Bill Coderre
Media Lab, Massachusetts Institute of Technology, 20 Ames Street, Cambridge, MA 02139

Modeling Behavior in Petworld

This paper describes a system called *Petworld* for modeling aspects of animal behavior. Behavior is modeled as a rigid hierarchy of simple agents that make recommendations to their superiors. Since recommendations are in the form of rankings of alternatives, agents can resolve conflicting behaviors either by straight choice, compromise, or displacement. Substantial improvements have been made since the Artificial Life conference in September, including a simple perception model and a method for concentration on long-duration tasks. Plans are discussed for the successor to Petworld, in which learning will be incorporated, among other new features.

INTRODUCTION

The Vivarium Project is a collaborative effort between Apple, MIT, and the Open School in Los Angeles. The aim of Vivarium is to teach grade- and high-school students animal behavior as an approach to learning about thinking. As part of Vivarium advanced research, Petworld is not specifically geared for school use, but

Artificial Life, SFI Studies in the Sciences of Complexity,
Ed. C. Langton, Addison-Wesley Publishing Company, 1988

bears in mind the issues of complexity, intuitiveness, and user interface. I claim that Petworld has enough sophistication to produce interesting behavior, without being so complex as to make creation of that behavior difficult.

Petworld is a system for modeling non-species-specific behavior in an intuitive environment. I divide the system into four parts.

- how the pets *perceive* the world;
- pet *locomotion*;
- the *physics* of the world; and
- pet *intellect.*

Although all the parts are important, I have concentrated on intellect, especially the structuring of knowledge to produce behavior. Therefore, I tried to keep the first three aspects simple while still providing a reasonable environment.

The first section of this paper contains a description of the system and its commands, and descriptions of a sample brain. The second section contains an analysis of Petworld's intellect system in light of other systems, including more traditional knowledge-based systems, dataflow systems, and Minsky's Society of Mind model.

SECTION ONE: THE PETWORLD SYSTEM

ABOUT THE PETWORLD

Petworld is a world of pets, rocks, and trees which inhabit a two-dimensional, limited cartesian plane. Time passes in discrete quanta, and simultaneous action by all the inhabitants is simulated. Each action executed in the world takes one tick, and brain calculations are instantaneous. This is similar to a discrete-time feedback system. Another possible model, one with longer-duration actions and computer-style interrupts, was considered. Although both offer about the same power, the feedback-style approach was chosen as more intuitive.

Pets have a body orientation. The pets have a limited field of view, typically 90 degrees. Also, they can move for a limited distance in the direction of their body orientation, typically about one unit. Pets cannot push or throw things, but can carry one rock at a time. (Pets often use rocks to build nests.) Trees are food sources, and pets are browsers. Every time a pet eats, the tree loses some meal points and vanishes when the count reaches zero. Trees also grow (accrue meal points) as time passes. New trees appear spontaneously around the play field. Famines are prevented by creating a new plant when the last one dies.

Several pets are usually living in the world at once. In general, pets are mutually antagonistic, and can attack each other. Pets cannot reproduce, but can die from starvation or wounds.

ABOUT PETS

Each pet has a very limited set of internal states which indicate the condition of the animal. These are the location of its nest, the variables HUNGER, FEAR, and INJURY, which have as values numbers between 0 and 100, and the flag PAYLOAD. A hunger of 0 indicates being sated, and a hunger of 100 indicates death by starvation. Eating reduces the count, and the passage of time increases the count. Fear is determined by the distance of other pets. A low fear indicates that others are far away, and a high fear tells proximity. Injury is caused when one pet attacks another. Each attack increases the pet's injury count, and with time the count goes back down. If an animal's injury count exceeds 100, the animal dies from its wounds. If the payload flag is true, the pet is carrying a rock.

Since time passes in quanta, pets perform a "SEE—THINK— DO" loop similar to a "READ—EVAL—PRINT" loop in Lisp. First, every pet is given a chance to perceive the world, then a chance to decide what to do, then all actions are performed. Since there is still an order in which pets act (which might be important when, say, there is very little food left), this order is shuffled occasionally.

The pets' actions are MOVE-TOWARDS, TURN, LIFT, DROP, EAT, and ATTACK.

ABOUT PET BRAINS

A pet brain is a hierarchy of modules that I call *experts*. Each expert has inputs from its subordinates and the world, and provides as output a *ranking*—a list of possible actions with numeric weights attached—telling how good those actions are in the opinion of the expert. The action with the greatest weight of the ranking that the topmost expert recommends is the one that the pet executes.

Decision information (in the form of rankings) can be thought to flow up from the bottom-most experts, getting processed along the way. Typical expert processing strategies involve assigning weights to rankings, filtering rankings to remove unwanted elements, merging two rankings while possibly emphasizing one more than another, and reducing the elements of a ranking to a single object. Thus, the ethological constructs of competitive, compromise, and displacement behaviors can be directly implemented.

A SAMPLE BRAIN

As an example of the Petworld system, I offer a sample pet which is capable of several basic behavior patterns: interacting with other pets, foraging for food, building a nest and staying in it once it is built, and exploring areas not visible.

The strategy of the pet is decided by the topmost expert, *brain5*. Finding food and interacting with other pets are most important, followed by building a nest, exploring the world, and homing into the nest. In the case of conflicting recom-

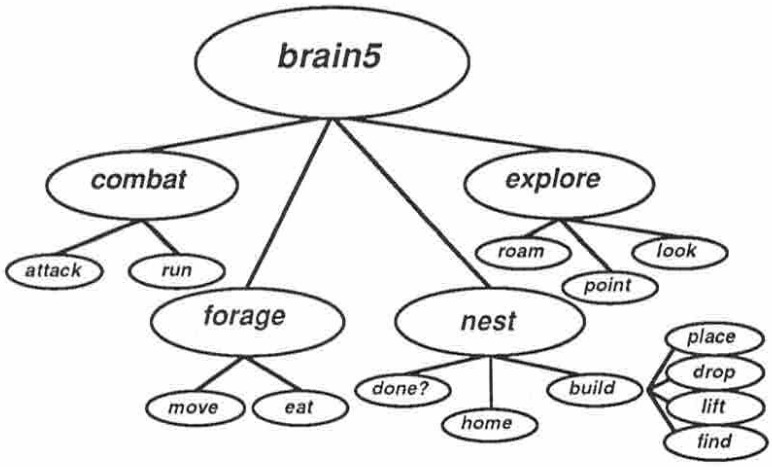

FIGURE 1 The sample brain.

mendations, several decision strategies are pursued. If there is a conflict between foraging and combat, a compromise is made, dependent on how serious the danger of starvation or attack is. Thresholds are also used to ignore experts. Assuming no life-or-death situations are imminent, nest building takes priority over exploring.

The basic *combat* strategy is to avoid other pets. Another pet is attacked only when the other is close enough and damage is low.

Foraging consists of either eating what is readily edible, or moving toward the closest available food.

Nesting is actually the most complex of the behaviors, incorporating several important functions. A functional expert called *done*? reports true, false, or can't tell dependent on the completeness of the nest. *Home* tries to get the pet to stay in the nest. And *build* models a strategy sometimes suggested for bird nest building: "If you are in your nest and have a rock, put the rock in the nest; otherwise get a rock and bring it back."

Last is the *exploring* expert. It is called by default, when nothing else works. Since pets have limited sensory range, they will often be in a spot where no useful objects are visible. In that case, they should first turn around, and if nothing becomes visible, go somewhere else. This mechanism makes good use of Petworld's feedback style of control, and of its history mechanism. As soon as the exploring expert brings something useful into view, another expert will be able to take control. And since the exploring expert is able to remember what it was doing previously, it can act as if it were performing long-duration actions.

Below are the actual rules from the brain, translated into English.

BRAIN5

1. If both HUNGER and FEAR are high, effect a tradeoff between COMBAT and FORAGE.
2. If FEAR is high, COMBAT.
3. If HUNGER is high, FORAGE.
4. If FORAGE is recommending that there is a food element immediately available, then FORAGE.
5. If NEST has some non-trivial action to perform, then NEST.
6. Otherwise, EXPLORE.

COMBAT

1. If you have an available attack, and your damage is low, then recommend a tradeoff of attacking and running away.
2. Otherwise, recommend running away from any visible pets.

ATTACK Construct a ranking of things you can attack standing right where you are.

RUN Construct a ranking which scores movements on how well they bring you away from other pets. *(This includes the current positions of the other pets, and does not take into account their potential movement.)*

FORAGE

1. If you are standing next to food, then recommend EAT.
2. Otherwise, recommend moving toward visible trees (MOVE). If none are visible, no recommendation is made.

MOVE Construct a ranking which scores movements on how well they bring you toward trees.

EAT Construct a ranking of things you can eat standing right where you are.

NEST

1. If DONE? returns true, then HOME.
2. Otherwise, BUILD.

DONE? *(The nest is done when an internal, hardwired template representation of the nest, when matched against the state of the world, appears complete.)*

HOME Return a ranking that gets the pet into the center of the nest.

BUILD *(In any of the following cases, if no forward progress can be made toward some destination, a displacement behavior of wandering randomly is undertaken.)*

1. If you have a rock, and the nest is completed, drop the rock. *(This in effect will drop it outside the nest.)*
2. If you have a rock, and you are standing in the right spot, then drop the rock. (DROP)
3. If you have a rock and and are not in the right spot, then move toward the right spot. (PLACE)
4. If you don't have a rock, and are standing next to one that is not part of your nest, then pick it up. (LIFT)
5. If you don't have a rock, and are not standing next to one that is not part of your nest, then move toward the closest rock that is not part of your nest. (FIND)

FIND Return a ranking scoring movements on how effectively they bring you towards the closest rock that is not already part of your nest. If no rock is visible, no recommendation is made.

PLACE Return a ranking of movements on how effectively they bring you towards the top-ranked spot in the nest to place a rock. *(The nest is defined as a ranking of squares in which to place rocks.)*

LIFT Return a ranking of possible lifting commands. Empty if no possible lifts.

DROP Return a ranking of possible dropping commands. Empty if no possible drops.

HOME Return a ranking scoring movements on how effectively they bring you towards "home" (the position on the board in which you started the simulation). The pet always knows where home is.

EXPLORE Exploration is allowed when no other strategy fires, probably due to a lack of visible objects. I have chosen a very simple exploration strategy: first turn around to look in all directions, then move in a random direction a few units and try again.

1. Recommend LOOK enough times in a row so that one complete revolution is made.
2. Recommend POINT once, to attain a random direction to travel in.
3. Otherwise, recommend ROAM several times in a row.

LOOK Turn clockwise by one field of view.

POINT Turn some random amount.

ROAM

1. If you can't move forward, turn right 90 degrees.
2. Otherwise, move forward.

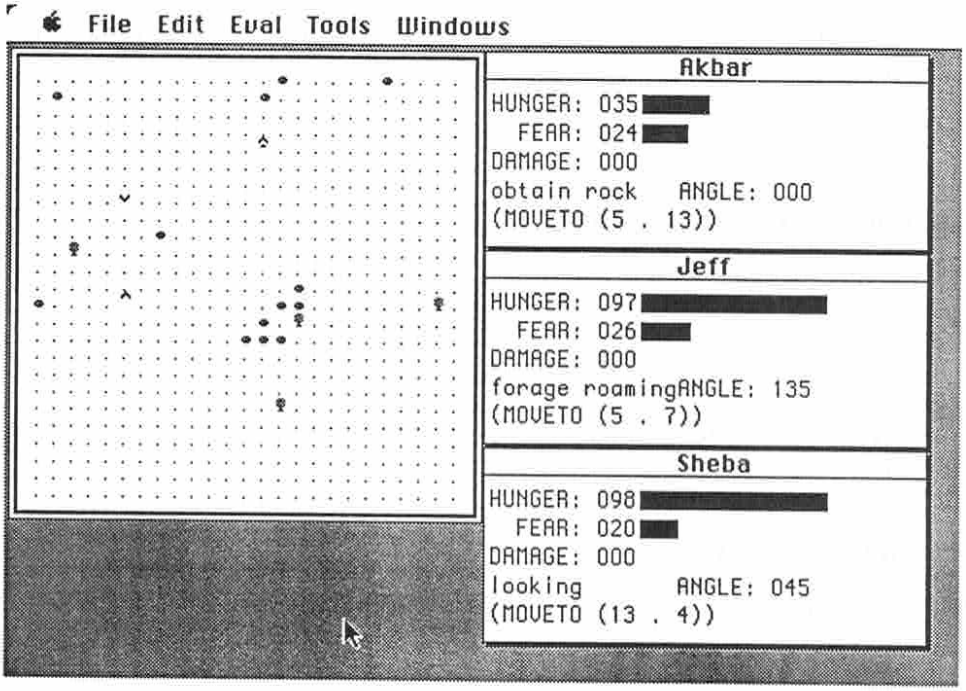

FIGURE 2 Petworld screen dump. Each pet has a window showing its current state, which part of its brain hierarchy is being utilized, and its next move. Pets, rocks, and trees are indicated in the world window with a distinguishing character set.

ANECDOTES

The screen dump in Figure 2 shows a sample pet simulation run. Three pets are shown in the example, all with identical brains and all with display windows. One may run a simulation with virtually unlimited numbers of pets, brains, rocks and trees, and the world can be made bigger. The practical limit seems to be about 10 pets and a world of about 50 by 50 elements. (The current graphic windows do not scroll, and the time taken for recalculation at that point would be on order of five seconds per round.)

I have run simulations for extended periods of time, finding that pets can be made that live for quite some time (thousands of rounds). Occasionally, pets exhibit unexpected or emergent behaviors.

When two pets build nests close to each other, for example, the nearest rocks to add to the nest are part of the other pet's nest, leading to many conflicts over rocks. These conflicts are usually stalemates, and get resolved when one pet gets hungry and heads off to forage.

Also, because the pet cannot see its nest when it is far away from it, the current nesting strategy causes pets to pick up extra rocks and stash them near the nest, a very advantageous strategy which arose out of the interplay of other behaviors.

SECTION TWO: KNOWLEDGE IN THE PETWORLD

Pets' brains contain strategic knowledge. In this sense, pets are knowledge-based systems. Unlike most other knowledge-based systems, the knowledge is organized in a rigid hierarchy of interacting agents called "experts." And unlike a decision tree, where the tree is used to score different moves, or a branching tree, where program control flows from top to bottom, Petworld utilizes a dataflow tree, where information is passed up from the bottom of the tree, being processed at each node in the hierarchy.

WHY A HIERARCHY?

Using a hierarchy to encode knowledge offers useful features:

Managing complexity: In a knowledge-based system, it is easy to end up with thousands of knowledge elements (ideas or rules) that must be chosen from. Some sort of structuring is needed—not only for efficiency in running the system, but also for clarity maintaining it. If we were to group ideas, they would share lots of information, and it makes sense to put them in an enclosure both to reduce the size of the knowledge file, and to indicate to the knowledge engineer the structuring of ideas. (In expert systems, these enclosures carry common preconditions for batches of rules.)

CONTROLLING EXECUTION: If we have grouped ideas together, it's easy to imagine some groups controlling other groups. For example, one group of rules could resolve conflicts between other groups. (This is the root of meta-rules as described by Randall Davis.[8]) It could resolve conflicts not only by having a built-in preference for one of the subordinates, but also by effecting a compromise between agents. If the animal is both scared and hungry, it might choose to run away in a direction that is favorable for finding food. Another possible resolution is called "displacement behavior": if two experts conflict, both are ignored and a third is employed. (Dogs, when unable to decide between running from an enemy and fighting, sometimes abruptly sit down and scratch an itch.)

INCREASING LEVERAGE: By taking the grouping strategy one step further and arranging experts into something resembling a hierarchy, a phenomenon similar to mechanical leverage arises: a little knowledge at the top of the hierarchy can have the same effect as attaching preconditions to many rules lower in the hierarchy.

WHAT KIND OF HIERARCHY?

The organization of the hierarchy of experts controls the flow of decisions. I examine several approaches to hierarchical structuring below.

1. *Decision trees*, from operations research, are similar to how many computer games work. A move generator feeds the hierarchy. Each expert passes to its superior a numeric rating for each move based upon world state, pet state, and the opinions of its subordinate experts. The action that gets the best numeric rating executes.

This approach has two drawbacks. It evaluates a lot to find out a little—an entire tree might be calculated for dozens of possible moves. It also requires that compromises be finagled as weightings of the scoring functions, rather than being written out as rules. So, some of the decision knowledge is hidden in the rating scheme, a circumstance I have chosen to avoid.

2. *Branching trees* are similar to the meta-rule and grouping approaches extended to their logical extreme: control branches down the hierarchy by running rules in each expert along the path until an expert at the bottom of the tree is chosen. One of its rules then fires and action is finally taken.

This approach, although theoretically as powerful as any other, in practice still has the reconciliation problem. To effect a compromise between two experts, a "compromise between a and b" expert will have to be constructed, and more rules will have to be added to the superior expert to decide when to choose it.

3. *Society of Mind*, a new theory by Marvin Minsky, seems the most powerful hierarchical structuring yet. Agents take control, and can invoke other agents

to assist them. Other agents also attempt to take control, and agent conflicts are an interesting part of behavior.

This approach has some difficult implementation problems, however. Conflict resolution between agents is currently an open question in SOM. For this reason, for simplicity, and because my research started as SOM was being finished, I opted for the simpler strategy below.

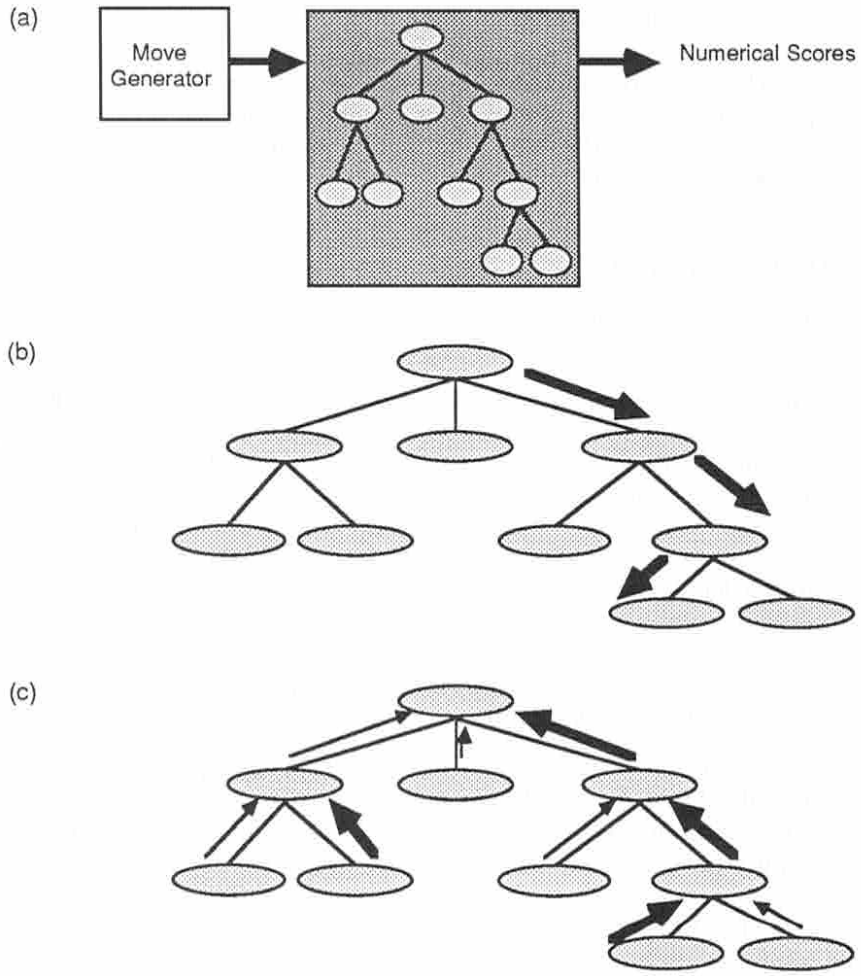

FIGURE 3 (a) Decision trees, (b) branching trees, and (c) dataflow trees.

4. *Dataflow trees*, the method I chose, has low-level experts "burble up" rankings from which higher levels choose actions. Each expert will be concerned with compromising between actions presented to it, either by chosing one action, or by creating a new action. Knowledge about making decisions is stored explicitly in the experts. The experts output only a relative ranking, not an absolute one.

LOCAL STATE IN PET BRAINS

One of my other design constraints was to build pets with as little local state as possible. Although originally undertaken as an exercise in seeing how much behavior could be obtained with how little a brain, I have since decided the decision was a good one since it echos the constraints of most of the animals I saw my system as capable of simulating.

One of the problems I have encountered is dealing with the concept of concentration: the ability to stay focused on one problem-solution path for some length of time. For example, because pets have limited vision, they must concentrate to find food they cannot see. Because experts do not have local state, and because the entire behavior network is reevaluated each time, food finding is impossible without some extra mechanism. To this end, I have incorporated a notion of recent history to allow concentration. Each expert can remember a history of its recommendations. In the case of the exploring expert above, the mechanism allows it to turn around precisely once, and then move several times in a row in a straight line. This history mechanism allows almost all of the power of Minsky's Society of Mind, with a simpler mechanism.

SECTION THREE: FUTURE WORK

Future work in Petworld will take place as my Master's thesis, including some of the topics below.

Since Petworld is part of Vivarium, issues of user interface, novice programming, and system complexity are crucial for any research to be used in a school setting. Several opportunities for user interface enhancements suggest themselves:

- A graphical dataflow programming language;
- A brain display showing the shape of the hierarchy and depicting the flow of information with animation and allowing for value display windows;
- Parameter controls that allow concepts such as "aggressiveness" to be added to brains; and
- A history recorder and forward-backward debugger, graphically similar to a video cassette recorder control panel, with a history line and markers for crucial moments.

The most recent addition to Petworld, the "recent history" mechanism, is currently incomplete and kludgy. I am designing cleaner functionality for a future Petworld.

LEARNING

But the most crucial missing feature is learning. Just about every living creature learns to some extent or another. Bacteria, for example, can learn to associate light level with temperature. Higher animals can remember locations and paths. Still higher level animals can make complex plans for action, and improve and radically change strategies.

As part of my Master's research, I have been developing a way to make pets learn somewhere on the third level mentioned above. Briefly, if there are many agents potentially employable for a given task, one could create managers to choose and remember agents that are more capable. Thus, managers can be expected to optimize behavior over time. If one then postulates an agent generator similar to those of Douglas Lenat,[12,13,14] pets could radically alter their behavior patterns. Research of this strategy is just beginning. (As a side note, one could then have pet brains reproduce sexually, since a lumping together of the two agent masses could be resolved by a simple mechanism. This does not happen in nature, but might be interesting to study.)

CONCLUSION

Petworld is a viable system for modeling some aspects of animal behavior. Pets can be created with simple, easily understandable brains that produce interesting, sometimes unexpected behavior. Although pets cannot reproduce or learn, they can undertake complex long-term actions because of the recent history mechanism. Work is underway to add learning to Petworld.

ACKNOWLEDGMENTS

Petworld 2.0 runs on Apple Macintosh computers with at least one megabyte of memory. It is preferable to use a machine with more memory and a faster 68020 processor, such as a Macintosh II. Petworld is written in Allegro Common Lisp. I wish to thank the people of Coral Software for their support. Inquiries about the code should be directed to the author.

Thanks go to Mario Bourgoin, Jim Davis, Marvin Minsky, Mike Travers, and the late Tom Trobaugh, all from the MIT Media Lab, for their crucial insights. Thanks also to the bizarre gang: BoB, Chris, Gypsy, Trrrisha, and even Terri.

REFERENCES

1. Agre, Phil (1985), "Routines," *MIT AI Laboratory Memo 828, May 1985.*
2. Alkon, Daniel L. (1983), "Learning in a Marine Snail," *Scientific American* **6/83**, 70–84.
3. Batali, John (1983), "Computation Introspection," *MIT AI Lab Memo No. 701, February 1983.*
4. Braitenberg, Valentino (1984), *Vehicles: Experiments in Synthetic Psychology* (Cambridge: Massachusetts Institute of Technology Press).
5. Camhi, Jeffrey M. (1982), "The Escape System of the Cockroach," *Scientific American* **12/82**, 158–172.
6. Chapman, David, and Philip E. Agre (1987), "Abstract Reasoning as Emergent from Concrete Activity," *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop* (Los Altos, CA: Morgan Kaufman).
7. Davis, James R. (1983), "Pesce," *a computer program modeling fish behavior*; a paper describing his system is in preparation.
8. Davis, Randall (1980), "MetaRules: Reasoning about Control," *MIT AI Lab Memo number 576, March 1980.*
9. Groelich, Horst (1986), *Vehicles* (Cambridge: MIT Press); a software package for Apple Macintosh.
10. Hofstadter, Douglas R. (1984), "The Copycat Project: An Experiment in Nondeterminism and Creative Analogies," *MIT AI Lab Memo No. 755, January 1984.*
11. Kehler, Thomas P., and Gregory D. Clemenson (1983), "KEE, The Knowledge Engineering Environment for Industry," *Intelligenetics, Inc., 1983; paper available from Intelligenetics, 124 University Ave, Palo Alto, CA.*
12. Lenat, Douglas B. (1975), "Beings: Knowledge as Interacting Experts," *Proceedings of the Fourth IJCAI, Tbilisi, U.S.S.R.*, 126–133.
13. Lenat, Douglas B., and Gregory Harris (1977), "Designing a Rule System that Searches for Scientific Discoveries," *CMU Department of Computer Science.*
14. Lenat, Douglas B., and John Seeley Brown (1984), "Why AM and EURISKO Appear to Work," *Artificial Intelligence* **23**, 269–294.
15. Lorenz, Konrad (1952), *King Solomon's Ring* (New York: Thomas Y. Crowell Company).
16. MacLaren, Lee S. (1978), *A Production System Architecture Based on Biological Examples, Ph.D. thesis, U. Washington Seattle, 1978* (available as University Microfilms Order No. 79–17604).
17. Maruichi, Uchiki, and Tokoro (1987), "Behavioral Simulation Based on Knowledge Objects," *Proceedings, ECOOP '87, Springer-Verlag Lecture Notes in Computer Science No. 276* (New York: springer-Verlag).
18. Minsky, Marvin (1986), *The Society of Mind* (New York: Simon and Schuster).

19. Scientific American, eds. (1979), *Brain: A Scientific American Book* (New York: W. H. Freeman Company).
20. Stefik, Mark, D. G. Bobrow, S. Mittal and L. Conway (1983), "Knowledge Programming in Loops: Report on an Experimental Course," *AI Magazine* **Fall 1983**, 3–13.
21. Travers, Michael (1988), *Animal Construction Kits*, these proceedings.