

# MIKROKOSMS<sup>1</sup> AND ROBOTS<sup>2,3</sup>

by

Leonard Uhr and Manfred Kochen  
U. of Wisconsin U. of Michigan  
U. S. A.

## Abstract

This paper presents a computer program that simulates situations (called "MIKROKOSMS") in which several entities (called "organisms") wander around in an environment that includes one another plus other, apparently simpler, entities (called "objects"). The program has a rather simple set of "laws" of objects, which can be thought of as the laws of physics of the MIKROKOSM. It also has other specifications for organisms - their input and output functions (called "perception" and "response"), their reward functions (called "motivation" or "needs"), and their mechanisms for building up internal memories (called "learning" and "hypothesis formation") that will help them to recognize objects in the future, and respond appropriately to them (for example, in order to maximize expected rewards). This program lays bare the processes that are needed to handle interactions among simulated organisms and objects, including the learning of hypotheses that will guide future action. The present program has only the simplest of pattern recognition, hypothesis formation and need-satisfaction capabilities. Its purpose is to make clear and concrete how such things can be interrelated in a complete system. Descriptors: Robots, mikrokosms, computers, integrative systems, pattern recognition, learning, hypothesis-formation.

## Background

Several relatively simple and special-purpose precursors to MIKROKOSMS have been reported in the literature. Toda<sup>16</sup> discussed the decision-making problem of organisms that move at some cost in energy in order to get energy-giving "mushrooms." Doran<sup>12</sup> also examines very simple simulated organisms moving through a space. Travis' simulation and discussion of problem-solving in the form of a chess knight moving in order to cover a board can be thought of from this point of view<sup>17</sup>. Indeed, all game-playing and theorem-proving programs can be thought of as simulating the pieces of the game or the expressions of the logistic system as they move about the environment of the gameboard or the proof tree. (For examples, think of Samuel's checker player<sup>14</sup>, Greenblatt's chess player<sup>4</sup>, Zobrist's GO program<sup>20</sup>, Gelernter's geometry theorem prover<sup>3</sup>, Newell and Simon's logic theorist<sup>9</sup>. These programs become more pertinent when they attempt to be general over several games, for example, Newell and Ernst's General Problem Solver<sup>8</sup>, Pitrat's<sup>12</sup> and Newman and Uhr's<sup>10</sup> to-some-extent-general game players.)

But none of the above seem to be exploring the issues for which the MIKROKOSM programs were developed (Kochen and Uhr<sup>5</sup>). They do not try to tear apart what we mean by environments, objects, organisms, hypotheses, learning, and so on. They do not try to vary, and to generalize about, these things. Rather, they look at relatively peculiar and rigid spaces (a checker board, a logistic system), and their "organisms" (which contain little in the way of perceptual systems, memory, or learning abilities) are rather ad hoc to their space (e.g., chess pieces, logical expressions).

During the past few years four large "robot" projects have been developed (Pingle et al<sup>11</sup>; Minsky & Papert<sup>6</sup>; Raphael<sup>13</sup>; Sutro & Kilmer<sup>15</sup>) and there turn out to be certain similarities between robots and MIKROKOSMS. Robots are real physical objects that wander around in our real physical world. Movies can be made of them; when they bump into walls or people they can leave scars. The objects in a MIKROKOSM are computer simulations that consist of internal representations of numbers and letters, of bits, or of magnetic or electrical impulses - whichever you prefer. They can indeed be photographed if a program monitors them onto a scope; and still another equally trivial program could monitor them by having plastic and metal physical objects moving around through a room, just as the scope monitor has grades of light moving around over a phosphorescent screen.

The differences of "reality," of physical "hardness" and ability to bump and scar, are differences that we had best leave to the ontologists. But there are important differences that we can understand. MIKROKOSMS force us to understand, or at least to code, our environments and their interfaces with our organisms (which simulate robots); whereas the robot researchers simply stick their real robots into the real world, and thus

<sup>1</sup> For people partial to acronyms, MIKROKOSM might stand for: "Models of Inductive Knowledge in Responding Organisms Konstruced (from CHAOS by Concept-formation, Heuristics and Adaptation to Organization Sensed) from Only Sensation, Memory, and Sweat.

<sup>2</sup> Robots on their way to the moon might be thought of as ReaL Orbiting Bits of Ostensible Things.

<sup>3</sup> This research was partially supported by NIH Grant, 12977.

(potentially) get all the advantages of real-world complexity into their environments. The cost they pay is the cost and the trouble of building robots (which is something that has turned out to take several years, several people and several hundreds of thousands of dollars). This cost will go down and disappear. They may also pay the more important costs of reduced complexity of the organisms they can handle, and rigid behavior on the part of these organisms. Or at least this seems to be the case to date. Finally, they have not yet actually taken advantage of the complexity of the real world into which they can put their robots. We will return to these issues in the discussion at the end of this paper.

### MIKROKOSMS Described

We will now give a brief description of MIKROKOSMS in the following way: 1) An actual running computer program (called "MIKRO-1") will be discussed. This program is presented in the Appendix, along with an English-language "Precis" that gives a detailed statement-by-statement description of its processes and its flow of control. 2) Comments will be made about the general function embodied in each section of the program, with special emphasis on how it might be simplified to a bare-bones minimum. 3) We will discuss and suggest ways in which the function might be made more powerful.

The MIKROKOSM programs, at least as we conceive of them at present, break down into the following sections:

- I. Initialize parameters.
- II. Generate the initial state of the MIKROKOSM.
  - A. Generate objects, and put them into the space.
  - B. Generate organisms, and put them into the space.
- III. Compute environment-organism interaction (for each organism).
  - A. Print out the present state of the MIKROKOSM.
  - B. Present its updated view to the organism.
  - C. Let the organism process this view:
    1. Recognize objects.
    2. Conjecture pertinent hypotheses.
    3. Choose the most highly valued hypothesis.
- IV. Update the MIKROKOSM to advance to the next time interval.
  - A. Compute and effect physical law changes.
  - B. Compute and effect changes resulting from organisms' acts.
  - C. Have the organism learn as a function of feedback from eating
  - D. Go to step III, advancing time by 1 unit.

### Initialization of Parameters

The program must first be given a picture of the kind of mikrokosm space it should generate. This includes the dimensions of the space, the number of organisms and of objects in the space,

and the characteristics of the organisms and the objects. This might be done in a very general way, by parametrizing all of these characteristics and having the program choose random values for the parameters. Or it might be done with precise descriptions of all the pertinent details.

MIKRO-1 is a simple mixture of easily parametrized and built-in characteristics. The basic parameters (PARAMS) for a space are taken to be its row and column size (all spaces are 2-dimensional), the number of organisms and the number of objects in the total space, and the distance that each organism can see. The program is given a set of standard values for these parameters (7 rows, 20 columns, 3 organisms each seeing 5 columns in either direction, and 4 objects). But the program will attempt to read a data card at run time, on which new parameter values can be given to it.

The objects that will be placed into the space will be exact copies of the general object types (GENOBSJS). These are descriptions of the set of points, (that is, the shape) of the objects, and of the physical qualities (such things as color, breakability, motion, weight, and caloric value).

Whereas information about the space is parametrized, and information about objects is stored in tables, information about organisms is embodied in code. [This probably reflects the relative simplicity of the space, and the relative complexity of organisms. But it probably also reflects the primitive stage of this program: a better program would have better reasons for using parameters, tables, or code.] Essentially, an organism is some calls to subroutines that a) take a look, b) pattern recognize, c) test hypotheses, d) choose acts, and e) generate and learn new hypotheses.

### Generation of the Initial State of the MIKROKOSM

The program uses the parameters that define the space to compute the average distance between objects and between organisms, and then generates and places the specified number of each. MIKRO-1 makes several simplifications to shorten the code and speed up processing time. Some of these lead to peculiar characteristics, but all could be coded properly - in most cases with very little trouble.

Objects. Objects are generated first, by making one specific object from each general object, in turn, until the specified total number of objects has been made. To avoid any need to check whether several objects are put into the same place (though in fact in MIKRO-1 space this will happen later, causing peculiarities but no fatal problems), all objects are put into the top row of the space, at distances computed so that the specified total number of objects will be equally distributed. Initially, all specific objects of each general type will be identical, except for position in the space. But as time passes objects may move, bounce, break, and in other ways change in value.

Organisms. MIKRO-1 now generates the specified number of organisms and distributes them uniformly along the bottom row of the space. An organism is simply a name that contains a set of lists with that organism's qualities (energy level, location, direction and force of motion), its hypotheses (after it has learned some), its present view of the space, and some memory as to its own acts and changes it has noted in its environment.

Organisms resemble objects in that they both have physical characteristics; but they differ in that only organisms store and learn hypotheses, view the environment, and have some memory. At present these differences reflect our own primitive thinking on the subject; it will be of great interest to see what differences remain when we have tried to force as much similarity as possible.

There are several major peculiarities to the space of objects and organisms generated up to this point. There is insufficient space to describe, discuss, and justify them all in detail, but briefly: Although objects have shapes in 2-dimensions, they are stored as though at points. This avoids the necessity of detecting and computing overlap between objects, which would either force us into 3-dimensional space (which would be expensive of memory and processing time) or a peculiar physics of objects occupying the same point at the same time. The proper way to handle the space would be for all objects to reside, properly positioned, in a background of emptiness or noise (chaos). But this would introduce a number of processing steps that would make this aspect of the program far more sophisticated than needed for the low level of sophistication of the rest. So we store objects and organisms on lists, and designate their positions explicitly by the coordinate numbers, rather than implicitly by their actual positions in the framework of the space.

#### Computation of Environment-Organism Interactions

Printout. The program outputs the present state of the mikrokosm before it begins the set of object-organisms interactions that will lead to change, and before each subsequent time period. This printout draws a picture of the current space, with its organisms and objects, and also exhibits the present state and innards of all objects and organisms.

Organism Views its Environment. That part of the total space that lies within its field of view is presented to each organism, in turn. Only the externally-visible shape of objects (including the viewing organism) can be seen, and the position of each object is computed and given to the organism relative to its own position. Changes in these relative positions are also detected and given to the organism.

Pattern Recognition and Hypothesis-Testing. The organism now begins to apply its set of hy-

potheses to its seen view. [Specific hypotheses might be coded in advance into each organism's hypotheses table, but the alternative procedure that we prefer is to rely upon the organism's learning abilities to form these hypotheses through experiences gained in interaction with the environment.] For each object in its view, the organism checks to see if there is an hypothesis in its memory with that object as one of its premises. It is here that the program does a very primitive kind of "pattern recognition," in which the description of the object must match a template representation stored in the organism's memory.

The interesting thing about this process is the structure of an hypothesis, even in its overly-simple present form. An hypothesis is a statement about what the organism should do, and subsequently expect, when a certain thing(s) is recognized in the environment and the organism is in a certain state(s). At present the thing recognized is one of our simple and undeviating objects, recognition is effected by a perfect template, the acts are simple built-in sequences of code, and the expectations are of a certain amount of positive or negative change ("pleasure" or "pain"). But we feel justified in suggesting that it is fine to keep pattern recognition as simple as possible, when we have so many other problems in the total system, and we know a great deal about how to make more sophisticated pattern recognizers. An especially appropriate type of pattern recognizer will be the sort of flexible heirarchical compound learning program discussed in Uhr<sup>19</sup>. We should also note that the recognition of more complex compounds of varying internal states within the organism can be handled in exactly the same way as external recognition, for these are merely two (sets of) premises of an hypothesis. The issue of "planning" is captured in the setting up and learning of sequences of acts for the organism to do, which form the consequences of an hypothesis, and also sequences of hypotheses that the program might act upon over time. It is in the individual hypothesis and the methods for choosing among hypotheses, that much of what we call "pattern recognition," "planning," and "problem-solving" lie. MIKRO-1 handles these problems in the grossest and most primitive ways, but it also makes painfully and hopefully clear how our mechanisms for hypothesis-application and hypothesis-formation can be improved.

An hypothesis is conjectured to apply because something in the environment was found to be one of its premises, and hence implied it, and then the organism found that its own present state satisfied the internal state premised by the hypothesis, and finally computed a positive expected value of this hypothesis (which is a function of expected return from the object involved, the object's distance: a rough measure of difficulty of applying the hypothesis, and the weight of assurance with which this hypothesis is held). After the program has found all such pertinent hypotheses, it chooses the one with the highest positive weight. [A better program would

allow an organism to do more than one thing at a time, to decide what to do before it had spent all the time needed to examine everything, and to make decisions as a function of sets of mutually complementary hypotheses. A simpler program might ignore internal states, and merely choose a single act as a function of a single recognized object. Such a program would be doing straightforward pattern recognition. The program might merely have one act, rather than sequences of acts, stored as the consequence of each hypothesis. It would then be incapable of building up coherent strategies or plans. It is at this point, when a program is given the ability to build up sequences and compounds of characterizers of environments and of internal states, and of acts and expectations, that it begins to build up the capability of doing interesting pattern recognition, concept formation, serial prediction, problem-solving, and planning.]

#### Up-dating the MIKROKOSM to Advance to the Next Time Interval

After all organisms have been given the chance to interact with their perceived environments, the UPDATE routines change the MIKROKOSM, as indicated by two lists: a) the physical qualities of all the organisms and objects and b) the UPDATE list of the organisms' chosen acts.

Physical Changes. The only physical quality that is examined by MIKRO-1 is the object's motion. The object is moved as indicated and, if it has hit a border of the space, is kept at that border. [A better program would allow objects to accelerate in their motion, and to change direction, for example after hitting one another. The program should also compute other changes in objects, for example decay and cracking over time, and changes that are a function of interactions between objects, for example breaking or fading.]

Organisms' Acts. The acts that the organisms chose to put into UPDATE are now carried out. An act is a sequence of one or more things to do, and each such thing names a subroutine that does it. At present, the program has subroutines that allow the organism to: 1) GET (move 1 step toward the specified object), 2) DESTROY (wipe out the specified object if it is at the same location as the organism), 3) FLAIL (move 1 step in a randomly chosen direction), and 4) EAT (wipe out the specified object, and also add the object's specified caloric content to its own internal energy level). This is a rather arbitrary set of primitive acts, chosen primarily to get MIKROs started doing something. But note that some are binary (DESTROY, EAT) and some unary (FLAIL, GET), and that strings of acts can be compounded together.

Learning. EAT is of special interest because only here is there real feedback to the organism,

when its energy level is changed as a function of the object's caloric value (which can be negative, denoting a noxious object). It is therefore here, and here only, that the organisms in MIKRO-1 "learn." [A more sophisticated program would have learning occur from a wider variety of feedback information - for example, from noting that objects are getting closer, or that a certain sequence of acts (as stored in the organism's running memory of past-done things) resulted in desirable or undesirable consequences. In MIKRO-1 there is only one overall "energy level," which is changed when an object is eaten and its "caloric content" adds to or subtracts from that level. There might be several components of proper functioning - eg. protein, fat, water, oxygen, touching - and there might now be a need to develop characterizers about the self's patterning of these components, and changes in this patterning.]

When the organism's energy level is changed as a function of eating an object, the program hunts through the past done acts of that organism to find an hypothesis that led to acts with the eaten object as their object. Two types of hypothesis are stored on the organism's past-done list: the single hypothesis chosen at each time interval, and the rest of the hypotheses conjectured. If no chosen hypothesis turns out to be about the object eaten, the organism goes through its conjectured hypotheses and, if one is found about that object, raises the value of that hypothesis, so that, since it is pertinent, it will more likely be chosen the next time. [This is a rather arbitrary thing to do, and should not be taken very seriously; it merely demonstrates how easy it is to program in variant types of learning.]

The first pertinent hypothesis that is found is then up-weighted or down-weighted, depending upon whether the object eaten was positive or negative (noxious) in caloric content. [A better program might change the weight as a function of the actual caloric value, as well as its sign.] If down-weighted, the new weight is examined to see if it has fallen below some acceptable minimum. If it has, the hypothesis is discarded. [MIKRO-1 arbitrarily sets the initial weight of an hypothesis at 5, and discards the hypothesis when its weight goes below 1. A more sophisticated program might keep a better record of the good and bad consequences of an hypothesis, and run more sensitive statistical tests to see whether the hypothesis has proved itself good or bad over a sufficiently large sample.]

MIKRO-1 generates a new hypothesis a) when an hypothesis is discarded and b) after each time period. [It would be better to keep some count of the amount of space available for the organism's hypotheses - its memory size - and then eliminate the least valued hypotheses when there was not enough room for everything. Alternately, the

organism's level of functioning and improvement rate on this level might be stored, with hypotheses being discarded and generated as deemed best to improve further. A number of extremely interesting aspects of learning pop up at this point.]

The variety of new hypotheses MIKRO-1 can generate is limited. If the caloric content of the eaten object was positive, the consequent acts of the hypothesis will be "get, eat;" if negative, they will be "get, destroy." The object's description is stored as one premise, so that the simplest kind of whole-template pattern recognition is learned and used. The initial expectation and weight of all hypotheses are set at the same level-90 and 5. The program checks whether a newly-generated hypothesis has already been discarded in the past; if it has, it doesn't bother to generate it again. [This is a subtle issue; if the mikrokosm were noisy or changed over time, organisms should be able to try hypotheses again. On the other hand, if there is a large set of possible hypotheses, learning might be slowed dangerously if the organism tried again hypotheses it had already found worthless.]

#### Printout

The program listing ends with the code that prints out the state of the mikrokosm after each time period. This printout is cumbersome, and need not be described in detail. Essentially, the actual 2-dimensional representation of the mikrokosm is printed as a matrix, all organisms and objects that still exist are listed, and the contents (the internal states) of these organisms and objects are given.

#### Discussion

MIKRO-1 is already too complex to lay bare the minimal structure of a mikrokosm, yet it is far too simple to be very convincing or interesting. Our intent is that this is the first in a continuing series of more complex and more sophisticated programs. These will look into each of the aspects of a mikrokosm separately (pattern recognition, hypothesis formation, hierarchical compounding, planning, inductive learning, discovery, and so on). It should also be instructive to simplify, to try to get at the essence of, mikrokosms by: a) getting at their bare bones, and b) generalizing, eliminating as much as possible of the built-in.

#### Motivation

There are a number of reasons why MIKROKOSMs seem interesting things to worry about and to simulate.

A) They force those of us interested in modeling intelligent processes (whether "artificial" and/or "natural") to take into account all aspects of our problem, and to contend with the central

issue of integrating into a visible whole the various functions that we typically study separately. On the one hand many of us (including the authors) argue that our problem is far too big, so that we should simplify as much as possible; and that, further, not until we have far larger computers will our models exhibit diverse, flexible, and interesting behavior, which we feel may to a great extent be a function of size of memory. So we separately study "pattern recognition" or "concept formation" or "verbal learning" or "problem solving" (e.g., "game playing," "theorem proving," "serial prediction") or "decision functions" or "responding" (moving a "hand" or an "arm" or a "robot"). This is, we hope, right and proper. But we should also begin to combine these various functions. They are merely pieces of a total program, and we had better start worrying about whether we can ever get those pieces working efficiently together.

B) By putting the different functions together at the precise level of computer code, we gain the opportunity to generalize across them. We should force ourselves to do so. One gets the impression from papers and talks about some of the "robot" projects that they intend to take a pattern recognition program, and a question-answering program, and a concept formation program, and a theorem proving program, and maybe a few other programs, and put them together into a Frankenstein monster. On the contrary, we should use this opportunity to tear apart and try to understand our code and the functions we are trying to compute, so that general-purpose routines are achieved. For example, almost certainly most if not all of "pattern recognition" and "concept formation" should be performed by a single subroutine. Those aspects of pattern recognition that involve deductive inference should have subroutines in common with problem-solving. And so on. We must confront ourselves with general criteria for theory-building: keep the set of constructs as simple, elegant, non-redundant, powerful, and insight-producing as possible.

C) The MIKROKOSM situation raises some extremely interesting new questions of generality. For example, when we work with typical pattern recognition or artificial intelligence programs the patterns or other external objects with which they interact are merely presented to them. But now we must describe objects in the same kind of computer code with which we describe our perceiving, problem-solving organisms. We can now try to describe both objects and organisms with the same subroutines and the same tables, and we can begin to ask how they are similar, and how they differ. This focuses us on the fascinating issue of what turns an object into an organism. It prepares us to wonder about how a self-organizing system of objects might begin to evolve some organisms. It forces us to think about what must minimally be given to a MIKROKOSM for it to contain, or evolve, objects

among which will be organisms.

D) We can hold one or more parts of the total MIKROKOSM constant or trivially simple, vary the rest, and examine the resulting behavior.

E) We can ask a number of questions about learning that cannot be so clearly asked in any other situation. We can ask what is a minimal organism that will learn; what kinds of environmental experiences must this organism undergo; what kinds of information must this organism store. Since we have close control over both environment and organism (at least until they start to interact and to learn), we can constantly think about one with respect to the other, and we can continually try to simplify both.

F) We can ask two closely related questions that have simply not been asked before at the precise level of computer modelling: 1) How do two or more organisms learn to talk, including the development of vocabulary, grammar, and semantic reference, and the development of the mutually understood convention that these things should be developed, and should have common meaning to all organisms belonging to that linguistic community? 2) How do two or more organisms come to compete, and to cooperate; and what is the relation of such social behavior to a) their basic needs and ways in which they can be satisfied on the one hand, and b) their development of language on the other hand?

G) We can try to simplify to the point where we may pinpoint what is absolutely essential. For example, it is not clear whether such a MIKROKOSM must have a motivating force in the form of the organisms' internal needs. We keep being forced back to such a beginning, or some close equivalent, such as a diffuse curiosity or a neural itch, despite the fact that this somehow, in a very vague way, feels ad hoc and intuitively unsatisfying.

#### MIKROKOSMS and Robots

The Stanford University robot project (Pingle et al,<sup>11</sup>) has developed a robot that (at least so far) can do the following: if several large wooden blocks are scattered on a table, a robot hand with several fingers will zero in to hover above each block, close its fingers around that block, pick it up, and put it on top of a tower of like-sized blocks it is building. It thus perceptually sorts out the blocks by size, positions itself to pick up a block, and then places the block very carefully, so that it sits square enough on top of its tower so that the tower does not too often come tumbling down. This involves some extremely complex matters of precise positioning of fingers around blocks, and of blocks on top of other blocks. These problems may well be horrendous, if not impossible; but unfortunately this may be the case because they are being handled with several built-in

strikes against the robot. For it gets no feedback about, and has no ability to adapt to, slight variations in position; it is in the same unfortunate position as a guidance system that must compute its trajectory to hit the moon without any opportunity for subsequent self-correction from feedback as to its deviations.

The Stanford Research Institute robot (Rosen, Nillson, Raphael; see Raphael<sup>13</sup>) can wander through a room that has irregularly shaped objects placed in it, and learn through experiences of bumping into objects how to avoid them, and finally wander through the room without bumping into these objects. But its methods appear to be rather rigid and ad hoc - essentially, it stores a graph-paper representation of the room, and it fills in squares as "containing object, so avoid" whenever either its TV camera "eye" or its bumpers find a block covering the corresponding section of the room.

Both of these robots do things that MIKROKOSM organisms do not do, and they force the researcher to contend with problems that MIKROKOSMs avoid. It is up to the individual researcher's interests and tastes which he decides to be the more important problems, or the problems most central to modelling of intelligence. Robot projects must contend with the noises introduced by TV camera inputs and by mechanical contraptions that can't move around without joggling their TV eyes, stop suddenly without randomly overstepping their intended stopping-point, or place one finger on an object without moving that object slightly so that the computed position for the next finger is no longer correct. New and interesting pattern recognition problems are confronted when the images of possibly interposed 3-dimensional objects on a table or in a room, with their shadows and gradations of intensity, must be recognized. These are problems the real world environment forces upon one, and it is good to be confronted by the real world.

A MIKROKOSM simulation could be expanded to handle 3-dimensional partially-viewed, shaded patterns, and these are large problems of pattern recognition on which research should be done. But most pattern recognition researchers would, we think, agree that such research should be done with computer programs and special-purpose pattern recognition computers, not with robots. In fact the actual pattern recognition programs incorporated into these robots are at a very low level of sophistication compared with existing pattern recognition research (see, e.g., Uhr<sup>19</sup>, Nagy<sup>7</sup>).

It would be more difficult to simulate problems of bouncing objects and falling towers; but it is hard to judge their centrality.

The above is intended to suggest that robot researchers may be tackling difficult problems that are not really their central problems - the mechan-

ics of putting a robot together; the issues of illumination, contrast, color, and noisiness of patterns input; and the mechanical problems of exact motion and placement - and are simplifying the artificial intelligence aspects of their robots to the point where they are actually less sophisticated than existing computer simulations, including functions, that have been put into our simple MIKROKOSM organisms. But it is not at all intended to criticize the intent of robot research, which we take to be to confront artificial intelligences with the enormous and, intriguingly, inexhaustible and infinite complexity of the real world.

On the contrary, it seems to us crucial to at some point put our simulated organisms into the real world. The complexity of the real world may be necessary for adequate learning; there may even be something to the infinite variety of the real world that is fundamentally unprogrammable, and we will never know and benefit from this until our learning programs are hooked into the real world. (We assume that good artificial intelligences will be learning programs - they must be able to adapt, and they will be far too complex for anyone to succeed in formulating, or programming.)

But there is a lot to discover about our organisms before they are ready to be put into the real world. People who don't have the mechanical bent, skills, or money to build robots can still play an important role in this aspect of robot research. The interfaces between our organisms and the real world almost certainly should be richer than those developed so far in the robot projects. For example, the lack of ability to continually monitor feedbacks and modify behavior accordingly puts existing robots at a tremendous disadvantage. And the "real world" has to be an interesting part of the "real world." To what extent is a robot confronting the real world when the room it is in contains two sizes of blocks, sufficiently illuminated so that they can be perfectly resolved through the input device used? This is merely a world of blocks and positions, even simpler than the simulated worlds of MIKROKOSMs.

#### Summary

This paper describes and discusses a simple program, "MIKRO-1," that simulates the interaction between a set of "objects," some of which are "organisms," wandering around within a little "mikrokosm" space. This is the first of what we hope will be a series of programs to explore complex intellectual processes by combining the various functions (pattern recognition, concept formation, problem solving, decision making, remembering, learning, and hypothesis formation) that have typically been studied separately.

The basic structure of a mikrokosm appears to be the following: it must include a set of objects,

some of which are organisms, that interact within some space. There must, therefore, be ways of describing or generating objects and organisms, and placing them in interrelations. Both objects and organisms must have qualities that are interpreted according to "physical laws" of the mikrokosm; among these will be n-ary qualities that are functions of interactions (e.g., "breaking," "eating"). Organisms must be able to generate acts as a function of interactions. In particular, sensory interaction (seeing at a distance) and hypothesizing and learning (generating and storing sets of premises as to the states of the external environment and of the internal characteristics of the organism) seem necessary for any interesting variety of organism behavior. The mikrokosm must be able to collate all the changes of organisms and objects, and up-date itself to the next time period.

MIKRO-1 contains a special kind of object, a "shout," that is located everywhere. This we take to be the basis of future language learning. But MIKRO-1 makes no use of shouts. Nor does it have a very sophisticated set of basic acts (eat, get, flail, destroy), reasons for learning (the positive or negative caloric content of an eaten object), or pattern recognition ability (whole-template matching). It has rigid methods for forming hypotheses, and it has no ability to put anything - hypotheses, pattern characterizers, or acts - into interesting compounds or hierarchies. But mikrokosms make painfully apparent the need for such improvements, and we think that they are an especially good test-bed within which to construct and examine more sophisticated processors.

#### APPENDIX

##### \*ABSTRACT FOR MIKRO-1.

- A Get PARAMS: ROWS, COLS, NORGS, VIEWORG, NOBJS, OBJDESCRIPTORS.  
GENERATE SPECIFIED NUMBER OF OBJECTS,  
SPREAD ACROSS ROW 1.  
GENERATE SPECIFIED NUMBER OF ORGANISMS,  
SPREAD ACROSS ROW N.
- B PRINT OUT THE MIKRO, CHANGES, AND THE ORGANISM'S INTERNAL STATES.
- C PRESENT THE NEXT ORGANISM WITH THE VIEW OF WHAT IT CAN SENSE, BUILDING UP A CHANGES LIST FROM LAST TIME.

WHEN NO MORE ORGANISMS, GO TO D.  
THIS ORGANISM FINDS ALL HYPOTHESES IMPLIED BY WHAT'S IN VIEW, COMPUTES A VALUE FOR EACH HYPOTHESIS, CHOOSES THE SINGLE MOST HIGHLY VALUED HYPOTHESIS, AND PLACES IT IN UPDATE AND IN ITS OWN PASTDO LIST.

GO TO C.

- D UPDATE THE MIKRO FOR THE NEXT TIME PERIOD:  
MOVE EACH OBJECT (INCLUDING THE ORGANISMS) AS SPECIFIED IN THEIR DXY (CHANGE\* OF-LOCATION) VALUE.

|                                                                                                                                                                            |                                   | <u>Statement</u>                                                                                                                                                                                                                                                                                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                            |                                   | <u>Number</u>                                                                                                                                                                                                                                                                                                                |
| FOR EACH ORGANISM, DO THE STRING OF ACTS SPECIFIED: (INCLUDE MOVING - FLAILING, GETTING - , EATING, DESTROYING.)                                                           |                                   | No - Let I2 equal 1. (puts all objects in row 1).                                                                                                                                                                                                                                                                            |
| REWEIGHT THE ACTED-UPON HYPOTHESIS UP OR DOWN.                                                                                                                             |                                   | From GENOBS, get the first TyPe of object and its DEScription, and put it after the REST, at the end of GENOBS (so will generate one example of each type of object in turn).                                                                                                                                                |
| IF WEIGHTED DOWN, TEST WHETHER IT SHOULD BE THROWN OUT.                                                                                                                    |                                   | Add 1 to the Number-of-Objects-generated (NO).                                                                                                                                                                                                                                                                               |
| IF YES, PUT IT ON DISCARDS LIST, AND TRY TO GENERATE A NEW HYPOTHESIS FOR THIS ORGANISM.                                                                                   |                                   | Let the column where the next object will be PUT equal the present PUT (which is 0 initially) plus DISTO (the DISTance-between-Objects)                                                                                                                                                                                      |
| WHEN ALL ORGANISMS ARE DONE, GO TO B.                                                                                                                                      |                                   |                                                                                                                                                                                                                                                                                                                              |
| END GO TO A.                                                                                                                                                               |                                   |                                                                                                                                                                                                                                                                                                                              |
| *PRECIS MIKRO-1. SIMPLEST 2-DIM. ENVIRON. OBJS PUT IN ROW 1.                                                                                                               | <u>Statement</u><br><u>Number</u> |                                                                                                                                                                                                                                                                                                                              |
| **INITIALIZE THE PROGRAM'S PARAMETERS.                                                                                                                                     |                                   |                                                                                                                                                                                                                                                                                                                              |
| GO                                                                                                                                                                         | 0                                 | *NOTE THAT HERE AND ELSEWHERE THE RANDOM NUMBERS CAN BE USED TO VARY POSITIONS.                                                                                                                                                                                                                                              |
| Let GENERateOBJects (GENOBS) contain a list of the prototype objects and their descriptions.                                                                               |                                   | Add to OBJS (the list of all OBJECTS-generated) this object's name (OB1, OB2, ...OB(NO)) followed by its TyPe, DEScription, and row (I2) and column (PUT) locations.                                                                                                                                                         |
| Let each prototype object contain a list of its physical qualities.                                                                                                        | 1-5                               | Let this object's name (OB(NO)) contain its LOCation, (I2*PUT) and a description of its physical qualities (stored in TyPe). Go to B6, to generate the next object.                                                                                                                                                          |
| DEFINE the function BORD (which keeps objects within borders).                                                                                                             | 5                                 | **PLACE ORGANISMS.                                                                                                                                                                                                                                                                                                           |
| DEFINE the function PUTOUT (which outputs information about the state of the mikrokosm after each time period).                                                            | 6                                 | B1                                                                                                                                                                                                                                                                                                                           |
| Let PARAMeterS contain a standard set of parameters.                                                                                                                       |                                   | Let DISTance-between-Organisms (DISTO) equal the number-of-CoLumnS divided by the Number-of-ORGanisms.                                                                                                                                                                                                                       |
| READ in a different set of parameters for this run (if given)                                                                                                              | 7                                 | Let the column where the first organism will be PUT equal $\frac{1}{2}$ DISTO.                                                                                                                                                                                                                                               |
| Get from PARAMS the individual parameters: RWS (RoWS of space); CLS (CoLumnS of space); NORG (Number of ORGanisms); OSEE (Organism's-SEEn-view); NOBJ (Number of OBJects). | 8                                 | If PUT equals 0, set it equal to 1 (so 1st organism will be inside).                                                                                                                                                                                                                                                         |
|                                                                                                                                                                            | 9                                 | B8                                                                                                                                                                                                                                                                                                                           |
| Let RandomBACKground (RBACK) contain a random list of symbols.                                                                                                             | 10                                | Add 1 to NO.                                                                                                                                                                                                                                                                                                                 |
| Let RANDOM contain a random list of 1, 0, -1.                                                                                                                              | 11                                | Add to ORGS (the list of all ORGanisms-generated) this organism's name (ORG(NO)), where NO are the integers that follow the integers used for objects, the number of this organism (NORG), this organism's name again, and row (RWS, the last row) and column (PUT) locations. (== indicates organism; != indicates object.) |
| Let SHOUT contain the physical qualities of shouts.                                                                                                                        | 12                                | Let this organism's name (ORG(NO)) contain its attributes (ENergy, initial location (RWS*PUT), and an initial description of its physical qualities and its past memory (initially blank)).                                                                                                                                  |
| Let OBJECTTYPES contain the names of the prototype objects.                                                                                                                | 13                                | Subtract 1 from the Number-of-ORGanisms (NORG)-to-be-generated.                                                                                                                                                                                                                                                              |
| Let PRIMDO contain the primitive acts an organism can do before learning.                                                                                                  | 14                                | Is NORG LessThan 1? Yes - Go to B9.                                                                                                                                                                                                                                                                                          |
| **BEGIN TO GENERATE OBJECTS.                                                                                                                                               |                                   | No - Add the DISTance-between-Organisms (DISTO) to PUT, for placing the next organism. Go to B8.                                                                                                                                                                                                                             |
| BEGIN                                                                                                                                                                      | 15                                | **PRESENT ITS VIEW TO EACH ORGANISM.                                                                                                                                                                                                                                                                                         |
| Let the average DISTance-between-Objects (DISTO) equal the number of CoLumnS (CLS) divided by the Number-of-OBJects.                                                       |                                   |                                                                                                                                                                                                                                                                                                                              |
| B6                                                                                                                                                                         |                                   |                                                                                                                                                                                                                                                                                                                              |
| Is the Number-of-Objects-generated (NO) GreaterThan the Number-of-OBJects-to-be-generated (NOBJ)?                                                                          | 16                                |                                                                                                                                                                                                                                                                                                                              |
| Yes - Go to B1.                                                                                                                                                            |                                   |                                                                                                                                                                                                                                                                                                                              |

|                                                                                                                                      | <u>Statement<br/>Number</u> |                                                                                                                                                                                                                             | <u>Statement<br/>Number</u> |
|--------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| B9                                                                                                                                   | 32                          | Fail to B18)                                                                                                                                                                                                                |                             |
| Add the list of ORGANISM\$ to the end<br>of the list of OBJECT\$.                                                                    |                             | Add this Type, DESCRIPTION, Name-of-<br>Object and its location (RO * CO) to<br>LVIEW (the Left-VIEW).                                                                                                                      | 53                          |
| TNEXT                                                                                                                                | 33                          | Do the old locations (ROA, COA) of this<br>object EQUALS the new, computed<br>locations (RO, CO)? Yes - Go to B11<br>(no changes).                                                                                          | 54                          |
| (after each Time-NEXT) PUTOUT the<br>present state of the mikrokosm.                                                                 |                             | Is this object of Type 'SHOUT' (which<br>is located everywhere)? Yes - Go to B11.                                                                                                                                           | 55                          |
| Advance TIME by 1.                                                                                                                   | 34                          |                                                                                                                                                                                                                             |                             |
| Is TIME GreaterThan 5? Yes - END this<br>run.                                                                                        | 35                          | B20                                                                                                                                                                                                                         | 56                          |
| TN1                                                                                                                                  | 36                          | To CHANGES, add this object (its Type,<br>DESCRIPTION, Name-Object, and its<br>changes in location (RO-ROA, CO-COA)).<br>Go to B11.                                                                                         |                             |
| No - Make a Copy-ORGANISM\$ (CORGS)<br>of ORGANISM\$ (ORGS) list.                                                                    |                             | B18                                                                                                                                                                                                                         | 57                          |
| B7                                                                                                                                   | 37                          | To the beginning of LVIEW, add this new<br>object (its Type, DESCRIPTION, Name-of-<br>Object, and location (RO * CO)). Go to<br>B20.                                                                                        |                             |
| Get the next Name-of-ORGANISM from<br>CORGS. (If no more, Go to UPDATER)                                                             |                             | **THIS ORGANISM NOW APPLIES ITS LEARNED<br>HYPOTHESES TO THE VIEW PRESENTED IT.                                                                                                                                             |                             |
| From NORG, get this organism's STATE,<br>HYPOTHESES, PASTDONE acts, VIEW,<br>and OLDCHANGES.                                         | 38                          | B12                                                                                                                                                                                                                         | 58                          |
| From STATE, get this organism's<br>LOCATION= (ROW, COL).                                                                             | 39                          | Let VIEW contain what LVIEW contains.                                                                                                                                                                                       |                             |
| Let the LEFT border of this organism's<br>view be its COLUMN minus OSEE<br>(distance-in-one-direction-of<br>Organism's-SEEN-view).   | 40                          | Let MAYDO equal PRIMDO (those acts<br>PRIMITIVELY-DONE).                                                                                                                                                                    | 59                          |
| Let the RIGHT border be its COLUMN plus<br>OSEE.                                                                                     | 41                          | B16                                                                                                                                                                                                                         | 60                          |
| B10                                                                                                                                  | 42                          | From LVIEW, get the next object's Type,<br>DESCRIPTION, Name-of-Object, and<br>location (RO * CO). (If no more, Fail to<br>B13.)                                                                                            |                             |
| Make a Copy-OBJECT\$ (COBJS) of OBJECT\$<br>(OBJS) list.                                                                             |                             | If this Type and its DESCRIPTION are found<br>on the list of this organism's HYPOTHESES,<br>get the acts-to-DO, state-of-SELF,<br>EXPECTATIONS, and Weight. (If not found,<br>Fail to B16.)                                 | 61                          |
| BLANK out (erase) LVIEW and CHANGES.                                                                                                 | 43                          | Does this organism's STATE contain the<br>SELF-state specified for this hypothesis?<br>(No - Go to B16.)                                                                                                                    | 62                          |
| *LIST FOR THIS ORGANISM THE OBJECTS IN<br>ITS VIEW, INCLUDING ITSELF.                                                                |                             | Let VALUE equal the EXPECTED-need-<br>change times the Weight of this<br>hypothesis, divided by the Row plus<br>COLUMN distance plus 10.                                                                                    | 63                          |
| B11                                                                                                                                  | 44                          | Get the absolute VALUE (delete any minus<br>sign (-)).                                                                                                                                                                      | 64                          |
| From COBJS, get the next Name-of-<br>Object (NO), its Type (TP), and<br>DESCRIPTION. (If no more, Fail to B12).                      |                             | Add this hypothesis to the list of things<br>the organism MAYDO: VALUE, acts-to-DO,<br>Name-of-Object, location (RO * CO), what<br>to EXPECT, the object's Type and<br>DESCRIPTION, and the SELF state found. Go<br>to B16. | 65                          |
| For this Named-Object (NO), get its<br>LOCATION (Row * COLUMN).                                                                      | 45                          | **CHOOSES TO DO AS MOST HIGHLY VALUED<br>HYPOTHESIS ON MAYDO SUGGESTS.                                                                                                                                                      |                             |
| Is this object of Type 'SHOUT'? Yes -<br>Go to B17.                                                                                  | 46                          | B13                                                                                                                                                                                                                         | 66                          |
| Is this object's COLUMN GreaterThan<br>the LEFTmost column viewed? No -<br>Go to B11 (it's not in this organism's<br>field of view). | 47                          | BLANK out VALA, XXA, LM.                                                                                                                                                                                                    |                             |
| Yes - Is CO LessThan the RIGHTmost<br>column viewed? No - Go to B11.                                                                 | 48                          | From MAYDO, get the first VALUEA and the<br>rest of the information (XXA) about this<br>hypothesis. (If none Fail to B15.)                                                                                                  | 67                          |
| *GET AND USE DISTANCES FROM ORGANISM<br>TO OBJECTS.                                                                                  |                             |                                                                                                                                                                                                                             |                             |
| Let RO (Row-of-Object) equal ROW<br>(ROW-of-organism) minus RO (to get<br>row distance from organism to object).                     | 49                          |                                                                                                                                                                                                                             |                             |
| Let CO (Column-of-Object) equal COL<br>(COLUMN-of-organism) minus CO.                                                                | 50                          |                                                                                                                                                                                                                             |                             |
| BLANK out ROA, COA.                                                                                                                  | 51                          |                                                                                                                                                                                                                             |                             |
| B17                                                                                                                                  | 52                          |                                                                                                                                                                                                                             |                             |
| From this organism's VIEW, get the Type<br>(TP), DESCRIPTION, Name-of-Object (NO),<br>and its location (ROA * COA). (If no more,     |                             |                                                                                                                                                                                                                             |                             |

|                                                                                                                                                                                                                                        | <u>Statement</u><br><u>Number</u> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| B14                                                                                                                                                                                                                                    | 68                                |
| From MAYDO, get the next value (VALB) and the rest of the information (XXB) about that hypothesis. (If no more hypotheses, Fail to B15.)                                                                                               |                                   |
| Is VALB GreaterThan VALA? (If yes, Go to B19.)                                                                                                                                                                                         | 69                                |
| No - Add to LM (Lesser-Maydo) VALB and XXB (the less valued hypothesis). Go to B14.                                                                                                                                                    | 70                                |
| B19                                                                                                                                                                                                                                    | 71                                |
| Add to LM VALA and XXA (the less valued hypothesis).                                                                                                                                                                                   |                                   |
| Let VALA equal VALB (which has a higher value).                                                                                                                                                                                        | 72                                |
| Let XXA equal XXB. Go to B14.                                                                                                                                                                                                          | 73                                |
| B15                                                                                                                                                                                                                                    | 74                                |
| Add to UPDATE this organism's name (NORG), location (ROW * COL) and the information about its most highly valued implied hypothesis (XXA)                                                                                              |                                   |
| From PASTDONE, erase the act-done farthest in the past (XX, which is first on this inverted list), and add the chosen hypothesis (VALA, XXA, followed by all Less-valued-Maydos (LM)).                                                 | 75                                |
| Re-store under this organism's name (NORG) its updated STATE, HYPotheses, PASTDones, VIEW, and CHANGES. Go to B7, to start processing for the next organism.                                                                           | 76                                |
| **UPDATE THE MIKROKOSM FOR THE NEXT TIME PERIOD.                                                                                                                                                                                       |                                   |
| UPDATER                                                                                                                                                                                                                                | 77                                |
| Let Copy-ObjectS contain what ObjectS contains; erase ObjectS.                                                                                                                                                                         |                                   |
| **UPDATE OBJECTS ACCORDING TO PHYSICAL LAWS.                                                                                                                                                                                           |                                   |
| U3                                                                                                                                                                                                                                     | 78                                |
| From COBJS, get the next object's Name-Object, TyPe, DEscription, the first symbol after the '=' (call it OX), and the rest (YY). (If no more objects, Fail to U1.)                                                                    |                                   |
| In this Named-Object, get its motion (DXY) in X (DX) and Y (DY) directions.                                                                                                                                                            | 79                                |
| In NO, get, and erase, its LOcation (RO * CO).                                                                                                                                                                                         | 80                                |
| BORDER this object (computing from its location (RO and CO) and motion (DX and DY) whether it will remain within the space, so that it will be made to stick to the border) - a function that starts with the statement labeled BORDI. | 81                                |
| In NO, put the newly-computed LOcation (NEWRoW-Column).                                                                                                                                                                                | 82                                |

|                                                                                                                                                                               | <u>Statement</u><br><u>Number</u> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Add to OBJECTS this object: Name-of-Object, TyPe, DEscription, what's in OX, and its new location (NEWRc).                                                                    | 83                                |
| Add to REPL this object's location (RO * CO) and TyPe. Go to U3.                                                                                                              | 84                                |
| **FUNCTION THAT COMPUTES NEW LOCATION AND KEEPS OBJECT WITHIN BORDERS.                                                                                                        |                                   |
| BORDI                                                                                                                                                                         | 85                                |
| Let NRO = RW + DR (new-ROW = RoW + Delta-Row).                                                                                                                                |                                   |
| Let NCO = CL + DC (New-CoLumn = CoLumn + Delta-Column).                                                                                                                       | 86                                |
| If NRO is LessThan 1, let NRO equal 1.                                                                                                                                        | 87                                |
| If NRO is GreaterThan RWS (the last RoWS), let NRO equal RWS.                                                                                                                 | 88                                |
| If NCO is LessThan 1, let NCO equal 1.                                                                                                                                        | 89                                |
| If NCO is GreaterThan CLS (the last CoLumnS), let NCO equal CLS.                                                                                                              | 90                                |
| Subtract 1 from CoLumn                                                                                                                                                        | 91                                |
| In the designated RoW of the Picture of the space, replace the SSymbol where this object used to be (CL symbols from the left) by a period (.) (which indicates empty space). | 92                                |
| Let NEWRoW-Column equal NRO * NCO. RETURN from this function.                                                                                                                 | 93                                |
| **UPDATE AS A FUNCTION OF ORGANISMS' CHOSEN ACTS.                                                                                                                             |                                   |
| U1                                                                                                                                                                            | 94                                |
| From UPDATE, get the next chosen-act-to-do (Name-of-ORGanism, ROW, COlumn, acts-to-DO, Name-of-Object, and the rest (XX)). (If no more, Fail to TNEXT.)                       |                                   |
| Get the LOcation (RO * CO) of this Named-Object.                                                                                                                              | 95                                |
| Erase WASH2 from NO.                                                                                                                                                          | 96                                |
| U2                                                                                                                                                                            | 97                                |
| Put LOBJS back at the left of OBJECTS.                                                                                                                                        |                                   |
| Erase (WASH) LOBJS.                                                                                                                                                           | 98                                |
| U2A                                                                                                                                                                           | 99                                |
| From DO, get the next specific DO-X.                                                                                                                                          |                                   |
| If succeed, go to the statement whose label is stored in DOX; if Fail, go to U1.                                                                                              |                                   |
| **THE ROUTINES THAT EFFECT THE SPECIFIC ACTS FOLLOW.                                                                                                                          |                                   |
| GET                                                                                                                                                                           | 100                               |
| BLANK out DX and DY.                                                                                                                                                          |                                   |
| PRINT out '///GETS///' (to inform that acting).                                                                                                                               | 101                               |
| Does the organism's ROW Equal the object's ROW? Yes - Go to G1.                                                                                                               | 102                               |
| No - If the organism's ROW is Greater-Than the object's ROW, let DX (change in the X direction) equal -1 and Go to G2.                                                        | 103                               |
| Or let DX equal +1, and Go to G2.                                                                                                                                             | 104                               |

|                                                                       | <u>Statement<br/>Number</u> |                                            | <u>Statement<br/>Number</u> |
|-----------------------------------------------------------------------|-----------------------------|--------------------------------------------|-----------------------------|
| G1                                                                    | 105                         | BLANK out PAST, OPAST, and DEX             | 125                         |
| Does the organism's COL Equal the<br>object's Column? Yes - Go to G2. |                             | From this Named-ORGanism, get its          | 126                         |
| No - If the organism's COL is                                         | 106                         | STATE, HYPotheses, PASTDOne,               |                             |
| GreaterThan the object's COL, let                                     |                             | VIEW, and OCHANGES (get, but don't         |                             |
| DY (change in the Y direction) equal                                  |                             | erase them.)                               | 127                         |
| -1 and Go to G2.                                                      |                             |                                            |                             |
| Or let DY equal +1.                                                   | 107                         | L3                                         | 127                         |
| G2                                                                    | 108                         | From PASTDOne, get the next chosen-        |                             |
| From OBJECTS, get this NORG's ROW                                     |                             | act's VALue, DZ, Name-of-Object-I          |                             |
| and COL.                                                              |                             | (NOI), location (ROI * COI), EXpectations, |                             |
| BORDER this organism (calling the                                     | 109                         | and the REST of the conjectured hypotheses |                             |
| function BORD, which will move it                                     |                             | for that time period. (If no more, Fail to |                             |
| and keep it within the borders).                                      |                             | L6.)                                       |                             |
| On OBJS, find NORG, and add its new                                   | 110                         | From EXpect, get EXPECT, TyPeI, DES-       | 128                         |
| location (NEWRC).                                                     |                             | criptionI, and SELF.                       |                             |
| Add this location (NEWRC) and Name-                                   | 111                         | Add 1 to PAST.                             | 129                         |
| of-ORGanism to REPL.                                                  |                             | Add the REST to Other-PAST                 | 130                         |
| Change the LOCation stored under                                      | 112                         | Does the Named-Object EQUALS the           | 131                         |
| this Name-of-ORGanism to NEWRC.                                       |                             | Named-ObjectI? No - Fail to L3.            |                             |
| Go to U2, to get and do the next act.                                 |                             | **RE-WEIGHTS AN HYPOTHESIS ABOUT THE       |                             |
| DESTROY                                                               | 113                         | OBJECT EATEN.                              |                             |
| Get the NEXT object from OBJECTS. (If                                 |                             | L7                                         | 132                         |
| no more, Fail to U2.)                                                 |                             | Yes - On the organism's HYPotheses,        |                             |
| Add this object to Left-OBJECTS.                                      | 114                         | if TyPeI and DEScriptionI are found,       |                             |
| From NEXT, if the location is NEWRC,                                  | 115                         | get and remove DoY, SELF, EXPECT,          |                             |
| immediately following '=*', get                                       |                             | and WeighT.                                |                             |
| Name-Object, TyPe, and DEScription.                                   |                             | Add Delta-EXpectation (DEX) to EXPECT.     | 133                         |
| (If fail, go to DESTROY.)                                             |                             | WASH (erase) DEX.                          | 134                         |
| Succeed - Add this Name-of-Object,                                    | 116                         | Is the CALoric content negative (-)?       | 135                         |
| as 'DESTROYEDBY' the Name-of-                                         |                             | Yes - Go to L4.                            |                             |
| ORGanism, to GONEOBJs. Go to U6.                                      |                             | No - Add 1 to WeighT.                      | 136                         |
| FLAIL                                                                 | 117                         | L5                                         | 137                         |
| From RANDOM, get the first two (pseudo-                               |                             | Put back on HYPothesis the components      |                             |
| random) numbers (called DX and DY),                                   |                             | of the hypothesis that was taken off by    |                             |
| and put them at the end of RANDom.                                    |                             | statement L7 and reweighted. Go to U6.     |                             |
| Go to G2, where these will be used to                                 |                             | L4                                         | 138                         |
| randomly move and flail the organism.                                 |                             | Lower WeighT by 1 (this was a noxious      |                             |
| EAT                                                                   | 118                         | object that the organism should learn      |                             |
| Get the NEXT object from OBJECTS. (If                                 |                             | not to eat).                               |                             |
| no more, Fail to U2.)                                                 |                             | Is the WeighT now LessThan 1? No - Go      | 139                         |
| Add this object to Left-OBJECTS.                                      | 119                         | to L5 (to put this down-weighted           |                             |
| From NEXT, if the location is NEWRC,                                  | 120                         | hypothesis back onto HYPotheses).          |                             |
| immediately following the '=*', get                                   |                             | **DISCARD AN HYPOTHESIS WHOSE WEIGHT HAS   |                             |
| Name-of-Object, TyPe, and                                             |                             | GONE BELOW 1.                              |                             |
| DEScription. (If fail, Go to EAT)                                     |                             | Add to the list of DISCARDS the TyPeI,     | 140                         |
| Succeed - PRINT out '///EAT///'.                                      | 121                         | DEScriptionI, DoY, SELF, and EXPECT        |                             |
| For this Name-of-Object, get its                                      |                             | of this discarded hypothesis. Go to L8.    |                             |
| CALoric VALue.                                                        | 122                         | *IF NONE OF THE HYPOTHESES ACTED UPON WAS  |                             |
| From this Name-of-ORGanism, get its                                   | 123                         | PERTINENT, CHECK ALL OTHERS ON OPAST.      |                             |
| ENERgy level, and add the object's                                    |                             | L6                                         | 141                         |
| CALoric value to this Energy level.                                   |                             | From Other-PAST, get the next hypothesis   |                             |
| Add this Name-of-Object as 'ATEBY'                                    | 124                         | (VALue, DoY, Name-of-ObjectI, location     |                             |
| this Name-of-ORGanism to GONEOBJs.                                    |                             | (ROWI * COLI), and EXpectation). (If no    |                             |
| **RE-WEIGHTS LEARN HYPOTHESES AS A                                    |                             | more, Fail to L8)                          |                             |
| FUNCTION OF FEEDBACK FROM EATING.                                     |                             | From EXpect, get EXPECT, TyPeI,            | 142                         |
| **FINDS AN HYPOTHESES ABOUT THIS EATEN                                |                             | DEScription, and SELF.                     |                             |
| OBJECT.                                                               |                             | Does Named-Object Equal Named*             | 143                         |
|                                                                       |                             | ObjectI? No - Fail to L6.                  |                             |
|                                                                       |                             | Yes - Let Delta-Expectation (DEX)          | 144                         |

|                                                                              | <u>Statement</u>                                                                                                                                     | <u>Number</u> |
|------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
|                                                                              | equal 3 (so that this hypothesis will have a higher expectation, and will more likely be chosen and acted upon, in the future). Go to L7.            |               |
| **A NEW HYPOTHESIS IS GENERATED                                              |                                                                                                                                                      |               |
| L8                                                                           |                                                                                                                                                      | 145           |
|                                                                              | Is the CALoric content of the eaten object negative (-)? Yes - Go to L9.                                                                             |               |
|                                                                              | No - Let New-Do (ND) equal "GET, EAT.". Go to L10.                                                                                                   | 146           |
| L9                                                                           |                                                                                                                                                      | 147           |
|                                                                              | Let the New-Do (ND) equal "GET, DESTROY.".                                                                                                           |               |
| L10                                                                          |                                                                                                                                                      | 148           |
|                                                                              | See if DISCARDS already has this new hypothesis' TyPe, DEScription, and New-Do. (If yes, Go to U6.)                                                  |               |
|                                                                              | Add this newly-generated hypothesis to the organism's HYPotheses: its TyPe, DEScription, New-Do, and initial expectation-level (90), and weight (5). | 149           |
| U6                                                                           |                                                                                                                                                      | 150           |
|                                                                              | Erase information about this eaten or destroyed Named-Object.                                                                                        |               |
|                                                                              | Put the Left-OBjectS back onto the start of the OBjectS list.                                                                                        | 151           |
|                                                                              | WASH (erase) Left-OBjectS.                                                                                                                           | 152           |
|                                                                              | Erase this (eaten or destroyed) Named-Object from the OBjectS list. Go to U2A.                                                                       | 153           |
| **FUNCTION FOLLOWS TO PRINT OUT INFORMATION ABOUT MIKRO AT EACH TIME PERIOD. |                                                                                                                                                      |               |

#### Bibliography

- Doran, J.E., Experiments with a pleasure seeking automaton. In: Machine Intelligence 3, (ed. Michie, D.) Edinburgh: University Press; 1968a, pp. 195-216.
- Doran, J.E., A simulated robot/environment system: progress and problems, 1968b.
- Gelernter, H., Realization of a geometry theorem proving machine. Proc. International Conference on Information Processing, Paris: UNESCO, 1959.
- Greenblatt, R.D., Eastlake, D.E., & Crocker, S.D., The Greenblatt chess program. Proc. Fall Joint Computer Conf., 1967.
- Kochen, M. and Uhr, L., A model for the process of learning to comprehend. In: Information Science (M. Kochen, Ed.) New York: Scarecrow Press, 1965.
- Minsky, M. and Papert, S., Discussion of MIT robot project. Presented at Int. Pattern Recognition Conf., Puerto Rico, 1967.
- Nagy, G., State of the Art in Pattern Recognition, Proceedings of the IEEE, 1968, Vol. 56, no. 5.
- Newell, A. and Ernst, G., The search for generality. Proc. IFIP Congress 65. New York, 1965, pp. 17-23.
- Newell, A., & Shaw, J.C., & Simon, H.A., Empirical explorations of the logic theory machine. Proc. West. Joint Comp. Conf., 1957, 218-239.
- Newman, C. and Uhr, L., BOGART: A discovery and induction program for games. Proc. 20th Annual Meeting of the ACM, 1965.
- Pingle, K.K., Singer, J.A. & Wichmann, W.M., Computer control of a mechanical arm through visual input. Proc. IFIP Congress 68, Edinburgh, 1968, pp. H140-H147.
- Pitrat, J., Realization of a general game-playing program. Proc. IFIP Congress 68, Edinburgh, 1968, pp. H120-H124.
- Raphael, B., Programming a robot. Proc. IFIP Congress 68, Edinburgh, 1968, pp. H135-H140.
- Samuel, A.L., Some studies in machine learning using the game of checkers. IBM J. Res. and Develop., 1959, 3, 210-229.
- Sutro, L.L. & Kilmer, Assembly of computers to command and control a robot. Proc. Spring Joint Comp. Conf., 1969.
- Toda, M., The design of a fungus-eater: a model of human behavior in an unsophisticated environment, Behav. Sci., 1962, 7, 164-183.
- Travis, L.E., Experiments with a theorem-utilizing program. Proc. Spring Joint Computer Conf., 1964.
- Uhr, L. (Ed.) Pattern Recognition, New York: Wiley, 1967.
- Uhr, L., Pattern Recognition, Problem Solving and Learning. 1969 (in preparation).
- Zobrist, A., A model of visual organization for the game of GO. Proc. Spring Joint Comp. Conf., 1969.

B 5 5 0 0      S N O B O L 3      S Y S T E M

THE DATE IS 10 MAR 1969  
COMPILATION STARTED AT 9:32: 7

-553-

```

61 $HYP TP "," DES "," *DO* "," *SELF* "," *EXPECT* "," *WT* "/"
62 /F(B16)
63 STATE SELF "," /F(B16)
64 * VALUE IS FUNCTION OF EXPECTED NEED-SATISFACTION WT AND DISTANCE
65 VALUE = ( EXPECT * WT ) / ( ( RO + CO ) + "10" )
66 VALUE "==" = VALUE "," DO "," NO "," RO "==" CO " (" EXPECT ","
67 TP "," DES "," SELF "," ) /F(B16)
68 * CHOOSE TO DO MOST HIGHLY VALUED "MAYDO"
69 B13 BLANK *VALA* *XXA* *LM*
70 MAYDO *VALA* "," *XXA* "," = /F(B15)
71 B14 MAYDO *VALB* "," *XXB* "," = /F(B15)
72 GT( VALB , VALA ) /S(B19)
73 LM = LM VALB "," *XXB* "," /F(B14)
74 B19 LM = LM VALA "," *XXA* ","
75 VALA = VALB
76 XXA = XXB /F(B14)
77 * PUT ACT WITH HIGHEST EXPECTED VALUE ON UPDATE
78 B15 UPDATE = UPDATE NORG "," ROW "==" COL "==" XXA "=="
79 PASTDO *XX* "==" *YY* = YY VALA "==" XXA "==" LM "/"
80 $NORG = STATE "$" HYP "$" PASTDO "$" VIEW "$" CHANGES /F(B7)
81 * LETS 2 OBJS OCCUPY SAME SPACE (BETTER TO TEST OVERLAP , BOUNCE THEM)
82 ** UPDATE MIKRO FOR NEXT TIME PERIOD
83 UPDATER OBJS *COBJS* =
84 ** UPDATE FROM OBJECTS QUALITIES
85 U3 COBJS *NO* "==" *TP* "==" *DES* "==" *OX* "1" *YY* "/" = /F(U1)
86 $NO *OXY* "==" *DX* "==" *DY* "=="
87 $NO *LOC* "==" *RO* "==" *CO* "==" =
88 *WILL STICK TO BORDER
89 BORD( RO , DX , CO , DY )
90 $NO = "LOC=" NEWRC "," $NO
91 OBJS = OBJS NO "==" TP "==" DES "==" OX NEWRC "/"
92 REPL = REPL RO "==" CO "==" TP "==" /F(U3)
93 BORD1 NRO = RW + DR
94 NCO = CL + DC
95 NRO = LT( NRO , "1" ) "1"
96 NRO = GT( NRO , RWS ) RWS
97 NCO = LT( NCO , "1" ) "1"
98 NCO = GT( NCO , CLS ) CLS
99 CL = CL - "1"
100 $( "P" RW ) *XX/CL* *SS/"1" = XX ","
101 NEWRC = NRO "==" NCO /F(RETURN)
102 ** UPDATE FROM ORGANISMS CHOSEN ACTS
103 U1 UPDATE *NORG* "==" *ROW* "==" *COL* "==" *DO* "==" *NO* "==" *XX* "=="
104 = /F(TNEXT)
105 $NO *LOC* "==" *RO* "==" *CO* "=="
106 NO *WASH2* =
107 U2 ORJS = LOBJS OBJS
108 WASH *LOBJS*
109 U2A DO *DOX* "==" = /S($DOX)F(U1)
110 *CODE FOR THE SPECIFIC ACTS FOLLOWS
111 GET BLANK *DX* *DY*
112 SYSPOT = " ///GETS///"
113 EQ( ROW , RO ) /S(G1)
114 DX = GT( ROW , RO ) "-1" /S(G2)
115 DX = "1" /F(G2)
116 * ALREADY GOT - RETURN
117 G1 EQ( COL , CO ) /S(G2)
118 DY = GT( COL , CO ) "-1" /S(G2)
119 DY = "1"
120 G2 OBJS NORG "==" *ROW* "==" *COL* "/" = NORG "=="
121 BORD( ROW , DX , COL , DY )
122 OBJS NORG "==" = NORG "==" NEWRC "/"
123 REPL = REPL NEWRC "==" NORG "=="
124 $NORG *LOC* "==" *XX* "==" = "LOC=" NEWRC "==" /F(U2)
125 DESTROY OBJS *NEXT* "/" = /F(U2)
126 LOBJS = LOBJS NEXT "/"
127 NEXT NO "==" *TP* "==" *DES* "==" NEWRC /F(DESTROY)
128 GONEOBJS = GONEOBJS NO "==" DESTROYEDBY "==" NORG "==" /F(U6)
129 * FOR RANDOM MOTION (ALSO WANT BABBLE)
130 FLAIL RAND *DX* "==" *DY* "==" *XX* = XX DY "==" DX "==" /F(G2)
131 * BEGINS DEFINED FUNCTION (RIGHT BEFORE TNEXT)
132 EAT OBJS *NEXT* "/" = /F(U2)
133 LOBJS = LOBJS NEXT "/"
134 NEXT NO "==" *TP* "==" *DES* "==" NEWRC /F(EAT)
135 SYSPOT = " ///EAT///"
136 $NO *VAL* "==" *CAL* "=="
137 $NORG *EN* "==" *EN* "==" = "EN=" EN + CAL "=="
138 GONEOBJS = GONEOBJS NO "==" ATERY "==" NORG "=="
139 *** LEARN NEW HYP *****
140 BLANK *PAST* *OPAST* *DEX*

```

```

126 $NORG *STATE* "$" *HYP* "$" *PASTDO* "$" *VIEW* "$" *OCHANGES*
127 L3 PASTDO *VAL* " " *DZ* " " *NOI* " " *ROI* " " *COI* "("
    *EX* " " *REST* " " = /F(L6)
128 * EX *EXPECT* " " *TPI* " " *DESI* " " *SELF*
    DONT YET USE THIS COUNT OF HOW FAR PAST
129 PAST = PAST + "1"
130 OPAST = OPAST REST
131 EQUALS( NO , NOI ) /F(L3)
132 L7 $HYP TPI " " DESI " " DY " " *SELF* " " *EXPECT* " " *WT* "/" =
133 EXPECT = EXPECT + DEX
134 WASH *DEX*
135 CAL " " = /S(L4)
136 WT = WT + "1"
137 L5 $HYP = TPI " " DESI " " DY " " SELF " " EXPECT " " WT "/" $HYP
    /(U6)
138 L4 WT = WT + "1"
139 .LT( WT , "1" ) /F(L5)
140 DISCARDS = DISCARDS TPI " " DESI " " DY " " SELF " " EXPECT "/"
    /(L8)
    NONE OF THE HYP ACTED ON CHOSEN, CHECK THE REST ON OPAST
    IGNORES "$" WHICH TELLS ABOUT TIME PERIOD
141 L6 OPAST *VAL* " " *DY* " " *NOI* " " *ROI* " " *COI* "("
    *EX* " " = /F(L8)
142 * EX *EXPECT* " " *TPI* " " *DES* " " *SELF*
143 .EQ( NO , NOI ) /F(L6)
144 DEX = "3" /(L7)
    FORMS NEW HYP
145 L8 CAL " " = /S(L9)
146 ND = "GET.EAT" /F(L10)
147 L9 ND = "GET.DESTROY"
148 L10 DISCARDS TP " " DES " " ND " " /S(U6)
149 $HYP = $HYP TP " " DES " " ND " " ,90,5/"
150 U6 $NO =
151 ORJS = LOBJS OBJS
152 WASH *LOBJS*
153 ORJS NO " " *XX* "/" = /F(U2A)
154 PUT2 SYSPOT = " AT TIME = " TIME " , MIKRO IS= "
    *REPL GETS ONLY SOME
155 PUT1 BLANK *JJ* *OUTI* *II* *LISTO*
156 P1 JJ = JJ + "1"
157 .GT( JJ , CLS ) /S(P2)
158 OUTI = OUTI " " /F(P1)
159 P2 II = II + "1"
160 .GT( II , RWS ) /S(P3A)
161 $( "P" II ) = OUTI /F(P2)
162 P3A COBJS = OBJS
163 WASH *II* *LISTO*
164 P3 COBJS *NO* " " *TP* " " *DES* " " *XX* "/" = /F(P4)
165 $NO *LOC* " " *RO* " " *CO* " " *XX* "DX=" *DX* " " *DY* " "
166 LISTO = LISTO NO " " RO " " CO " " ,DX=" *DX* " " *DY* "/"
167 CO = CO + "1"
168 CO " " *XX* = "0"
169 $( "P" RO ) *XX/CO* TP /S(P3)
170 $( "P" RO ) *XX/CO* *SS/" " = XX TP /F(P7)
171 REPL = REPL RO " " CO " " $S " " /F(P3)
172 P7 $( "P" RO ) = $( "P" RO ) " " TP /F(P3)
173 P4 II = II + "1"
174 .GT( II , RWS ) /F(P4)
175 P5 SYSPOT = " REPLACED = " REPL
176 WASH *REPL*
177 SYSPOT = " OBJ (NAME=LOC,MOVEMENT) = " LISTO
178 CORGS = ORGS
179 SYSPOT = " ORGS AND THEIR INFO ARE= "
180 P6 CORGS *NORG* " " *XY* " " = /F(RETURN)
181 SYSPOT = " ORG " NORG " "
182 $NORG *STATE* "$" *HYP* "$" *PASTDO* "$" *VIEW* "$" *CHANGES*
183 SYSPOT = " STATE
184 SYSPOT = " HYP = " $HYP
185 SYSPOT = " PASTDO = " PASTDO
186 SYSPOT = " VIEW = " VIEW
187 SYSPOT = " CHANGES = " CHANGES /F(P6)
188 END GO

```

0 SYNTAX ERRORS DETECTED

COMPILATION COMPLETED AT 9:34:36

CPU TIME = 58.4 SEC:

I/O TIME = 82.1 SEC:

EXECUTION STARTED AT 9:34:36

