a certain desired set of outputs when presented with certain inputs. In accordance with the terminology of the application, this program can correspond to the plan of action of a robot, to a strategy of control, to a decision tree, to an economic model, to a system of equations regulating state changes, or, more generally, to a *composition of functions*. The inputs associated with these programs can be called *sensory information, independent variables, attributes,* or, more commonly, *functional arguments*.

The programs sought in genetic programming are symbolic expressions representing compositions of functions - like those of the LISP programming language. The solutions that are tested and improved by the algorithm consist of trees whose nodes can be functions or terminal symbols. These functions and symbols belong to sets predefined by the programmer aeccording to the problem to be solved. For instance, these funetions can be simple arithmetic functions, Boolean operators, functions specific to the domain under consideration, control structures such as IF . . . THEN . . . ELSE, or iterative structures like REPEAT . . . UNTIL. The terminal symbols, similarly, can be variables - such as the state variables of the system - or input data or constants. Figure 9 illustrates how, given the predefined set of logical funetions AND, OR, and NOT and a predefined set of Boolean-valued terminals DO and Dl, genetie programming can discover a program that calculates the XOR (that is, the *exclusive* OR) of the two logical variables D0 and D1 thanks to a crossover between two incorrect solutions.

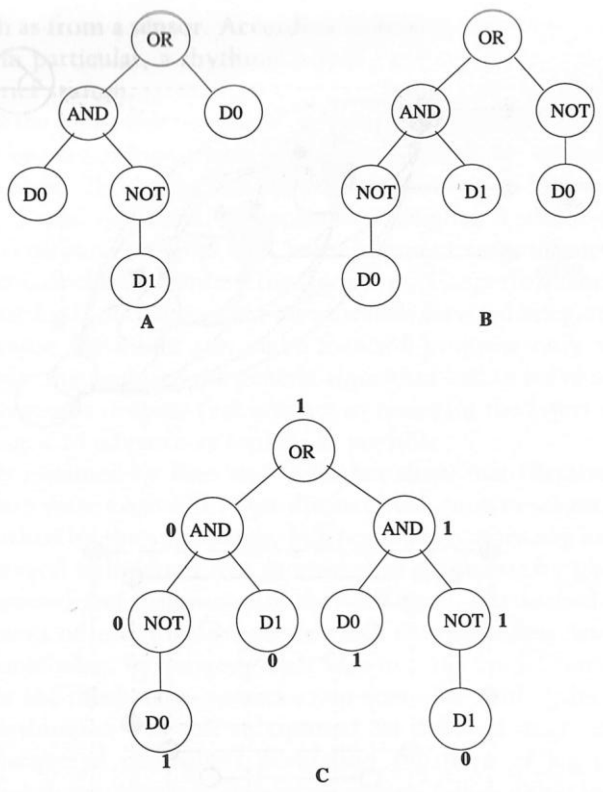

FIGURE 9 How a program able to evaluate the XOR of two logical variables $D0$ and $D1$ can be discovered by genetic programming. The correct solution (C) is obtained from incorrect solutions (A) and (B) by a crossover operation that exchanges two branches of the parents' trees. There are four possible combinations of the two variables $D0$ and $D1$, and the upper node of each tree gives the solution proposed by the corresponding program; that is, the XOR value of each possible combination of $D0$ and $D1$. For instance, tree C suggests that, when $D0 = 1$ and $D1 = 0$, the corresponding XOR is 1, which is true. It is easy to verify that tree C gives four correct answers out of four possible and that its fitness can accordingly be evaluated as 4. Likewise, it can be seen that the fitness of tree A is 2 and that of tree B is 1.

- Evolving the Control Parameters of a Neural Net

Beer and Gallagher (1992) considered how to discover, with the aid of a genetic algorithm, a combination of parameters which permit a neural net to control effectively the walking of an artificial insect. To control the movements of the six legs of this insect, they used a a continuous-time, recurrent network consisting of six input neurons, eighteen motor neurons, and twelve interneurons (Figure ...). The input neurons detect the angular position of each leg. The motor neurons govern the force with which each leg is propelled forward or backward, and whether or not the corresponding foot is set down. The role of the interneurons is unspecified. The instantaneous activity of each neuron is regulated by the equation:

$$\tau_i \frac{d\gamma_i}{dt} = -\gamma_i + \sum_{j=1}^{N} w_{ji}\, \sigma_j\, \gamma_j + I_i\,(t)$$

where N is the number of neurons j connected to neuron i .

In this equation, γ can be interpreted as the mean membrane potential of the neuron; $\sigma_j\ (\xi) = (1 + e^{(\theta j\ -\ \xi)})\ ^{-1}$ is a sigmoidal function giving the short-term average firing frequency of the neuron; θ is a bias term that controls the firing threshold; τ is a time constant associated with the passive properties of the cell membrane; w_{jj} represents the strength of the connection from the jth to the ith neuron; and I_1(t) represents an external input to the network such as from a sensor. According to such an equation, each neuron can exhibit, in partitular, a rhythmic activity and temporally extended responses to brief stimuli.

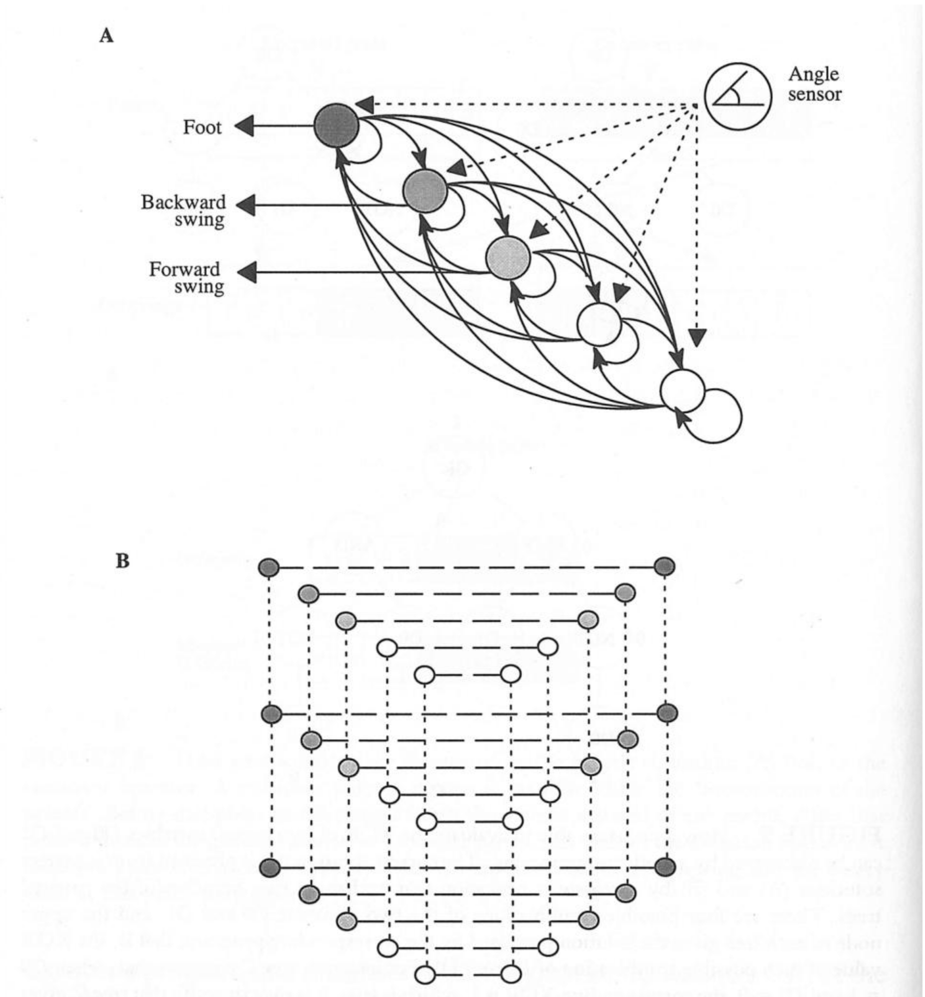

FIGURE 10 Locomotion controller network architecture: (A) Architecture of a simple leg controller. (B) Coupling between leg controllers. Identically shaded neurons have identical time thresholds and time constants. Dashed lines represent cross-body connections, and dotted lines represent intersegmental connections. Connections between identically shaded neurons with the same line style have identical weights. All coupling connections are symmetrical. (Adapted from Beer and Gallagher, 1992, Figure 10, p. 107.)

Because of the back-front and left-right symmetries of this control architecture, only one set of leg-control parameters needs to be optimized by the genetic algorithm. To accomplish this, a combination of 50 parameters -5 thresholds, 5 time constants, 25 connection weights, 5 sensor weights, 5 cross-body connection weights, and 5 intersegment connection weights - were coded in each chromosome of the algorithm. The performance measure to be maximized was the distance the insect travels forward in a given amount of time. Because the insect can make forward progress only when it is statically stable, the problem the genetic algorithm had to solve was that of generating 18 motor outputs that interact to maintain the insect in balance while allowing it to advance as rapidly as possible.

The results obtained by Beer and Gallagher show that effective locomotion controllers were evolved in four distinct evolutionary stages. The first stage was marked by the appearance, in a population of insects incapable of motion, of several individuals that succeeded in advancing by placing their feet on the ground and pushing until they fell over. The second stage saw the development of individuals that, although they fell often, succeeded in advancing nonetheless by moving their legs in a rhythmic, but uncoordinated way. In the third stage, insects using statically stable gaits appeared, but their coordination was still suboptimal. In the final stage, the genetic algorithm discovered controllers generating a pattern of leg movement known as a *tripod gait,* which is ubiquitous among fast-walking insects. The discovery of this form of locomotion is owed to the fact that the performance measure puts pressure not just on the development of locomotion controllers per se but on the development of controllers that cause the insect to move as quickly as possible.

The controllers discovered by the genetic algorithm exhibit interesting adaptive capabilities. In particular, when the insect is allowed to evolve alternately in conditions in which it can avail itself of information provided by the angle sensors and in conditions in which it cannot - thus simulating a series of functional breakdowns - the genetic algorithm discovers mixed controllers. Although such mixed controllers exhibit a higher stepping frequency and clearer phasing when the sensors are intact than when they are not, the performances of the controllers are still quite good even in the latter case.

- Evolution of a Control Program

Koza, Rice and Roughgarden (1992) studied how an evolutionary process could help the Caribbean Anolis Lizard to optimize its hunting strategies.

This animal is a "sit-and-wait" predator, typically perching head-down on tree trunks and scanning the ground for desirablc insects to chase and eat. It can be supposed that the insects the lizard eats appear infrequently enough that the lizard starts every chase from its perch. It can also be supposed that the evolutionary process thus hypothetized tended to favor those lizards that avoided losing time in attacking insects that were too far away. The problem of developing an optimal hunting strategy can therefore be viewed as a search for a control program permitting the lizard to decide whether or not to attack its prey according to how far away the prey is and to how densely the prey populates the hunting territory under consideration.
The search for such a program can be conducted using genetic programming. To do this, Koza et al. decided that the programs thus generated will use the four variables *X, Y, AB,* and *VEL* for terminal symbols or input data, in addition to numerical constants. The variables *X* and *Y* represent the

coordinates of the prey within the lizard's planar field of vision. The variable *AB* represents the average density of prey in a certain area (expressed as the probability of appearance per unit of surface area in a singlc unit of time), and *VEL* represents the lizard's sprint velocity. The programs were also permitted to use arithmetic functions such as addition, subtraction, multiplication, and division, in addition to mathematical operations like exponentiation or to control structures such as IF . . . THEN . . . ELSE.

Thus, by convention, the program

```
(IFLTE (SREXPT (% (* 3.0 VEL) (* 3.14159 AB)) (% 1.0 3.0)) (SREXPT (+ (SREXPT x 2.0)
(SREXPT Y 2.0)) (% 1.0 3.0)) -1.0+1.0)
```

corresponds to a control strategy more simply expressed as

$$\text{IF } \left(\frac{3.0 \times VEL}{\pi \times AB}\right)^{1/3} \leq (X^2 + Y^2)^{1/2} \text{ THEN ignore OTHERWISE chase.}$$

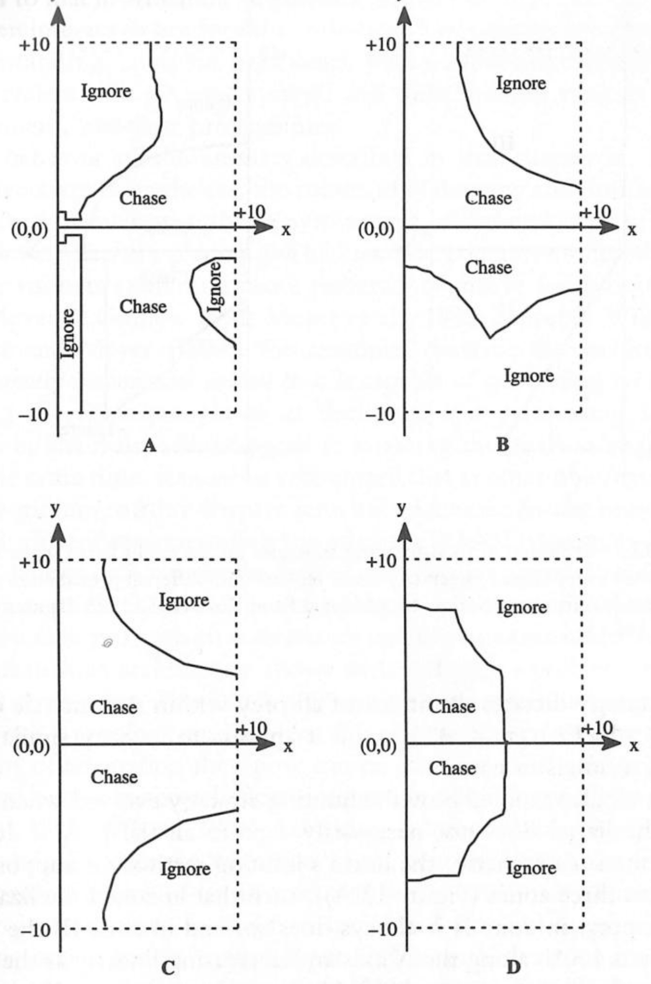

Figure 11 Evolution of food-foraging strategies for the anolis lizard when it necessarily catches every chased insect: (A) Individual at generation 0, scoring 1,235. (B) Best individual from generation 0, scoring 1,460. (C) Best individual from generation 10, scoring 1,514. (D) Best individual from generation 60, scoring 1,625. (Adapted from Koza et al., 1992, Figures 7, 8, 9, and 11, pp. 186, 187, 188, and 190.)

However, the former formalism presents the advantage over the latter of readily accommodating mutations and crossing over to generate new programs that continue to describe meaningful control strategies.
The fitness of each program was evaluated by simulating the strategy it implemented in various velocity and insect frequency combinations and then counting the total number of insects it allowed the lizard to consume over the course of the simulation. A population of 1000 such programs was allowed to evolve freely.

In a first approach to the optimization problem studied here, it was supposed that the lizard always captures the prey it chases. In such conditions, it turned out that a control program randomly generated when the system was initialized allowed the lizard to capture 1235 insects. The corresponding strategy is represented graphically in the *X*, *Y* space in Figure 11(A). Another individual initialized at random scored 1460 (Figure 11(B)). By generation 10, a control program capturing 1514 insects (Figure 11(C)) was discovered. By generation 60, a control program capturing 1625 insects appeared (Figure 11(D)). This program defines a strategy very similar to the optimal strategy determined by a mathematical model of optimization. This optimal strategy dictates the attack of all prey within a semicircle of radius $R = [(3.0 \,.\, VEL)/(\pi \,.\, \mathrm{AB})]^{1/3}$, and it appears to be very similar to that applied by animals in nature.

Koza et al. also studied how the hunting strategy evolved when supposing that the lizard does not necessarily capture all the prey it decides to chase. In one such scenario, the lizard's hunting territory is supposed to be divided into three zones (Figure 12(A)), such that in zone I the lizard never catches its prey, in zone II it always does so, and in zone III the catching probability is 100% along the *X* axis and decreasing linearly as the angle of vision increases between 0° and 90°. Under such conditions, 46 generations of genetic programming led to the hunting strategy represented in Figure 12(B), which seems well adapted to the problem. As can be seen, not only does the lizard avoid an area that approximately corresponds to region I and chase insects in region II, but also the lizard is willing to travel a distance in region III that is all the greater as the angular location of the prey is closer to 0°.

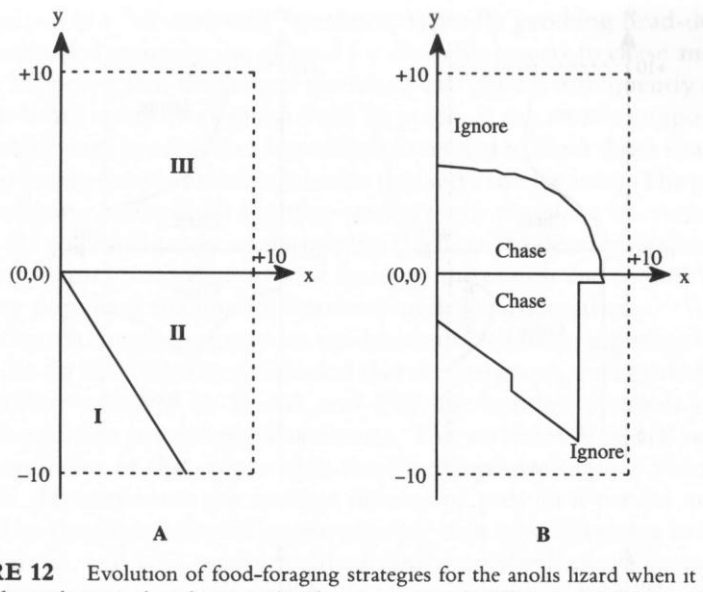

FIGURE 12 Evolution of food-foraging strategies for the anolis lizard when it does not necessarily catch every chased insect: (A) Three regions with different probabilities of capture. (B) Best individual from generation 46. (Adapted from Koza et al., 1992, Figures 13 and 16, pp. 195 and 196.)

Conclusion

The various efforts discussed here illustrate several of the objectives and methods of the animat approach. Particular instances were stressed where adaptive behavior appears as an emergent property at a level of integration superior to that in which the organisms, neural modules, genome segments, and so on interact to produce that behavior. The animats described here are, notwithstanding, complete organisms, with sensors and actuators linked by the equivalent of a nervous system; and these animats interact with their environment, not their programmer.

The behavior of the animats described in this chapter is, in general, purely reactive. Nonetheless, the robots of Nehmzow and Smithers learn to categorize and memorize their environment, and the animats of Sutton are capable of elementary planning. The literature contains numerous examples of other animats exhibiting more elaborate cognitive faculties (Cliff et al., 1994; Meyer & Guillot, 1991; Meyer et al., 1993; Meyer & Wilson, 1991). Donnart and Meyer (1994), for example, describe the architecture of a *motivationally autonomous animat* that is capable of generating its own goals, assessing the consequences of its decisions, and performing actions that seem to be the most advantageous in attaining the goals in question.

At the same time, it must be recognized that another objective envisaged in the beginning of this chapter remains unattained in the present state of research: that of understanding the adaptive role of cognition. The reason for this deficiency is clearly the lack of comparative approaches that investigate, for example, how well such and such a problem of adaptation can be solved by such and such an architecture and not by some other, or how well such and such an architecture allows such and such a problem to be solved and not some other. From such a perspective, tremendous progress will be made once a typology and a hierarchy of both the environments and the problems of adaptation they pose can be established. Fortunately, the past few years have seen several efforts initiated in this direction (Horswill, 1992; Littman, 1993; Wilson, 1991). Likewise, substantial improvements in our understanding of human cognition could be gained from the five-year research project described by Brooks and Stein (1993) or by Dennett (1994). This project proposes to build an integrated physical humanoid robot - including active vision, sound input and output, dextrous manipulation, and the rudiments of language. Its objectives are to study how a serious exploration of the relationships between intelligence and the subjective experience of a body helps in understanding the modularity of the human mind, how its "representations" are grounded in the sensory modality used to learn them, and how consciousness, symbols, and language are interelated. In other words, this approach might help to bridge the evolutionary gap that, according to Kirsh (1991), separates the sensory-motor connections of the earwig from human concepts and representations.

References

Altman, J. S., & Kien, J. (1989). New models for motor control. *Neural Computation, 1,* 173-183.
Ashby, W. R. (1952). *Design for a brain.* London: Chapman & Hall.
Beer, R. D., & Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior, 1,* 91—122.
Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, RA-2(1),* 14-23.

Brooks, R. A. (1989). A robot that walks: Emergent behaviors from a carefully evolved network. *Neural Computation, 1*, 253-262.
Brooks, R. A. (1990). Elephants don't play chess. In P. Maes (Ed.), *Designing autonomous agents. Theory andpractice from biology to engineering and back* (pp. 3-15). Cambridge, MA: MIT Press.
Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence, 47,* 139-159.
Brooks, R. A., & Stein, L. A. (1993). *Building brains for bodies* (AI Memo No. 1439). Cambridge, MA: MIT, Artificial Intelligence Laboratory.
Cartwright, B. A., & Collett, T. S. (1983). Landmark learning in bees. *Journal of Comparative Physiology, 151,* 521-543.
Cliff, D., Husbands, P., Meyer, J. A., & Wilson, S. (Eds.). (1994). *From animals to animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: MIT Press.
Colorni, A., Dorigo, M., & Maniezzo, V. (1992). Distributed optimization by ant colonies. In F. J. Varela & P. Bourgine (Eds.), *Toward a practice of autonomous Systems. Proceedings of the First European Conference on Artificial Life* (pp. 134-142). Cambridge, MA: MIT press.
Connell, J. H. (1990). *Minimalist mobile robotics. A colony-style architecture for an artificial creature.* San Diego, CA: Academic Press.
Dennett, D. C. (1978). Why not the whole iguana? *Behavioral and Brain Sciences, 1,* 103-104.
Dennett, D. C. (1994). The practical requirements for making a conscious robot. *Philosophical Transactions of the Royal Society of London, Series A.* 349, pp. 133-146.
Donnart, J. Y., & Meyer, J. A. (1994). A hierarchical classifier System implementing a motiva-tionally autonomous animat. In D. Cliff, P. Husbands, J. A. Meyer, & S. Wilson (Eds.), *From animals to animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior* (pp. 144-153). Cambridge, MA: MIT Press.
Fraenkel, G. (1980). On geotaxis and phototaxis in Littorina. In C. R. Gallistel (Ed.), *The Organization of action: A new synthesis* (pp. 149-165). Hillsdale, NJ: Erlbaum.
Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.
Harnad, S. (1990). The Symbol grounding problem. *Physica D (Amsterdam), 42,* 335-346.
Holland, J. H. (1975). *Adaptation in natural and artificial Systems.* Ann Arbor: University of Michigan Press.
Holland, J. H. (1986). Escaping brittleness: The possibilities of general purpose machine learning algorithms applied to parallel rule-based Systems. In R. Michalski, J Carboneil, & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2, pp. 593-623) San Mateo, CA: Morgan Kaufmann.
Horswill, I. (1992). Characterizing adaptation by constraint. In F. J. Varela & P. Bourgine (Eds.), *Toward a practice of autonomous Systems. Proceedings of the First European Conference on Artificial Life* (pp. 58-63). Cambridge, MA: MIT Press.
Kirsh, D. (1991). Today the earwig, tomorrow man? *Artificial Intelligence, 47,* 161-184.
Kohonen, T. (1988). *Self-organization and associative memory.* Berlin: Springer-Verlag.
Koza, J. R. (1992). *Genetic programming: On the programming of Computers by means of natural selection.* Cambridge, MA: MIT Press.
Koza, J. R., Rice, J. P., & Roughgarden, J. (1992). Evolution of food-foraging strategies for the Caribbean Anolis lizard using genetic programming. *Adaptive Behavior, 1,* 171-199.
Langton, C. G. (Ed.). (1989). *Artificial life.* Reading, MA: Addison-Wesley.
Langton, C. G., Taylor, C, Farmer, J. D., & Rasmussen, S. (Eds.). (1992). *Artificial Life II.* Reading, MA: Addison-Wesley.
Levy, S. (1992). *Artificial Life. The quest for a new creation.* London: Jonathan Cape.
Littman, M. L. (1993). An optimization-based categorization of reinforcement learning envi-

ronments. Inj. A. Meyer, H. L. Roitblat, & S. W. Wilson (Eds.), *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (pp. 262-270). Cambridge, MA: MIT Press.

Maes, P. (1993). Behavior-based Artificial Intelligence. In J. A. Meyer, H. L. Roitblat, & S. W. Wilson (Eds.), *From animals to animats 2: Proceedings ofthe Second International Conference on Simulation of Adaptive Behavior* (pp. 2-10). Cambridge, MA: MIT Press.

Maes, P., & Brooks, R. A. (1990, May). Robot insect societies. *Data Manager Magazine*, pp. 1-6.

Meyer, J. A., & Guillot, A. (1991). Simulation of adaptive behavior in animats: Review and prospect. In J. A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings ofthe First International Conference on Simulation of Adaptive Behavior* (pp. 2-14). Cambridge, MA: MIT Press.

Meyer, J. A., Roitblat, H. L., & Wilson, S. W. (Eds.) (1993). *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: MIT Press.

Meyer, J. A., & Wilson, S. W. (Eds.). (1991). *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: MIT Press.

Nehmzow, U., & Smithers, T. (1991). Mapbuilding using self-organizing networks in "really useful robots." In J. A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 152-159). Cambridge, MA: MIT Press.

Newell, A. (1980). Physical symbol Systems. *Cognitive Science, 4,* 135-183.

Newell, A., & Simon, H. A. (1963). GPS, a program that simulates human thought. In E. A. Feigenbaum, & J. Feldman (Eds.), *Computers and thought* (pp. 279-293). New York: McGraw-Hill.

Newell, A., & Simon, H. A. (1976). Computer science as empirical enquiry: Symbols and search. *Communications of the ACM, 19,* 113-126.

Roitblat, H. L. (1987). *Introduction to comparative Cognition.* New York: Freeman.

Roitblat, H. (1995). Comparative approaches to cognitive science. In H. Roitblat & J. A. Meyer (Eds.), *Comparative approaches to cognitive science* (pp. 13-25). Cambridge, MA: MIT Press.

Roitblat, H., & Meyer, J. A. (Eds.). (1995). *Comparative approaches to cognitive science.* Cambridge, MA: MIT Press.

Shortliffe, E. H. (1976). *Computer based medical consultations: MYCIN.* Amsterdam: Elsevier.

Sibly, R. M., & McFarland, D. (1976). On the fitness of behavior sequences. *American Naturalist, 110,* 601-617.

Steels, L. (1991). Towards a theory of emergent functionality. In J.A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 451-461). Cambridge, MA: MIT Press.

Sutton, R. S. (1991). Reinforcement learning architectures for animats. In J.A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 288-296). Cambridge, MA: MIT Press.

Taylor, C. E. (1992). "Fieshing out" Artificial Life II. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial Life II* (pp. 25-38). Reading, MA: Addison-Wesley.

Watkins, C.J. (1989). *Learning with delayed rewards.* Unpublished doctoral dissertation, University of Cambridge, UK.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. Inj. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 16-23). Hillsdale, NJ: Erlbaum.

Wilson, S. W. (1991). The animat path to AI. In J. A. Meyer & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 15-21). Cambridge, MA: MIT Press.