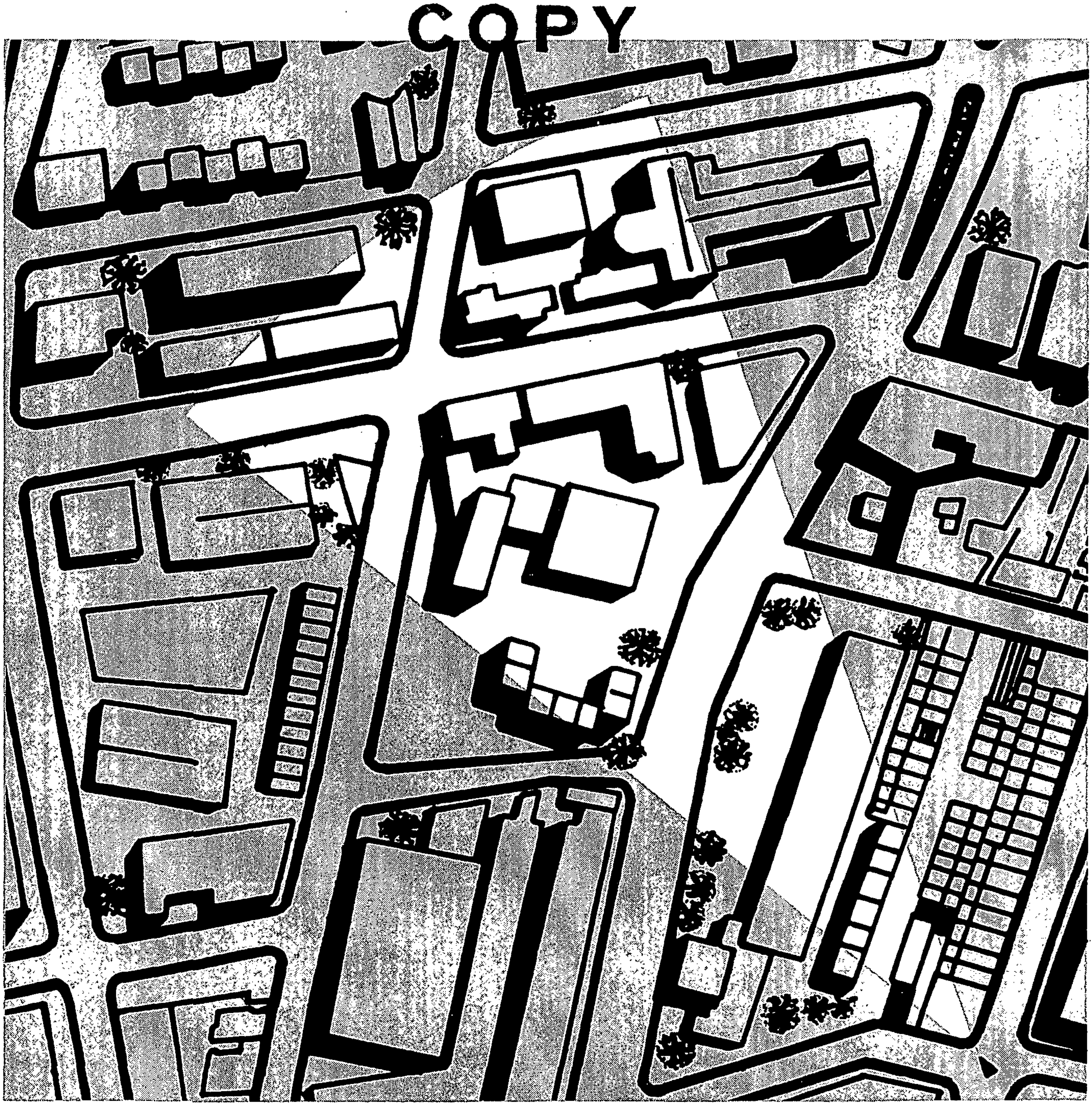
ROBOT
Computer Problem Solving System

Bolt Beranek and Newman Inc.
Report No. 2316

CR-129526

September 1972

Joseph D. Becker



prepared for
National Aeronautics and Space Administration
Washington, D.C. 20546

Report No. 2316

"ROBOT" COMPUTER PROBLEM SOLVING SYSTEM

Final Progress Report

For the period 23 August 1971 to 22 August 1972

Contract No. NASW-2236

Joseph D. Becker
Principal Investigator

15 September 1972

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

I. INTRODUCTION

The following is a report on progress made during a full year of work on NASA Contract No. NASW-2236, whose subject is the development of a "robot" computer problem solving system. We will begin with a brief summary of just what we mean by "robot" computer problem solving, and why we chose to study it.

A. The Nature of "Robot" Problem Solving

There have been a number of computer problem solving systems developed in the past, but most of these have been concerned with purely "mental" problems such as playing chess. These problem solvers do not necessarily work efficiently when presented with "physical" problems of the sort faced by a living animal or by an artificial animal ("robot") moving about in a complex environment. The reason is that interaction with a real environment involves several fundamental constraints that are not present in the abstract problem solving situation. Below we will give four major characteristics of the robot situation, and contrast them with the constraints on a purely "mental" problem solver.

(1) Richness of information

The predominant characteristic of the real world is that it contains far more information than any robot system can ever cope with all at once. There will always be many details of any situation that the robot does not have time to attend to, and there will always be regions around it that it does not have time to explore. This situation contrasts very sharply with a situation such as in chess, where the problem solving system has available to it *all* the information that is knowable about the problem. Much of the design problem in robot intelligence involves mechanisms for obtaining and selecting the necessary information in any situation from the flood of irrelevant information that surrounds the system.

(2) Uncertainty

A robot must always cope with uncertainty in everything it plans or does. It is uncertain of what it "knows", because the world may change, or the robot may have incorrectly identified the situation it is in. The robot cannot even be certain of the consequences of its actions: it may think it is picking up an object, when actually the object has fallen out of its grip. The chess-playing program, again, has infallible knowledge of all that is true in its little world, and all of its actions are quaranteed to have the intended result. Thus, a "robot" problem solving system must overcome obstacles to its performance that are totally absent in a "mental" problem solving system.

(3) Interaction

Because of the richness and uncertainty of its world, the robot depends on continual interaction with the world in order to update its knowledge. This is especially important in that it allows the robot to become aware of totally unexpected situations (e.g. an object in its path) that may interfere with or aid its plans. By contrast, chess-playing programs do not actually interact with their opponents; instead, they treat each new board-position as a totally new abstract problem. There is no need to seek information from the board, and no chance of anything utterly unexpected occurring. Indeed, the "mental" problem solving system lacks a whole mode of behavior, namely control of action by *feedback*, that is vital to the operation of any robot system.

(4) Commitment

In order to accomplish anything, a robot system must actually commit itself to action. But whenever it acts, it is taking a risk; for example, if it moves forward it might fall into a hole, or hit some obstacle, or crush something of value. Of course, the chess-playing program must eventually commit itself to a move, but during the course of the problem solving operation itself, it makes all trial moves "in its head", and it can mentally take back any move that turns out to have unfavorable results. The robot does not have the luxury of conducting all its problem solving "in its head". For example, in order to gain enough information to come to a decision, it may have to operate a sensory device, or move an object, or move itself, all of which involve risks. Thus, the robot must take the risk of making real actions, even within the process of deciding whether to commit itself to other actions.

Because the task of a real-world robot has special constraints such as those just discussed, we felt that we should start from scratch in designing a problem solving system that would meet these special problems -- what we call a "robot" problem solving system. Accordingly, we programmed a simulated robot and a "simulated real world" that possess the fundamental informational properties of a real robot situation, but which bypass the many technical problems of actual hardware systems. This simulation proved to be simple enough to give us some theoretical insights into the nature of robot problem solving, while at the same time being complex enough to serve as a challenging task environment within which to construct experimental problem solving systems.

B. Contents of the Report

In terms of the simulated robot, our final achievement in this year of work is a system which can visually scan its environment and create an internal model of it, this being done in an informationally efficient manner (e.g., unusual objects receive more attention than commonplace ones, but no particular object monopolizes the robot's attention for long). Developmentally, we would have to rate the robot's behavior somewhat below that of a three-week-old kitten; but this is actually no small accomplishment, nor was it trivially achieved.

More important than the particular performance of our simulation is the theoretical understanding that we have gained in the course of working with it. The simulation model itself has no practical value, but it has given us experience with problems that must be faced in any robot system. In this report we have tried to emphasize the general principles that seem to underlie each particular problem.

This report is in fact organized so as to proceed from the specific to the general. Section II discusses the programming systems that we have used in constructing the problem solving system. Section III describes the simulated robot and its simulated world environment. Secion IV presents the experimental problem solving system at its present state of development, and analyzes the tasks that it is capable of performing. Section V sets forth some of the important problems that have not yet been dealt with in the current system. And Section VI outlines a very general framework for understanding the relationship between an observed behavior and an adequate description of that behavior.

II. PROGRAMMING SYSTEMS

If we had outlined this report a year ago, we would not even have included a section on programming systems. We are no longer so naive. When a theory is embodied in a computer program, the language in which the program is written takes on a direct theoretical significance. This assertion is supported both by our actual experience and by theoretical considerations (Section VI) concerning the manner in which behavioral systems are best described.

A. General vs. Special-Purpose Systems

There is a well-recognized trade-off between the flexibility of general-purpose programming systems and the ease of expression in special-purpose ones. Because the purpose of our investigation was to experiment with executive systems themselves (that is, problem solving executives for the robot), we chose to use one of the most flexible programming systems available, BBN LISP. We have occasionally been asked why we did not use the recently-touted PLANNER system developed at MIT. The reason is simply that PLANNER imposes a bias toward certain forms of program organization, and such a bias would be unhelpful or even antithetical to experimentation with various executive organizations.

The complete generality of BBN LISP, while necessary for our purposes, often proved to be burdensome in practice. This was especially true with respect to the absence of special data-structure types in LISP to correspond to the types of entities that we were creating in our model. To alleviate this problem, we implemented a "formatting" package that provides the following facilities:

(1) When the user defines the "format" of a data-type, the names of its sub-parts automatically become defined as accessing functions. For instance, suppose a STATE is defined as (CONDITIONS TIME) -- that is, as a list of conditions and a time. Then, if S4 is the name of a particular state, (TIME S4) will return the TIME part of S4. Different data-types can have the same sub-part names without confusion arising.

(2) The user can easily print selected parts of a data-type in a tree format. For example, the state S4 can be printed, among other ways, in the form:

```
S4  CONDITIONS  (C1 & & &)
                (C2 & & &)
                (C3 & & &)
    TIME  37.4
```

--where the ampersands indicate deeper list structures. Here the format of a data-type serves as a "grammar" for it, and the above sort of printout may be regarded as a selective "parsing" of the structure, presented in a very legible tree form.

(3) Formats can be changed at will, and data-structures can be reformatted in accordance with such changes. For example, if the sub-part TIME were eliminated from the *prototype* format for STATE, then a single function could be called which would eliminate the TIME sub-part from all *particular* states such as S4.

B. Temporal Organizations

There is a significant respect in which BBN LISP is *not* general-purpose, and this aspect of the LISP system has given us a good deal of trouble. This is simply the sequential, step-at-a-time, start-to-finish form of process execution that LISP shares with most other programming systems. In simulating the relationship between a perceiving, problem-solving robot and its environment, we often need to specify complex temporal interrelationships that are very cumbersome to express in LISP.

The most obvious case of a difficult temporal relationship is that of simultaneity. This important relationship can of course only be simulated on a serial computer. We have realized belatedly that there are several ways of performing such a simulation. One way is to establish a "clock time" (either simulated or real), and alternate computation among each of several processes, running each of them for a definite quantum of time. Another method (which is the one that we have actually employed) is to run each computation until it produces some sort of result, and then note how much clock time that computation took. A third possibility, which is a more elegant extension of the last, is to run each process a convenient distance into the "future", keeping a record of the results that it *will* produce at various future times, and then to advance the clock and simply read the inputs for that time-quantum off of the pre-computed records. The latter method has the advantage that computations can be run arbitrarily long until they hit logical break points; it has the disadvantage that some "future" results may have to be recomputed if there is too much interdependence among the various simultaneous processes.

An important variant on simultaneity is the notion of a "demon", a process which can go into a "dormant" state but which is set to "wake up" when some condition is fulfilled. It seems that many animal-like behaviors are very efficiently described in terms of demons (see Sections IV.E.2.B and VI.G), so it is important that there be a convenient way of programming them, even if there is no efficient way of implementing them on today's computers. Another related but more general concept is that of an "interrupt", which implies the active intervention of one process into another's computation. This notion appears to be indispensable in simulating the attention-shifting mechanisms which allow animals (and robots) to cope with the unexpected in real world environments.

As mentioned above, we have employed inelegant means to simulate simultaneous processing. We spent a bit of time devising a programming language for demons, but it was too clumsy in both form and execution to be of use. So, we have provided for interrupting processes by *ad hoc* means when the necessity has arisen. Clearly we are in need of programming (and computing) aids that will allow us efficiently to create temporal organizations that exceed the strict serialism of most present systems. Of course, various forms of multi-processing and even multi-computer organization are indeed becoming popular areas of investigation, often with a view to performing numerical calculations more efficiently. What we wish to emphasize here is that these novel forms of temporal organization, which are a luxury when applied to numerical calculations, may well be a necessity when it comes to creating the systems required for robot problem solving.

C. Inhomogeneous Systems

In speaking of the need for special-purpose programming facilities for creating robot problem solving systems, we should realize that some parts of such systems are more special-purpose than others. For example, the motor system in a robot, as in an animal, must cope with problems that are peculiar to the physical system that is being controlled; it would be pointless and inefficient to design such a system at the same level of generality as the problem solving executive. Indeed, any efficient large-scale system must be organized as a hierarchy of increasingly specific processes. In Section VI we discuss a number of theoretical problems created by such an organization. Here we will merely mention the practical problem that an efficiently designed system may require the use of several different special-purpose programming systems, and that furthermore these systems must be able to interact with each other in a coordinated way.

We have tried to avoid this problem by concentrating our efforts on the executive level, while simplifying the other levels such as the motor system down to the point of triviality. We feel that this approach has weakened the validity of our simulation model; we now believe that the problems of inhomogeneity and coordination should be faced more forthrightly. To our knowledge, the most impressive work on these problems to date is exhibited in the Stanford Research Institute's hardware robot, which successfully integrates systems in several different languages and even in different computers. This approach, while fraught with problems, apparently mirrors the organization of biological systems, and probably is the only efficient way of obtaining life-like behavior from a mechanical system.

III. THE SIMULATED ROBOT AND ITS WORLD

In order to have a definite problem space in which to study the design of a robot problem solver, we have chosen to simulate a two-dimensional environment which has the properties of a network of city streets. The simulated robot is equipped with a fairly complex visual system, with which it must scan its surroundings, identify its location, and find its way from point to point in the city network. In the first section below we will simply describe this simulation model as it now exists. In Section III.B we will consider some of the issues involved in the selection of this particular model.

A. The Simulation Model

A.1. Components of the Model

The world that confronts the robot is composed out of street sections that we may call "blocks", which are connected together in a geometrical pattern, and objects which stand along the sides of the streets. The objects are of three classes: buildings, roadsigns, and stoplights.

Blocks may vary in length (although all blocks in the current world have a length of 180 feet), and all blocks have the same non-zero width (currently 20 feet). They may join at any angle, and up to 8 blocks may join at a single intersection. With each block is associated a speed limit and a "safe speed" which is a single parameter incorporating such factors as number of lanes, density of traffic, quality of road surface, etc. The safe speed is always greater than or equal to the speed limit.

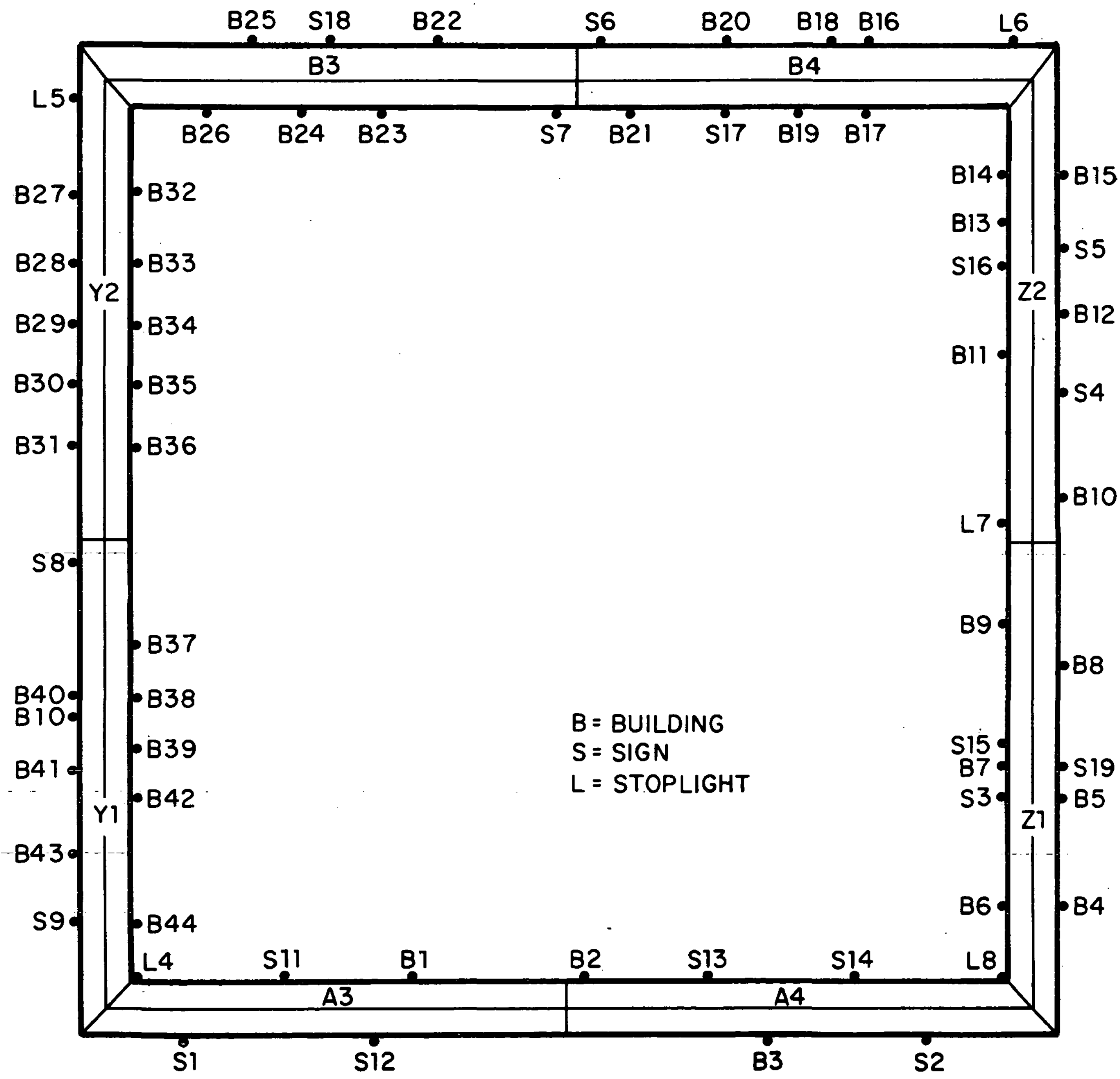
Each object in the world has a point location along the side of a block. It also has a list of visual features which are appropriate to its type. Thus, a building has the features HEIGHT, WIDTH, COLOR, TEXTURE, and DETAIL (the feature DETAIL is intended to provide a unique feature for each building). A roadsign has the features COLOR, SHAPE, and TEXT. A stoplight has a COLOR feature which is computed specially as a function of time.

The robot itself is represented by a set of variables that describe its location, velocity, and the state of its sensorimotor system. The driving system consists basically of an accelerator, brake and steering wheel. The sensory system consists basically of a speedometer and a single eye. The eye has a number of control parameters, which will be described in Section III.A.4. The robot perceives space in terms of an internal coordinate system described in Section III.A.5.

The world enforces a set of physical and (optionally) traffic laws, described in Section III.A.3.

A.2. Layout of the First Experimental "City"

For our first simulated "city" in which to develop the robot model, we have chosen a square that is two blocks on a side, as sketched below:



This square was chosen because it is the simplest layout satisfying the two criteria that:

(a) The robot can return to a given place without executing a u-turn, and

(b) The layout contains both straight and angular intersections.

The first of these criteria relates to the problem of recognizing a place on returning to it; the second relates simply to the mechanics of stopping and turning at intersections.

The simulated world at present contains 44 buildings, 5 stoplights, 3 stopsigns, and 16 other signs. The scenery is arranged in a varied way that is intended to test the kinds of information-selection that the robot must make. For example, one side of the square is a high-speed "superhighway" with a very sparse population of buildings; one block is a "housing development", whose houses all look the same except for their pastel colors.

We have made our initial "city" very simple because we feel that there is no need to introduce complex problems such as pathfinding until we have developed a sensori-motor system and an experiential representation that will allow the robot to move freely in this more restricted environment.

A.3. Dynamics of the Robot

The robot can accelerate and decelerate within the bounds of a maximum and minimum speed. The "steering wheel" setting determines the robot's angular velocity. It is permissible for the robot to attain a large angular displacement with respect to the road only if:

(1) The robot is near the end of a block, in which case it progresses onto the succeeding block determined by its angle (providing that there *is* a succeeding block at that angle - otherwise there is a crash), or

(2) The robot's forward speed is extremely small, in which case it executes a u-turn.

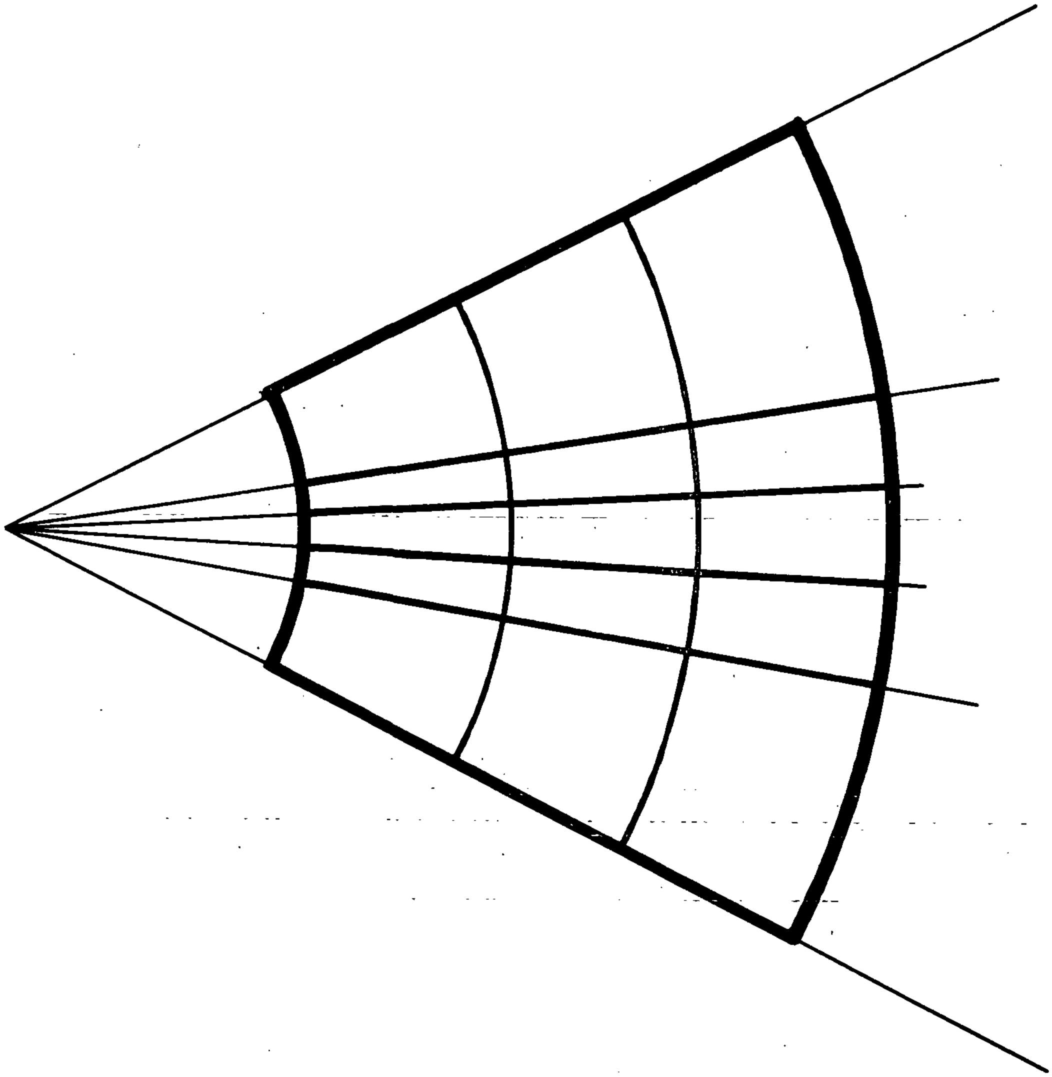
If the robot attempts to turn the steering wheel in any other circumstances, the result is a "crash", meaning a penalty input that involves a loss of speed, a waste of time, and perhaps a negative factor that may be said to simulate pain. For the purposes of steering, it is as though the streets were bordered with barrier walls: the robot can crash into the walls, but it can never actually drive off the road.

The world is also capable of enforcing a set of traffic laws, involving the running of red lights and stopsigns, speeding violations, and wrong-way travel on one-way streets (although there are no one-way streets in the current model). Also, any of these violations (plus exceeding the "safe speed") involves the risk of a crash as well as the risk of a legal penalty. There is a switch in the system which allows all of these hostile restrictions to be disabled, since in its early development the robot has troubles enough without incessantly incurring traffic tickets! However, in the full model, these traffic laws will provide an arena for experimenting with methods of inducing causal laws from direct experience.

A.4. The Visual System

The robot's only highly-developed sensory organ is its lone eye. The visual system has been made very elaborate and, hopefully, realistic, because it is the main tool in our study of the relationship between sensory experience and problem solving.

The eye is capable of seeing objects which fall within an area whose shape is shown in the figure below:



The dimensions and placement of the visual field are determined by four parameters: the radial and tangential coordinates of the center point of the field, and the radial and tangential widths of the field.

The visual field is divided both radially and tangentially into thirds, and the central angular sector is divided again into thirds, with the centermost ninth of the field representing foveal vision. The acuity of vision depends on the section of the field in which the object falls. It also depends on the total area of the visual field, such that the more tightly constricted the field, the more acute is vision throughout the field.

It is this "acuity" factor which determines which visual features of an object may be seen, because the successive features of an object have increasing thresholds for perceptability. Thus, the robot might see a building in a peripheral sub-field and perceive its HEIGHT and WIDTH properties; then, by moving the eye so as to bring the building into a more central sub-field, it might sense the COLOR and perhaps the TEXTURE properties as well. Or, it might gain greater acuity by narrowing down the field, although there is the danger that a peripheral object might be lost from view if this is not done carefully.

We feel that this ability of the visual system to pick up successive features simulates an extremely important property of all real perception, namely that in order to wrest a certain amount of information from the world, the system must (1) exert a certain amount of active effort, and (2) temporarily exclude a certain amount of other information from consideration. This is the basis of the mechanism of "focal attention", which we believe to be of utmost importance to a system that must cope with an information-rich environment.

It has been noted that buildings, roadsigns, and stoplights are treated as point-objects; they have no spatial properties. In the robot's world, the lowest level of *compound* object is the "place" -- that is, a *spatial configuration* of point-objects. As far as our investigation is concerned, it does not matter at all that buildings (which we normally think of as large) are taken as primitives with point-locations. All that matters is that the world contains some sort of primitive sensory entities, and some sort of compound entities built out of the primitive ones.

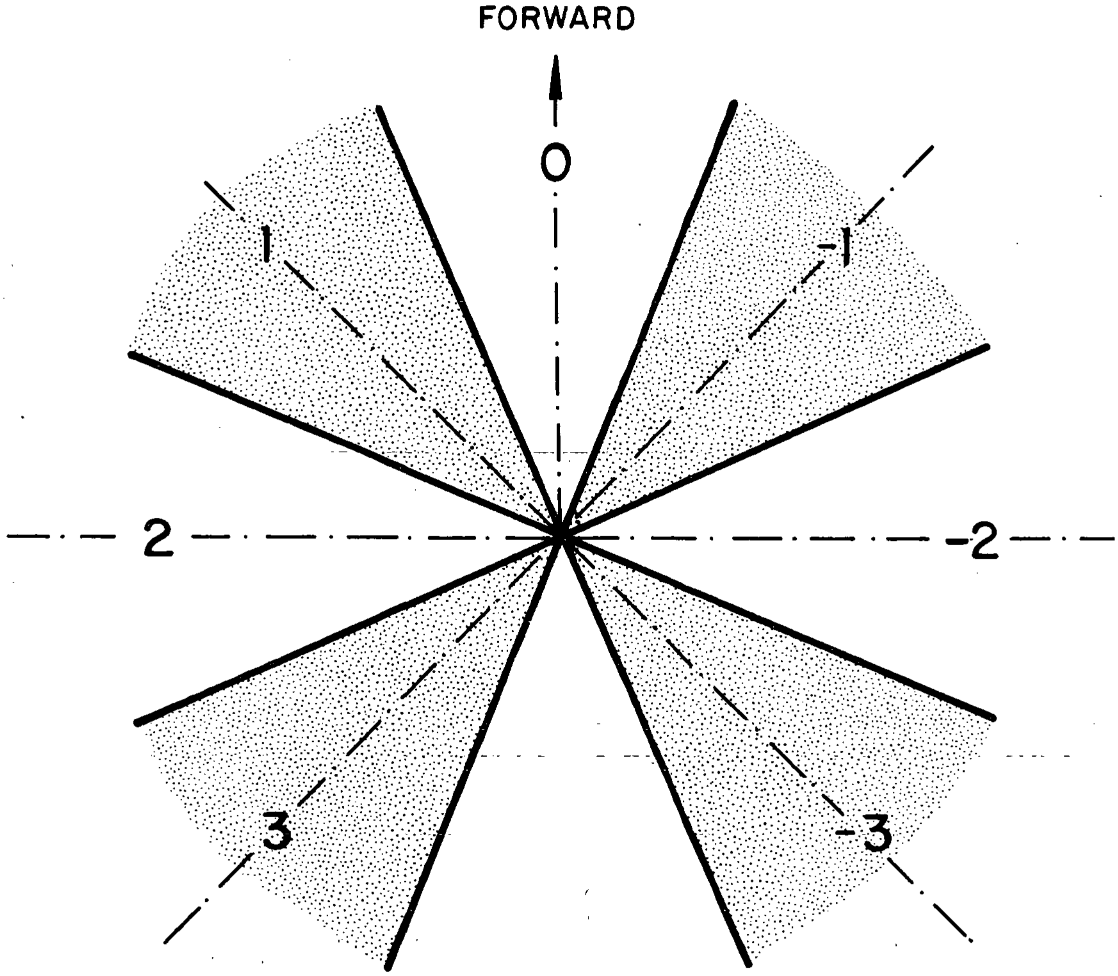
At present, the robot can potentially see any object that is on the block it is on, or on any succeeding block (providing of course that the object falls within its visual field). Also, if the robot is within 15 feet of the end of its block, it can see the intersection that follows the succeeding block. This rule is partially a simulation of the fact that we can see farther from an intersection than from the middle of a block, and partially a concession to certain trigonometric difficulties. We have not simulated the occlusion of one object by another, as this simulation would be extremely expensive and not exceptionally interesting.

We have gone to a great deal of trouble to enable the robot to see roads themselves, and to locate them within its visual field. This ability is clearly a necessity, since the only way that the robot can know where the road leads is by looking at it. On the other hand, we have not given roads any identifying visual features, because we want the robot to have to recognize them by the constellation of objects along them, and not by intrinsic properties.

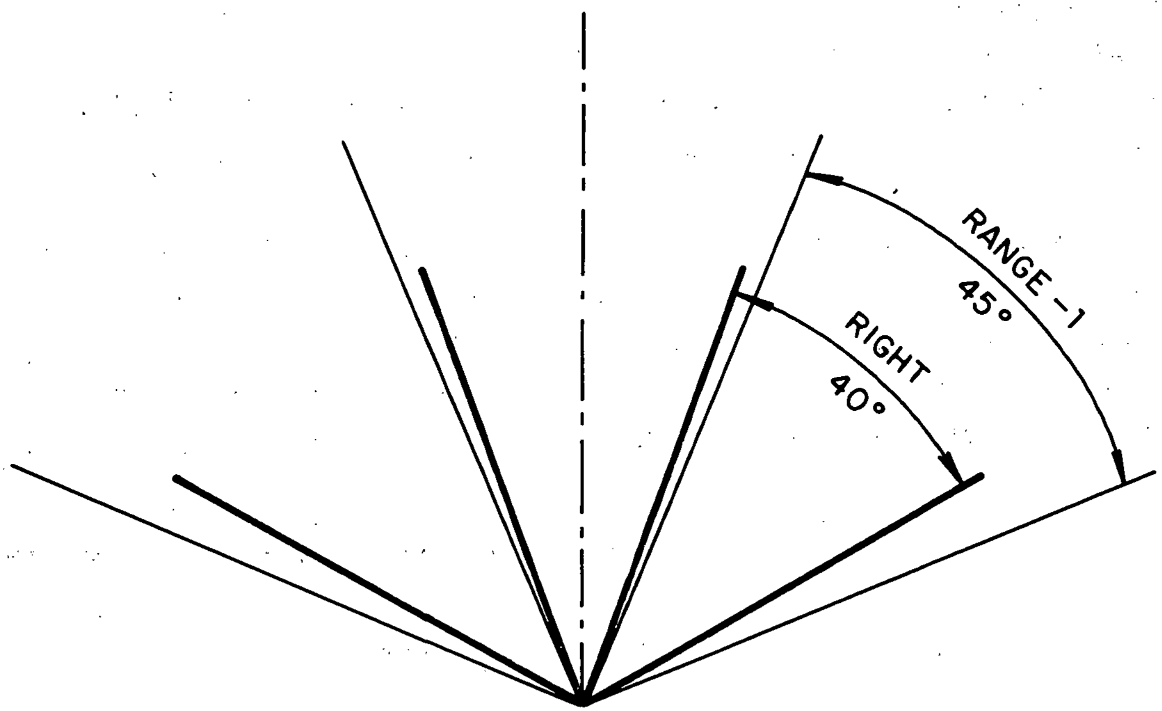
Finally, we have set up an extremely primitive motor system in which the input variable of the eye is its angular velocity (with a typical value of 225°/sec). Since time is quantized (currently in units of 0.2 sec), the act of moving the eye becomes one of setting it in motion and then monitoring its position over several instants of time (by contrast, the focusing of the eye is taken to be instantaneous). This feedback monitoring of actions is of course interesting in itself, but we are even more concerned with the interactions that arise between motor activities and cognitive operations. For example, the system can command an action and later, while the action is still in progress, decide that it is not worthwhile and abandon it. This sort of thing already happens in our present "looking around" program, but we do not yet have a good general framework for representing the time-course of actions.

A.5. The Internal Coordinate System

The robot is able to sense the angular position of its eye, independent of whatever visual sensations might be coming from the eye. This is a reasonable provision, since any vertebrate eye (or neck, for that matter) contains muscular-stretch receptors which inform the brain as to where the eye is turned. For our robot, we define seven different angular ranges, spaced 45° apart, as shown in the following figure.



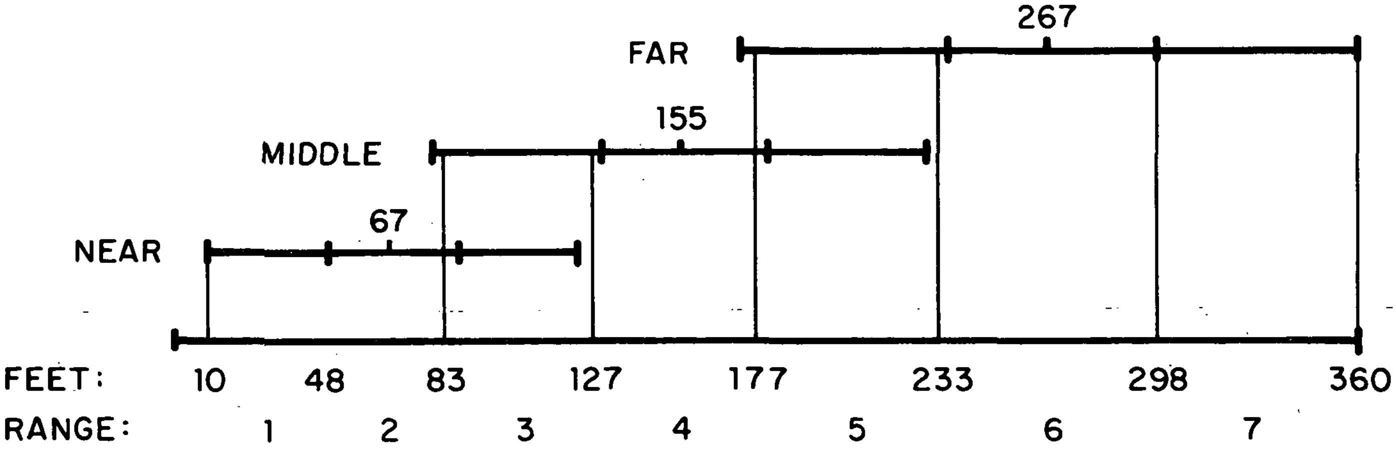
We allow the visual field to have a maximum width of 120°. When such a field is centered over one of the angular ranges, the three equal sub-sectors of the visual field nearly coincide with angular ranges (the figure below shows this with the eye centered on range 0). The inexactness of this match is intentional.



If the robot first centers on range -2, then on range 0, and then on range 2 (or in the reverse order), it will have divided whatever objects are visible into approximately their correct angular ranges. This procedure is obviously of great use in "looking around" at a scene.

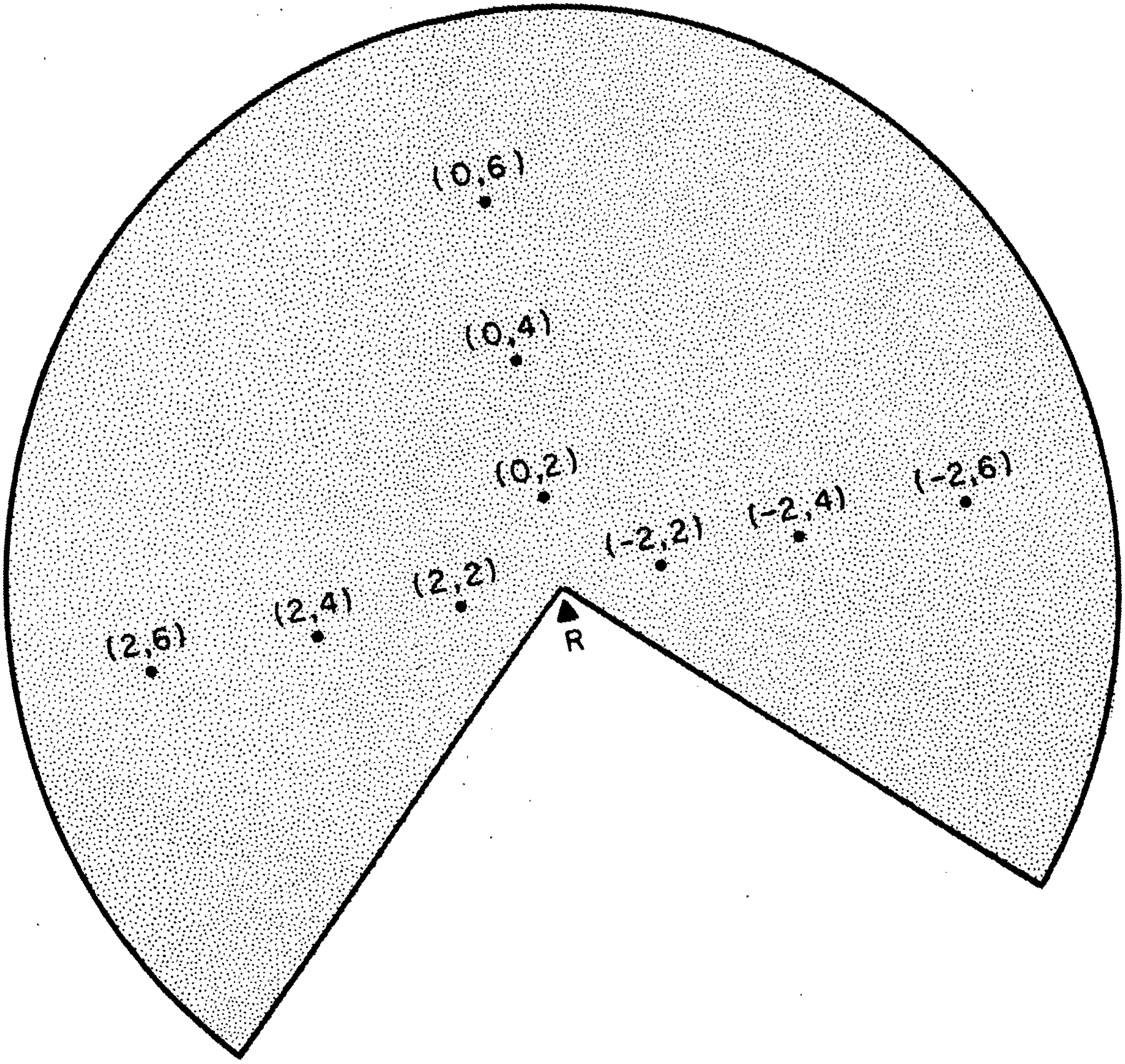
Note that the robot cannot turn its eye to the octant directly behind it. This limitation of most animals need not apply to mechanical robots, of course, but we have incorporated it in our robot because it puts an interesting constraint on the information-gathering processes. Also, this constraint results in more human-like behavior (e.g. in the scan patterns produced by the "looking around" procedure), which aids our intuition in evaluating the correctness of the robot's routines.

We have also divided the robot's radial (distance) discrimination into seven ranges. These ranges are chosen to allow the robot to cover the whole radial span of its vision with only three "focus" settings, as shown in the following figure (recall that the robot's visual field is divided radially into thirds).



Note that distance resolution is better for the nearer distances.

In summary, the robot can represent the position of an object in its own fairly inaccurate system of polar coordinates. The angular coordinate ranges in integers from -3 to +3, while the radial coordinate ranges in integers from 1 to 7. Angularly, the entire visible area can be covered by looking only at angles -2, 0, and 2 with a total field width of 120°. Radially, the entire visible area can be covered by looking only at distances 2, 4, and 6, with total field depths of 114, 150, and 186 feet respectively.



Therefore, the nine locations (-2,2), (-2,4), (-2,6), (0,2), (0,4), (0,6), (2,2), (2,4), and (2,6) take on great significance for "looking around" at all that is visible, as will be explained in Section IV.D. These locations are mapped in the figure on the previous page.

B. Discussion of the Simulation Model

The simulated city environment is made out of components which were designed to allow easy extension or combination into a new model. In fact, the system is capable of making a fairly realistic model of a real city, even one as geometrically complex as Boston. However, as indicated earlier, our simulated robot has not yet reached the stage of development where we can expect it to survive in the streets of Boston. There are many fundamental problems to be solved even in the case where the robot is not moving at all, but simply looking around at its environment and trying to record or recall its current location.

Along with being flexible, the simulation model is arbitrary in many ways. It was designed to balance off criteria of realism, computational efficiency, and theoretical validity. We tried to give precedence wherever possible to theoretical validity, by which we mean that we required the model to embody those fundamental characteristics such as richness of information, uncertainty, interaction, and commitment, which we believe to be essential to the robot problem solving situation. Within this dominating constraint, we felt that most other design decisions were arbitrary, and we did not give much attention to such choices involving inessential details.

What is crucial in such a model is that the problems which arise in terms of the simulation model be direct reflections of *general* problems involving intelligent real-world activity, rather than being specific properties of the particular simulation which have no applicability to wider contexts. It has happened again and again in simulation research that the model becomes so simplified that the ultimate system designed in terms of the model, while it "works" in the particular model situation, adds little to our general store of knowledge and understanding about the nature of intelligence. To illustrate the generality of the simulation model that we have developed in this project, we give on the following page a table of some important general characteristics of the robot problem solving situation, each of which is paired with an aspect of the simulation model which embodies that characteristic.

This list could be extended to considerable length. What it shows us is that our simulation model, however simplified and arbitrary, is still realistic in the most crucial sense, namely it provides meaty specific cases of many of the major conceptual problems involved in the study of robot problem solving. Indeed, in this first year we have not even begun to exhaust the potential for study presented by this model.

General Characteristic	Realization in Model
Interaction as source of information.	Robot gains information only through simulated sensation.
Uncertainty in information.	Robot senses are inaccurate; many objects are similar or identical; robot can never be sure that it has not wandered into unexplored territory.
Available information too rich in amount to be processed by limited sensory channels.	Robot has single eye which must scan and focus down to see detail; world is rich in objects, and they go out of view after robot passes by them.
Necessity of commitment to an action in order to gain information or solve problem.	Robot must often move eye or self to gain information, must move itself to progress toward goal locations; turning around is difficult.
"Pattern recognition" of a compound object consisting of a spatial pattern of simpler objects, received as a temporal pattern.	A "place" is a spatial arrangement of "primitive objects" (buildings, signs) that must be recognized as a temporal sequence of seen objects; the same place may be seen in different temporal sequences.
Task performed not to completion, but to criterion as subgoal for some larger purpose.	Robot needs to perform "place recognition" only well enough to enable it to follow a route; absolute place recognition is impossible, since there could always be an identical-appearing place somewhere else in the world.
Interruption of action on the basis of information gained during its performance.	Robot may begin to turn eye toward object, catch sight of a new object, and stop to look at it.
Control of a motor activity on the basis of complex sensations resulting from its performance.	Robot must slow down in order to execute a turn; its knowledge of the approaching intersection is based only on visual information.

IV. BEHAVIOR OF THE CURRENT ROBOT SYSTEM

In this section we will describe the behavioral repertoire of the present robot system, and indicate how the behaviors are produced. We have found (as have the builders of hardware robots) that even the simplest of activities requires a very considerable amount of analysis and programming before it can be performed by a system that interacts with a complex environment. Therefore, the attainments of our robot to date consist of modest, basic behavioral routines, out of which we expect that more complex behavior can eventually be built. From our present perspective, a behavior such as "visually recognizing a location" is not to be regarded as a "simple" activity which is a building block of more complex routines such as path-finding; on the contrary, we are at such a primitive stage that visual scene-recognition merely looms on the horizon of what we can understand at present, while path-finding is still beyond the horizon. (The difficulties of the scene-recognition process form the topic of our Section V.)

In the following sections, we will first introduce the notion of a primitive version of a behavior. We will then describe the robot's three primitive behaviors: tracking an object, focusing down on an object, and looking around at a scene. Then we will explain the functioning of the problem solving executive which attempts to integrate these abilities into a compound behavior, namely the "exploration" of a scene.

A. Primitive Behaviors

In order for an infant to recognize that an object maintains its identity over time, he must be able to follow it if it moves. But in order to follow an object, he must recognize that it maintains its identity, else there would be nothing to follow. These observations do not constitute a paradox, but they do indicate that the activities of recognizing an object over time and of visually following it are mutually dependent, and therefore an infant -- or a robot -- cannot possess only one of them and expect to learn the other. Since it is also impossible to learn both at once, it follows that both of these behaviors must be part of the innate equipment of a visually functioning child or robot. But it is only required that these innate behaviors be sufficient to support each other during the learning period of child or robot. Such a *minimal, innate* behavior we will call a primitive behavior.

The logic used in the preceding argument can be applied to almost any behavior of an adult organism. Since almost any activity can potentially make use of almost any other, we are forced to postulate the existence of a quintessential version of each behavior which does *not* make use of any other. The situation is very much like the definition of primitives in a mathematical system, or the provision of "system functions" in a programming system which allows recursive definitions. If you are going to build molecules, you have to have atoms.

Still speaking generally, we may point out at least five special characteristics of the primitive versions of a process, which set them apart from the more sophisticated versions that the organism later learns:

First, the primitive versions are *goal-independent*; that is, they are so straightforward that they function in the same way, regardless of why they were invoked.

Second, the primitive versions are inflexible, non-adaptive, for the same reason.

Third, because of their inflexibility, primitive versions tend to have rather narrowly-defined criteria for their success or failure. That is to say, if such a process is applied to a situation which is only slightly inappropriate for it, it may fail totally. The remarkable contextual tolerance that we are accustomed to in complex processes arises because each such process exists within a whole framework of goals, so that if one approach fails, the system can realize what is wrong and try another.

Fourth, when a primitive process *does* succeed, on the other hand, it operates very efficiently. This advantage also arises from its inflexibility, and balances against the disadvantage mentioned previously.

Fifth, these primitive processes do not make use of the system's acquired experience for their functioning. Rather, they serve as the basis for the acquisition of that experience.

As a concrete example of a primitive process, we might consider primitive object recognition. The sophisticated version of object recognition is ineffably complex (see Section V), but the primitive version is quite straightforward. A primitive object, such as a building, is seen as a list of visual features. Since the eye reports which visual subfield each feature comes from, the features visible at any time can be "clumped" according to the crude criterion of coming from the same visual subfield. For the robot, at least, this clumping operation is the first step on the road to its internal notion of "object".

One of the primary characteristics of an object is that it maintains its identity over time. In order to recognize this, the robot requires a process which identifies a clump appearing at time$_1$ with a similar clump that appeared at time$_0$. Now, there are several disasters that might befall a clump between time$_0$ and time$_1$. It might be displaced in the visual field; it might be split into two or more clumps (if it actually consisted of features of several distinct objects, which happened to fall in different subfields in the new view); it might grow (through the addition of new objects or through the more acute perception of objects already represented in the clump); it might shrink (if it wanders into a less acute subfield); or it might disappear from the field of view entirely. At the moment, we are using a matching function that computes the goodness-of-match between two clumps as the number of features that they have in common, minus the number of features that either one has but the other lacks. Certainly this simple measure is susceptible to being fooled in nastily-conceived situations, but in the actual performance of the program it has yet to fail. This is mainly because in its applications so far, the object has usually fallen near the fovea, and therefore many features of it were seen, so that it could fairly readily be distinguished from nearby objects. We feel that this is a valid reason for the matching to succeed. On the other hand, we would not be too upset if the matcher should confuse two nearly-identical nearby objects -- this too could be valid. (We tried to get it to confuse our three absolutely-identical nearby apartment buildings, but it refused to fall into the trap.)

Clearly this matching routine is the most primitive case of "recognizing" an object (note that the "objects" it recognizes are themselves spatially primitive). In particular, the above routine is goal-independent, whereas the experientially-based recognition of a compound object (a "place") is directly affected by its subservience to some higher goal such as following a route. The primitive version merely computes a simple measure, which can be fooled by a variety of special cases. Yet in the general case, it works efficiently and well. It does not make use of any acquired knowledge about objects, but rather serves as the basis for the acquisition of such knowledge.

The tracking, focusing down, and looking around processes to be described in the following three sections are all primitive versions of more elaborate, goal-dependent processes.

B. Tracking

The ability to track a moving object, or to track a stationary object while moving, is clearly an early prerequisite of visual perception. This facility appears very early even in human infants, and it is hard to imagine how tracking could be a learned behavior. We therefore chose tracking as the first of a series of "innate" primitive abilities that must be supplied to our robot.

During visual tracking, the eyes may be moved on the basis of position (saccadic tracking) or velocity (smooth tracking). In human tracking, the eyes initially make several saccades (jumps), during which the velocity of the object is presumably being estimated; then they switch to smooth tracking. For our robot, we have so far programmed only saccadic tracking, but we could easily have gone on to smooth tracking, as discussed below.

Feedback in Saccadic Tracking

One of our motivations in studying tracking is that it is a process which by its very definition requires an integrated program of activity and sensory feedback to be carried out over an extended period of time. We feel that such integration is one of the most important characteristics that distinguish a robot system from an "abstract" problem solver.

Stated in programming terms, tracking cannot simply be executed in the "one-shot" manner of an ordinary LISP function, so we had to create some sort of executive structure to maintain the tracking operation over time. Our initial executive was as simple as could be: We designated a small "goal memory" that would remember what the system was trying to do (in this case, simply the operator TRACK and a representation of the object that was being tracked), and at each quantum of time the executive performed whatever operations were retained in the goal memory. This was a very humble beginning, but it has already been much expanded, as described in Section IV.E.

The basic operation in saccadic tracking is to notice that the tracked object has moved out of the center of the visual field, and to reposition the eye accordingly. Now, in general this repositioning can itself be a feedback process. But eventually all feedback processes must be composed out of non-feedback (endogenous) primitives; otherwise there would be a logical regress, and nothing would ever get done. We chose to make the repositioning operation be an endogenous process, by making sure that the eye is always repositioned just enough to bring the object back into the center of the visual field. Since the field is divided into thirds both tangentially and radially, this means that repositioning simply requires moving the focus point by a third of the width of the field in the direction of the peripheral subfield that the object has moved into.

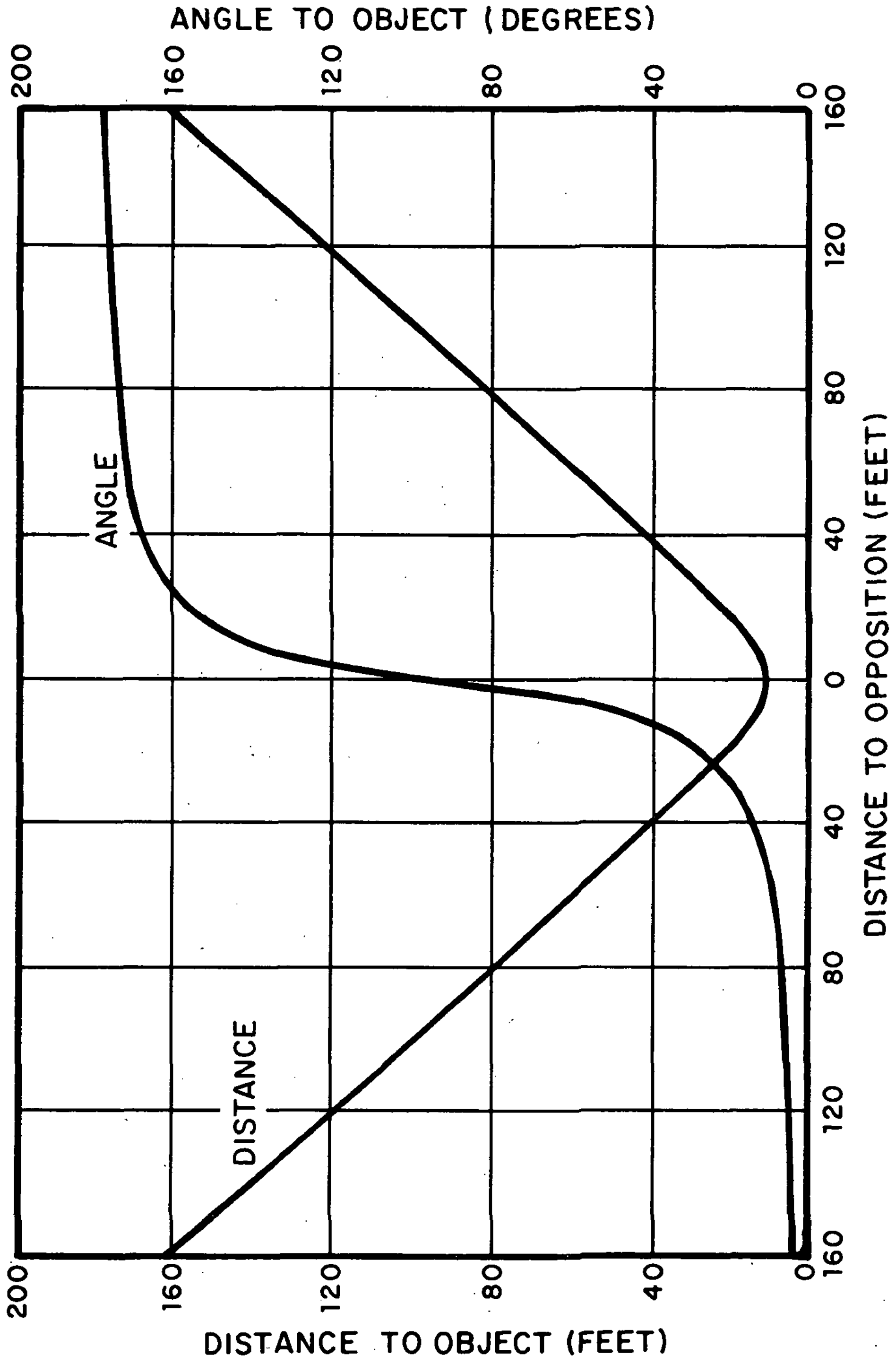
Our current solutions to the problems of feedback, while very simple, have begun to touch on some weighty issues. We saw that the use of feedback demands that the system retain, over time, some sort of representation of "what it is doing". This is already a great departure away from algorithmic programming and toward a "goal-oriented" system. We also touched on the notion of a hierarchy of feedback processes (in that tracking uses repositioning as a subprocess, but the latter may also be extended in time). This is the forerunner of many problems of *coordination* that we will not long be able to (nor want to) avoid.

Tracking by Velocity

Our program succeeded well in tracking an object while moving down the road at 25 feet per second, given a visual field 60° wide, and passing within 10 feet of the object at its nearest point. With other values of these parameters the object could move out of view in one quantum, in which case primitive tracking would fail.

When we graphed the focus position of the eye against a plot of the actual position of the object, we were surprised to find that the curve for the actual position interested us more than the performance of the saccadic tracker. The following figure shows two curves, one for the actual angular position and one for the actual radial position of the object as a function of road distance to opposition with the object (i.e. as a function of time, if the robot is moving with constant velocity). Strangely enough, the curve for angular position consists mainly of three sections of nearly-constant slope. Stranger still, the curve for radial position could also be approximated by three line segments that join at the same points as those which would approximate the other curve (about 15 feet on either side of the opposition point). These facts clearly suggest a program which tracks a passing object in three phases (not counting an initial saccadic lock-on phase), where each phase is characterized by a constant *rate of change* for both angle and radius of the eye's focus position. What is surprising is that this conclusion arises merely from the geometry of the situation, independent of any properties of the perceptual system! So one might expect to find a three-phase tracking procedure in living systems as well as in robots.

The principal reason that we have not yet programmed velocity-based tracking is that we became more interested in other problems. Also, the three-phase tracking process demands more sophisticated use of feedback than saccadic tracking does (especially at the points where a transition between phases is indicated), and we wanted to study the constraints on the executive in another context, rather than designing it solely around the tracking procedure.


ANGLE TO OBJECT (DEGREES)
200
160
120
80
40
0
DISTANCE TO OPPOSITION (FEET)
160
120
80
40
0
40
80
120
160
DISTANCE TO OBJECT (FEET)
200
160
120
80
40
0
ANGLE
DISTANCE

C. Focusing Down

Once the robot can, through tracking, keep an object centered in its visual field, the next problem is to focus down on the object so as to pick up more and more of its detail. The details so obtained are used to build up a stored representation of the object. When, later, the object is to be visually recognized, the focusing-down procedure is reapplied to it, and the newly seen features are matched against those already stored.

Again, we note that the objects involved here are themselves primitive, so that the details obtained by primitive focusing down are only visual features. The more general version of focusing down is applicable to compound objects, and is capable of detecting sub-objects and their relationships. Also, we should repeat that for convenience we will treat focusing down as a function of the eye, whereas in actuality the mechanism of "focal attention" has no known physical locus.

The process of primitive focusing down is actually a very simple one of matching, the only complication being that there are three memory structures involved. There is in Long-Term Memory (LTM) a representation of some object previously seen; in Short-Term Memory (STM) the system builds a representation of the object it is currently focusing down on; and in Immediate Sensory Memory (ISM) is found the set of features that are actually being seen at any given moment. In the best of cases, a subset of the features in ISM will map into the features specified in the representation in STM, which will likewise map into the features recorded in LTM. The system extends this mapping to include more and more features by narrowing the width and the depth of the visual field in alternation, so as to take in more features of fewer objects.

If no clump of features in ISM matches the representation in STM, then the object has inadvertantly been lost in the process of narrowing the visual field. If the ISM set matches but does not exceed that in STM, then more focusing down is required to assure identification of the object. If the ISM set matches and exceeds that in STM, then the excess is matched against the representation in LTM. If it still matches, then features are copied from LTM to STM; in other words, the robot assumes that it is seeing features of a known object. If the ISM set matches and exceeds the kernels in LTM, then the LTM representation itself is added to; this means that the object is being seen in more detail than it ever was before, and the detail is recorded. Finally, if the ISM kernels turn out to be inconsistent with those in LTM, then the excess kernels are entered into the STM representation, and that is copied over into LTM; a new object has been seen, and a new LTM entry is created.

In summary: The robot attempts to maximize the match between what it is currently seeing (in Immediate Sensory Memory) and what it once saw in the past (held in Long-Term Memory). In order to mediate this matching process over several quanta of time (and hence over several refocusings of the eye), a temporary representation in Short-Term Memory is built up. We should mention that the sort of "Long-Term Memory" representation we are using at present is extremely impoverished: it does not store the spatial relationships among objects, and hence is incapable of representing a "scene". We simply are not yet far enough along to have considered the problems of scene representation.

It should be noted that the storage of previously unseen features and the recognition of an object in terms of previously seen features are treated as parts of one and the same process. The robot *must* recognize an object before it can add new details to its internal representation; conversely, the recognition process often produces novel details as a by-product, and it would be senseless to throw them away. Here is an instance of a simple yet important principle in animal behavior: take heed of information which is fortuitously discovered. What makes this principle interesting is that it violates the strict goal-directedness that we often like to impute to animal behavior. That is, an animal (including a human, and hopefully also a robot) will remember information merely because it arrives free of charge, even though it was not sought and may not be connected to any immediate goal. Indeed, an organism would be in a difficult logical bind if it had to explicitly seek out information that was unknown to it!

Perhaps the most interesting feature of the focusing-down process is that it has no logical termination. The proper time to stop focusing down is when a certain amount of focusing effort has been expended, with no compensating results obtained. These concepts of "effort" and "results" -- combining to give a notion of "progress" -- are primary elements of the executive decision procedure which is responsible for choosing among competing processes. Therefore, we will defer their discussion until the section on the executive (Section IV.E). Suffice it to say that many activities besides focusing down have this same property, that they are terminated not when they reach some logical completion, but simply when they cease to be sufficiently productive.

D. Looking Around

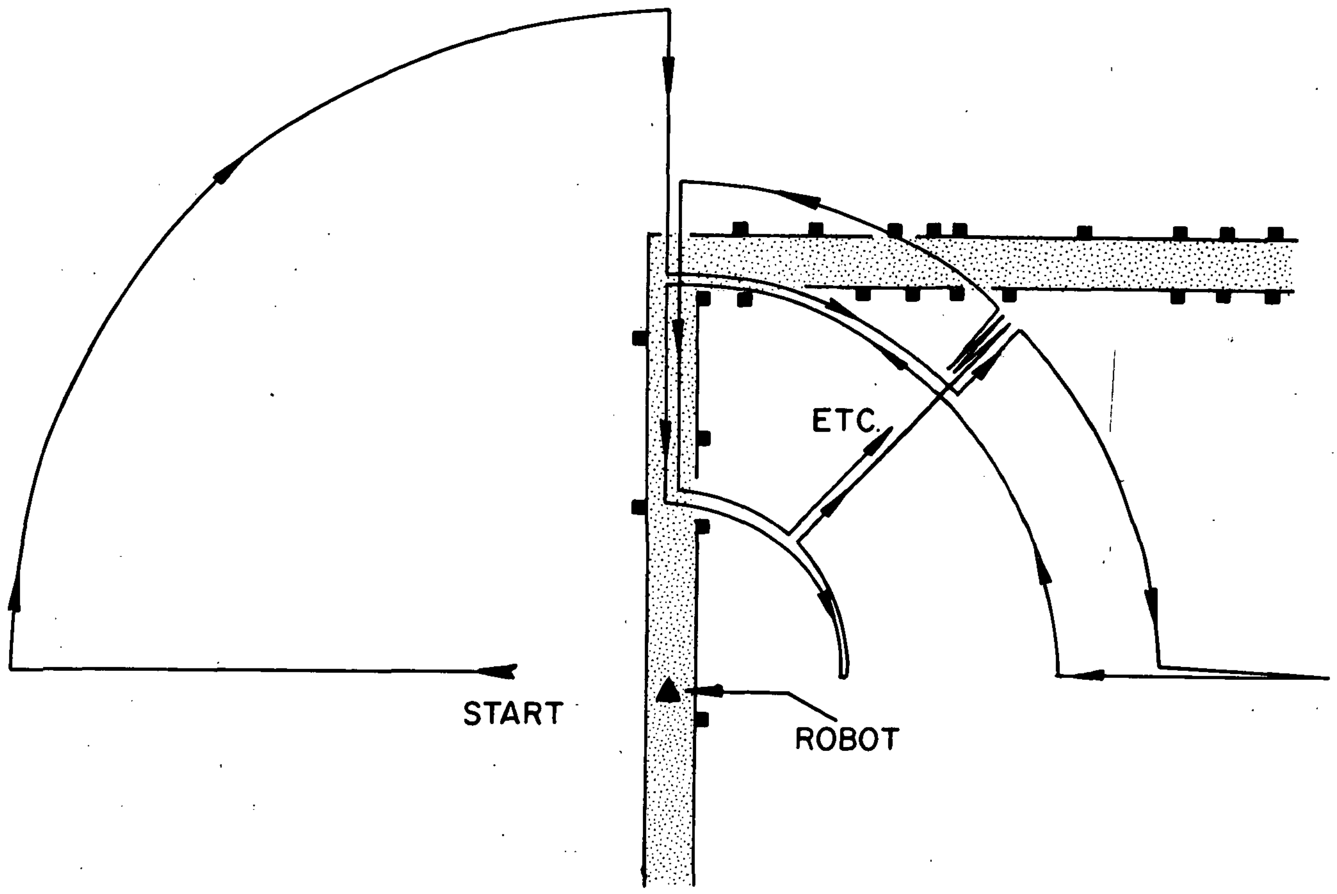
In order to gather information about its environment, the robot must look around at the scenery that surrounds it. Again we find it important to distinguish two levels of the looking-around process: *Experience-based* looking around makes use of an internal model of what surrounds the robot, and may participate in higher goals such as remembering or recognizing a route; it is at least partially a process of verifying (recognizing) that the robot is in such-and-such a hypothesized place. By contrast, *primitive* looking around requires only a temporary internal model of the scene, it does not involve the verification of a hypothesis, and it need not serve any definite higher goal other than the exploration of the robot's surroundings. In this section we will concern ourselves solely with primitive looking around.

Basically, the looking-around process is a solution to the problem of focusing down on several recently-discovered objects "all at once". The problem would be trivial except that there is only one eye, and so the robot's executive must time-share the focusing-down subprocesses in an efficient manner. A great deal of our effort has gone into making explicit the notion of "an efficient manner" in this context, as will become apparent in the next section.

A small complication arises because the robot must have some motivation to first look in a direction which is not known to contain any objects. For this reason, along with the focusing-down subprocesses, the robot maintains subprocesses which, when run, turn the eye to one of the nine major coordinate points shown in the figure on p.23. By intermittently running these positional subprocesses, the robot makes sure that it does not miss any of the scenery that lies around it.

Thus, the looking-around process itself consists in nothing more than the temporal interleaving of a set of information-gathering subprocesses. These subprocesses are of two types: (1) *focusing-down processes*, one for each object that the robot has noticed in the scene, which attempt to see more and more features of the objects, and (2) *positional processes*, one for each of the nine main coordinate points in the robot's surroundings, which turn the eye towards those points in the hope that some new objects will be seen there. The heart of the looking around process is the executive which is responsible for interleaving these subprocesses in an efficient manner. The executive will be discussed in detail in the next section.

Without further ado, we present an example of the behavior of the current looking-around program, in the figure below:



In this figure, we have superimposed the eye's scan-path over a map of the territory in front of the robot. The scan-path is the locus of the center point of the robot's visual field.

In this example, the robot was standing still, facing a right-angled street corner. Objects along the roadside are shown as boxes in the figure. The robot's eye was greatly attracted by the cluster of objects just to the right of center (the cluster to the far right was totally invisible to it, owing to limitations in the trigonometry routines which compute what the eye can see). The scan-path clearly displays the program's attempt to find a balance between moving to unseen areas so as to find new objects, and fixating on the ones it has already found so as to get a better look at them. The figure of course fails to show the small local eye movements which were involved in successively focusing-down on one or several objects within each fixation.

It can be said that the scan-path shown above looks very reasonable considering the distribution of objects around the robot, and it seems to have a qualitative similarity to actual scan-paths that have been recorded in eye-movement studies of human subjects. It would be very difficult to state more rigorous criteria for knowing when this program was behaving "correctly". Indeed, one of the most interesting characteristics of the looking-around process is that it appears to be "ill-defined", in the sense that there is no obvious algorithm for doing it, nor is there an obvious algorithm for determining when it has been successfully done. Ultimately, we expect that the performance of any process will be evaluated, and if necessary corrected, by a higher-level process that makes use of it (e.g., in this case, a process that is trying to recognize where the robot is). But for the moment, our inability to assess correctness is a very unpleasant frustration of scientific rigor -- not to mention a source of headaches for the robot designer who would like to know when his program is finally working correctly!

E. The Executive

In this section we will describe the operation of the problem solving executive, which is the heart of our current simulation system. The present executive is by no means a general problem solver; rather, it is designed to control the looking-around process described in the previous section. We have not yet integrated tracking into the looking-around process, but we believe that this would be fairly easy, and that it would not have much effect on the structure of the executive.

In the first subsection below, we state the fundamental problems that the executive must cope with. In the second subsection, we list many of the particular factors that must go into the decisions made by the executive. In the third subsection, we sketch the major steps in the executive's actual operation.

E.1. Basic Notions Concerning the Executive

The function of the executive is to decide, at each moment, what the robot should be doing. It must make a decision which, if not the best possible, is at least a good one given the information available at the moment. Note that we have said "at the moment", for many of the executive's problems arise from the way in which the robot's circumstances and information vary from moment to moment. Implicit in the responsibility of the executive is a dynamic adaptiveness which reacts properly to new circumstances as they arise. In particular, it is important to note that the performance of any one action may affect the prospects for any other action -- even including the desirability of its own continuation.

The adaptive alertness of the executive must be counterbalanced by an ability to coordinate the robot's activities so as to produce useful results. One of the simplest aspects of such coordination is a sort of inertia which dampens the adaptiveness of the executive's response. Without this inertia, we have seen behavior (in our simulations) in which the robot's attention oscillates from one object to another, without ever being able to concentrate on any one object long enough to see it clearly. So, the executive must be distractable enough to respond to important unexpected conditions, yet single-minded enough to get something accomplished.

In order to discuss the kind of coordination over time that the executive must achieve, we should think of the executive as manipulating whole processes, rather than individual actions. Thus, although the basic duty of the executive is to select the proper action at each moment, this cannot be done efficiently except in the context of the actions that have gone before, and the actions that are anticipated in the future -- in other words, in the context of a *process* that the executive wishes to perform. Processes are the units in terms of which execution, interruption, resumption, progress, prediction, coordination and induction are defined. It is still an open theoretical problem to specify all that a process is, and in the course of our programming we have had several occasions to realize how tricky this notion can be.

So, the executive is seen as an agent which selects and organizes a cadre of individual processes, all in response to the flux of incoming information about the environment. It is especially important to realize that all of the decisions made by the executive are dependent on the *particular fine details* of a momentary situation. The executive is not like a philosopher building theoretical constructs in an informational vacuum; rather, it is fanatically attentive to the flux of detailed information that is coming in from the world around it. It uses only a few general decision rules to organize all of this data once it is collected. We believe that there is no way of understanding intelligent behavior unless we appreciate how closely it is tied in with the intricate details of the environment. This is where the complexity lies, and not in the mechanisms of intelligence themselves. Certainly in the case of our robot's executive, much of the design problem lay simply in keeping track of all of the information that had to be weighed in the executive's decisions.

E.2. Factors in Executive Decision-Making

Below we will briefly describe the various sorts of information that our executive takes into account in determining a good course of action for the robot. We will limit ourselves to discussing these factors individually, in isolation from each other. The reason for this is simply that the way in which they are combined cannot be described in much simpler terms than the computer program which actually combines them, and we doubt that the reader would benefit from a discussion of the program in all its detail. Moreover, the factors themselves each provide a little morsel of food for theoretical thought, whereas the devices by which they are combined cannot be said to have much theoretical interest.

a. Factors relating to individual processes

When the executive comes to consider each individual process, it tries to estimate an *expected payoff* from allowing that process to run one more step. In this section we will enumerate the factors that go into this computation. It is worth keeping in mind that that is indeed an *estimation* rather than a certain prediction, which implies among other things that the system could eventually be made to improve its estimates by comparing them with the payoffs that were actually received. (Our current system does not do this.)

(i) Steps: We assume that there is, for each process at each moment, such a thing as "the next step". There are some problems with this (as we will see), but for the moment assume that such a concept is available to the executive.

(ii) Drive Strength: Each process is assumed to have a time-varying motivational component, attributable to nothing more than the robot's need to perform that process. In the case of focusing-down processes, this "drive strength" is assumed to increase with time to a maximum, and then decrease, so that an object which goes unattended for long enough will eventually be forgotten about. The processes which cause the eye to move to new territory, on the other hand, show no such decrease (else the robot might eventually fall into permanent inactivity); their drive strength simply increases linearly with time.

(iii) Salience: Some objects are inherently more attractive, or salient than others. For our robot, as for humans, stoplights attract the visual attention considerably more than do buildings or signs.

(iv) Progress: Since the executive must estimate a future payoff, one of the most important quantities for it to consider is the time-derivative of payoff, which we think of as the "rate of progress" of the process. Besides playing a major role in estimation, a progress measure provides the most natural solution to the problem of processes which have no logical termination: they are simply run until their payoff yield falls below that of competing processes.

In the particular case of a focusing-down process, the payoff is in terms of "information" about the object that is under scrutiny. For a given set of visual features, the information value can be equated to the logarithm of the reciprocal of the number of objects in the world consistent with that set of features. We have provided our robot with a table from which it can in fact determine the number of objects consistent with any set of features; needless to say, this is a crude simulation of another sort of estimate that really should be made on the basis of the robot's statistical experience with the world. "Progress", then, is a bow-shaped function of the amount of information accumulated: slow at the beginning and end of the inspection of an object, and rapid in the middle. (This curve is also supplied artificially in our simulation, since we cannot take actual derivatives in our discrete-time system.)

Notice, by the way, that it is possible to compute the precise quantity of recognitional certainty that is contributed by each visual feature as it is perceived. This measure can be called the *criteriality* of each feature to the recognition of the object. One way of regarding the predictive application of the progress measure is feature-by-feature: that is, if the next feature is expected (on the basis of previous experiences) to be one of high criteriality, the estimated payoff for the process is high, and it should be given high priority in the competition for execution. Low-criteriality features, on the other hand, may not be worth looking for.

(v) Effort: One way to maximize the payoff of an action is to minimize the effort that the action requires. In our simulation, actions are attributed with varying amounts of effort requirement (measured in arbitrary units), according to reasonable guidelines; for instance, it requires more effort to move the eye a long distance than a short one. Among other things, this attention to effort causes the robot to spend a considerable amount of time looking at objects which are near to each other, rather than scanning wildly around to look at objects strictly in order of their interestingness. We believe that people and animals behave in the same way, for the same reason.

(vi) Confidence: The estimates made by the executive will vary in their accuracy, because some actions have more predictable consequences than others, and because there may be more information available about some processes than others. Over the course of its experience, the system can gather statistics as to the accuracy of its predictions, and then use them to estimate the accuracy of future predictions. Since our current simulation does not do any statistical learning, we have had to supply these "confidence" factors in advance.

b. Factors relating to the coordination of several processes

The factors mentioned in the preceding section are combined to yield an estimated payoff for the next step of each available process. But the choice of the actual next step to perform is still complicated by the need to coordinate the interaction among individual processes.

(i) Combined Payoffs: It may happen that several processes require the same action as their next step, in which case the estimated payoffs for each process are combined. Thus it can happen that the robot will choose to turn its eye toward, say, three mildly-interesting buildings which are close together, rather than toward a single, isolated building which itself is more interesting than any of the three. What we have here is really a first inkling of the notion of sub-object: the three buildings can be said to form a complex or sub-scene which has its own identity, and which can be considered as a single object that is more interesting than the isolated building. We believe that the hierarchical structure that the human mind is known to impose on scenes arises precisely because a hierarchical organization simplifies the executive coordination of such processes as looking around at a scene.

(ii) Variations in Individual Factors: If we consider the factors used to evaluate individual processes, listed in the last section, we see that some of them change with the mere passage of time: certainly drive strength and progress do. These factors can be even more strongly disrupted by unexpected events in the external world, or by the activities of the robot itself. To give a simple example, if the eye moves away from the location of a particular building, then the "next step" in the focusing-down process for that building changes: the "next step" is now to restore the eye position. It is apparent that the executive must continually monitor the condition of all available processes, since even the next step required by a process is subject to unexpected change.

In our simulation, this "continual monitoring" is performed by incessant serial evaluation. One cannot help but feel that processes ought to be independent entities, capable of monitoring *themselves*. This brings us to the "demon" conception discussed in Sections II.B and VI.G.

(iii) New Processes: The executive is also responsible for realizing when it has received input information that cannot be assimilated by any existing process, and for creating a new process to deal with this information. In our model, a new process is created whenever a visual feature is seen that is not claimed by any focusing-down process.

(iv) Single-mindedness: We have already mentioned the need for a sort of cognitive inertia or damping to prevent the executive from oscillating among several closely-competing processes. One way in which this is enforced is by charging a "changeover overhead" effort to each process which would seek to displace the current one. Another is by giving the current process first crack at the visual input which was generated by executing its own previous step. This latter privilege is surprisingly important in allowing the focusing-down process to concentrate stably on a single object. Finally, we should mention that the tenacity with which the current process is pursued is a function of its rate of progress; when progress falls off, the time has come to turn to some other process.

(v) Interruption: In our system, the executive interrupts a process when it finds another process to be sufficiently meritorious. In order to be interruptable, each process must of course carry with it enough information to allow it to resume later on. As we have pointed out, the resumption of a process may require some explicit actions, such as moving the eye back to where it was when the process was interrupted. In the most general case, it may be difficult to tell which preconditions for a process have changed since it was interrupted, which of its results have come undone, and so it may be hard to know just how to resume.

Here we have an instance of the "frame problem" (anything may change the conditions for anything else), which has no general solution. It must be solved for the particular case, as we have solved it. In our particular case, the focusing-down process records the parameters of the eye when it is interrupted, and restores them if necessary when it resumes.

(vi) Temporal Integration: In our simulation, it is also possible for an action to engender input information which leads to its *own* interruption. For example, if the eye is moved through a substantial distance, it may well catch sight of some new object, for which a new focusing-down process will be created, and this focusing-down process may supercede the positional process which had originally invoked the eye motion (indeed, the very purpose of the positional processes is to turn up new objects in this manner). Note that the eye motion may be interrupted before it has carried through to its full extent. This sort of interruption in mid-step can take place when there is a discrepancy between the time-scales of two activities, in this case, between the time required to move the eye a long distance and the time required to notice a new object. In other words, there is a coordination problem when a "step" of one process takes longer than a "step" of some other. In such cases, the interruption of the first process in mid-step can present the executive with severe difficulties in keeping track of what is going on. In our program we have solved these problems by *ad hoc* devices. We still are just beginning to have a feel for the more sophisticated problems that may arise in the temporal integration of interacting processes.

E.3. The Executive Cycle

Apart from the details noted in the preceding section, the basic structure of the executive's operation is very simple:

1. New sensory input is received, and each process is given a chance to propose a next step, and to evaluate the various measures for that next step.

2. Any visual features that were unclaimed in the previous step are used as the basis for forming new focusing-down processes.

3. Measures are combined for all processes having the same next step.

4. The next step with the highest rating is chosen.

5. If the next step is consistent with the process that proposed the last step, it is simply readied for execution. Otherwise, a comparison is made, taking into account the rate of progress of the previously-selected process, and the "overhead" effort of switching to a new one. As a result, either the old process is continued, or the new candidate is enstated as the current process.

6. The selected next step is executed. The program for that step bears full responsibility for assuring that its process is resumed correctly. (Probably this responsibility should be given at least in part to the executive.)

7. Physics is invoked, and new sensory input is computed.

8. The executive begins again at Step 1.

From the simplicity of this framework, it should be clear that the executive really embodies only one general principle: "At each moment, do the best thing you can." Aside from this, all is specifics, as outlined in the previous section. One great reward of the present study is that we have been able to isolate these particular factors that go into the executive decision, and include them in our program in as *explicit* a manner as possible. In most computer programs, even "intelligent" ones, it is the *programmer* and not the program who keeps track of goals, estimates progress, effort, and confidence measures, plans out alternative actions, anticipates the need for interrupts, and so on; all of these considerations become implicit in the structure of the program.

To the extent that we have been able to represent the bases for executive decision-making explicitly, we have in a limited but significant way allowed the system to *program itself* in response to its interactions with an unpredictable environment. We believe that this is a first step toward a system which can exercise its own active intelligence in responding to problems that confront it, rather than doggedly applying a few clever techniques supplied by its human programmer.

V. THE REPRESENTATION AND RECOGNITION OF COMPOUND OBJECTS

In our work on the robot simulation, one major problem has emerged as a natural aiming point: the question of how the robot should internally represent a scene so that it can later recognize the same scene, and the inseparable question of how the recognition process should operate, given a scene and an internal representation.

To be more specific, the "primitive" objects in our simulated world are buildings, signs, and stoplights. They are primitive because the eye, if properly set, can gather in all available information about the object (i.e. all its features) at one time, without performing any sequence of actions to seek them out. By contrast, a "compound" object is one that requires some sort of active sensory operation (such as eye movements), or at least some time-course of sensory predictions (such as, in music, following a single voice in counterpoint), in order to be fully perceived. In the robot's world, the compound objects are "places", i.e. spatial constellations of primitive objects.

The "recognition" of a compound object is the process of matching a current sensory situation to a stored characterization. The "representation" of the compound object is that stored characterization which is used in its recognition. Since the whole recognition process is viewed as black-box behavior, there is no theoretical basis for drawing the line between a static "structural" component (the representation) and a dynamic "process" component (the recognition procedure itself). We make this distinction only because it seems to produce for us the most comprehensible description of a very complex behavior.

The place recognition problem for the robot is a perfect miniature model for the "pattern recognition" problem that has dominated much thinking in the fields of artificial intelligence and psychology. Obviously, we can view this problem only as a long-term goal of our investigation. Still, we feel that we have already gained some insights into the place-recognition problem from working on our simulation, and we will set forth these ideas in the present section.

A. Relativity of Definition

The first problem in understanding place-recognition is that a "place" is not well-defined, either in common usage or in terms of the model. Places need not have definite boundaries where one leaves off and the next begins. We will therefore have to assume that the criteria which determine the limits of the *representation* of a place are not necessarily present in the physical structure of the world. Then they must come from within the cognitive system itself. Evidently they derive from *pragmatic* constraints, having to do with improving the ability to attain some survival goal of the system, such as finding its way around. That is to say, a "place" is delimited by the sufficiency of its representation to enable some *higher* goal, such as recognizing where to turn in order to get from point A to point B. We may express this relativity as follows:

The representation and recognition of places, and of compound objects in general, are not defined except in terms of higher pragmatic goals of the organism.

Since recognition is always a sub-goal of some other process, the amount of effort allotted to it depends on the economics of attaining the larger goal. For example, the robot might find that it could get a much better look at a certain place if it were to stop and back up, but the act of stopping and backing up would be antithetical to the larger goal of getting from point A to point B.

B. Graded Nature of Recognition

Another difficulty in the recognition process is that it can never be an all-or-nothing affair, even if the robot were to possess error-free sensory systems. The reason is simply that the robot, in concluding "this is Place P", cannot logically exclude the possibility that it is actually in some *other* place which merely *looks* like Place P, so far as its internal representation of Place P goes. This is true so long as the robot's world is effectively unbounded (i.e. the robot cannot explore all of it in its lifetime); it can also be a problem in a bounded world if the robot's place-representations are habitually underspecified. A converse problem also exists in any world that is subject to change: the robot may actually *be* at Place P, but it may look different than it did when the representation was created, either because it has changed or because the robot is seeing it from a different viewing angle. Under any of these conditions, all of which pertain to any realistic situation, it is impossible for the recognition process to confirm or reject a hypothesis with absolute certainty.

Object-recognition is never a binary decision process, but is always one of computing a certainty measure that is less than absolute certainty.

C. Information Density

In order to see what this certainty-of-recognition measure depends on, let us imagine that one of the characteristics of Place P is that there is a stopsign there. Now, if it happened that there were only one stopsign in the world, the sighting of a stopsign would uniquely identify Place P. On the other hand, if there were a stopsign at every intersection, the sighting of a stopsign would provide no recognition information at all. Of course, we have just pointed out that the robot can never know the exact statistics of the occurrence of objects in its effectively-unbounded world, but in a homogeneous environment it can gain a good approximate knowledge of those statistics. These statistics, then, define a measure of information density over the possible features of the world: density accumulates around the rarer features or groups of features.

The informativeness of an observation, or of the inclusion of an observation in the representation of an object, is dependent on the statistics of the occurrence of that observation.

Thus, the efficient representation and recognition of places depends on the collection of a very considerable body of statistical information, so that only the most informative features may be recorded and looked for. All of this must seem obvious to those who think of information in Shannon terms, but workers in artificial intelligence have too long gone on simply reckoning any predicate-clause as "information", without recognizing that crucial *degrees* of informativeness exist.

D. Adapting the Sequence of Recognition

Since the primitive objects that make up a place vary in their informational value, it would seem that the most efficient procedure for recognizing a compound object would be first to look for its most characteristic sub-object, then to look for the second-most characteristic one, then the third, et cetera. This is not necessarily true, for two reasons. First of all, looking for each of the primitive objects requires intervening motor activities, and the expense of such actions (in time and effort) varies with the physical proximity of the objects. Thus, if the first, third, and fifth most characteristic sub-objects were on one side of the street, and the second, fourth, and sixth most characteristic on the other, it would certainly be inefficient to look for them in serial order. Notice also that a *grouping* of sub-objects may be very distinctive, even if its component primitive objects are not. Such a grouping forms a compound object of its own, and this compound object is a sub-object of the larger compound object.

The second reason for scanning in an order different from that given by informativeness is that there is no telling which objects will have been noticed fortuitously before the given recognition hypothesis was made. In other words, if the robot has scanned ten primitive objects before it hypothesizes that it is in Place P, it would be horribly wasteful to throw away all the information just collected, and begin some pre-sequenced scan. Hence:

The recognition process must adapt inself to the sequence of events in each particular case, and to the geometry of the informative features of the object being recognized.

This supple adaptiveness may prove to be the hardest aspect of the recognition process for us to come to understand. Certainly the process is not performed optimally in all cases, but just as certainly it is performed at least efficiently in the majority of cases. It is not easy to see exactly how this is accomplished.

This adaptiveness also argues for a representation of the locations of sub-objects in terms of spatial coordinates rather than in terms of sequences of relative movements. This is an unfortunate conclusion, because certainly the most efficient representation of a compound object would be in terms of a definite sequence of actions, such as:

see sub-object A → turn eye 40° → see sub-object B → turn...

However, if this sequential order is often going to be violated, it becomes necessary, although less efficient, to store coordinates for each sub-object, and compute the proper action to get to it at the time that the action is needed. We suspect that in biological recognition systems a combination of both representations is used: a representation in terms of movements is available to give speed, and a coordinate representation is available to give flexibility. Unfortunately, when it comes to modeling a complex process, this redundancy of means only makes it harder for us to comprehend how the process operates.

E. Hypothesis Evocation

In mentioning that it might require the sighting of several sub-objects before a place-recognition hypothesis is evoked, we omitted to ask how this evocation process itself is carried out. What first suggests to the robot that it might be in a given location, so that it can run the recognition procedure on that hypothesis? In some cases, to be sure, a perception is *predicted* on the basis of a formerly-experienced sequence -- e.g., Place B followed Place A before, so look for Place B to follow Place A now. But in many cases (such as being lost, or unexpectedly discovering a new route to a known place) it is necessary to recognize a place that appears by surprise.

It seems that there is little alternative to postulating an "associative net structure" for the process which retrieves place-representations from memory for use as hypotheses. The point is simply that there is no definable retrieval "key"; rather, any sufficiently-characteristic subset of the scene will retrieve the hypothesis of the whole. It should be clear that the greater the number of sub-objects seen, and the greater their informational value, the more certain the correct hypothesis is to be retrieved and tried out first. We are still a very long way from being able to easily make models of processes that behave in this way.

Recognition begins with the evocation of hypotheses, by a process which appears to be a non-sequential, keyless, content-addressing search.

In the above paragraphs we have sketched some of the major factors in describing the representation and recognition of compound objects. Needless to say, we could expand this discussion by a large factor, and still be only at the beginning of understanding how general "pattern recognition" is performed. But still, we feel that we have isolated the essential ingredient of the process, namely a high degree of adaptiveness, supported by a very attentive use of statistical information. If we continue investigating object-recognition from this point of view, we should eventually be able to puzzle out one of the fundamental processes of intelligent behavior.

VI. THE ORGANIZATION OF BEHAVIORAL SYSTEMS

We have devoted a fair amount of time to the consideration of a possible general theory of the organization of behavioral systems. Our immediate motivation for such a study arose out of direct experience in programming the robot simulation. It often happened that we could propose two or more different ways to organize a particular behavioral routine, and we were distressed by the fact that we could find no theoretical basis on which to decide such choices.

We were further motivated by the fact that certain organizational properties recur with remarkable consistency in large behavioral systems (computer programs), even in different systems with very diverse purposes. For example, feedback control of an external condition is found in many programs, as are means for coping with the attendant problems of waiting for the condition to become satisfactory, of interrupting some higher process if the condition goes out of bounds, and so on. Other examples include hierarchical organization of processes, predictive decision-making in the face of uncertainty, and priority scheduling in cases of competition for limited resources.

These considerations led us to undertake the investigation of complex behavioral systems in general. Our grandiose ideal hope was that we might uncover some comprehensive theory, along with an attendant notation, which would allow us to describe and design behavioral systems at will, much as the calculus allows us to describe and design various physical systems. Certainly we did not evolve such a general theory, and indeed we now see some deep reasons why it may not be possible to create one at all. Still, we have learned a great deal about ways of organizing behavioral systems, and about the circumstances under which a given organization is appropriate. The results of our investigation are presented in the following pages.

A. The Relativity of Behavioral Description

Traditional mathematics arose from the attempt to describe the physical world. In the last two centuries, man has learned that this descriptive tool can be turned around and used to *design* new physical systems to suit particular needs. In the case of computer programming languages, the story is the opposite: These languages were created in order to enable the design of computational algorithms, and only later have they been applied to the description of natural systems. This is certainly a case in which the solution (computer languages) supplied the problem (describing behavioral systems). Unfortunately, this order of events has led to the common assumption that computer languages do in fact solve the problem, that is that they are an adequate mathematics for describing behavioral systems. There may be some difficulties with this assumption.

Traditional mathematics will allow an engineer to analyze a design for a new electronic circuit (say), but it usually will not lead him to a *good* design. For this he must rely on experience, inspiration, or at best on a much more complex sort of mathematical computation. Thus, in the traditional case, design is basically an *optimizing* process, while description is not. We are beginning to believe that in the case of behavioral systems, this difference does not exist; rather the problem of describing a behavioral system is also inherently one of finding a *good* description, so that describing behavioral systems is just as much an optimizing process as designing them.

To see how this might be true, consider a desk calculator, of the sort that is 100% mechanical. A set of blueprints for such a machine might tell us all that we could possibly know about its structure; they might even allow us to build a working copy of the device ourselves, so that in some sense they constitute a complete description of the calculator. But certainly the blueprints fail in some fundamental way to tell us *how* the machine works, and what it *does*, for they do not describe any of the *operations* that it performs. Here is the apparent paradox: Even a *complete* description of the structure of a system may fail to be a description of its behavior. But what is there left to describe?

What is left is a set of criteria which exist in the mind of the (human) *recipient* of the description. Suppose we undertake to describe how the calculator divides. To one man we must say: "You punch in A, press the Divide button, punch in B, and then press the Equals button." To another man we must say that it divides by repeated subtraction. To a third man we must give a long rigmarole about which ratchets turn which shafts. In short, there is no answer, no description of "how the machine divides", except in terms of the questions that the description is intended to answer. It is in this sense that a description is not a description unless it is a "good" one.

Now, no doubt this sort of relativity holds trivially for any descriptive system, but the mere existence of traditional mathematics proves the existence of broadly agreed-upon, and therefore implicit, description criteria for physical systems. Apparently such implicit agreement is lacking when it comes to behavioral systems, with the consequence that there is no canonical description of such a system, and hence no simple "mathematics" in the traditional sense of a symbolic system which can be applied descriptively in a straightforward way.

We might remark that the fact that there is no single "correct" or "true" description of the behavior of a complex system does not, of course, mean that there is no true substrate to the behavior. The desk calculator clanks away unconcernedly, leaving *us* to puzzle out behavioral notions such as "the process of division".

We have belabored the point of this section because we feel that it dominates the rest of our discussion. Indeed, if the intuitions expressed here are correct, then it may never be possible to find the sort of calculus of behavioral organization that we set out in search of. Still, we believe that there is much to be learned, even if we cannot formalize our results as fully as we had hoped.

Since our desire for a general calculus appears impossible to fulfill, we have retreated to our secondary motivation, namely the observed commonality of organizational devices in widely differing behavioral computer systems. Concepts like "interrupt", "backtracking", "executive", and so on are known to be important, and they will not disappear on us like the notion of a calculus of behavioral organization. Therefore, as a first step we have set out to examine such concepts piecemeal - that is, without any attempt at synthesis. By concentrating on these concepts, we can gain useful insights into important behavioral mechanisms, and at the same time we can slowly flush out the underlying relationships among various aspects of behavioral organization.

B. Hierarchical Organization of Processes

Any behavior that we observe must unfold linearly with time; why then should we describe or design a behavioral system in terms of a hierarchy of processes? Why do we not represent every system simply as a linear sequence of actions? The reason, evidently, is that we are able to see significant recurring patterns in a linear sequence of events, and we attribute the appearance of similar sub-sequences to the presence of a single "sub-process". That is, we form the concepts of individual sub-processes, such as "squaring a number" or "grasping an object", by *induction over time*, in precisely the same manner that we form object-concepts such as "dog" or "sunset". The nesting of process-concepts gives us the same sort of hierarchy that we have in the case of object-concepts, where "collie" is a sub-concept of "dog", which is a sub-concept of "mammal".

Because of our experience with hierarchically *structured* systems (e.g. computer programs and human management structures), we tend to think of hierarchical *behavioral* organization as being similarly "real", i.e. part of the mechanism that actually generates the behavior. This need not necessarily be the case. For example, we can take any activity, such as "grasping an object", and break it down into further ones, such as "opening the hand", "orienting the hand", "moving the hand to the object", etc.; but this analysis does not mean that grasping *actually* proceeds in phases. The activity *could* be entirely preprogrammed and integrated, or it could be organized in some very different way. (Recall the example of the desk calculator: Its "behavior" is not the same thing as its "mechanism".)

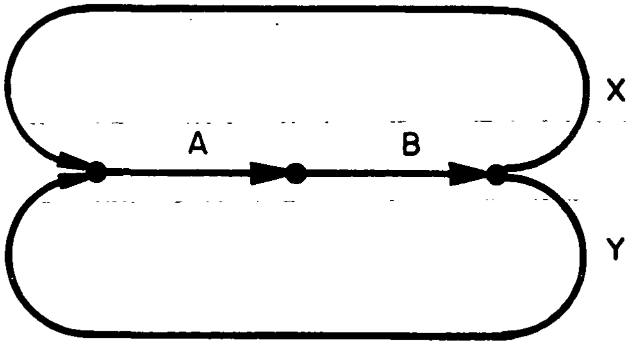
Thus, the act of temporal induction, and hence the description in terms of a hierarchy, come from *us*, and not necessarily from the system that we are describing. This relativity implies that the level of detail and the descriptive particulars in a hierarchical representation depend on the needs of the person performing the induction, and not on absolute properties of the behavior in question. This fact is familiar to any programmer, who must continually decide whether or not a sequence of actions is worth encapsulating as a closed subroutine.

C. Branch Points and Information

One of the major problems in induction is what to do with event sub-sequences which are similar but not identical. An important solution is the use of branch points to allow some elements of a sequence to be collapsed while others remain distinct. For example, suppose that a system has been observed to emit activities A, B, X, and Y, in the following sequence:

... A B X A B Y A B Y A B X A B Y A B X A B X A B X A B Y ...

We might well represent this system by a finite-state device:

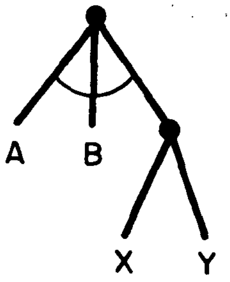


Here, the state after the emission of B constitutes a branch point, where the system "decides" whether to emit an X or a Y. This use of the word "decides" is critically important. It is a prime example of a behavioral imputation that need not correspond to any mechanism actually used by the system that we are describing. In other words, when our inductive analysis leads us to postulate a branch point, we also postulate a decision process.

Furthermore, we inevitably go on to ask *on what basis* the decision was made. We ask what *information* goes into determining the choice at the branch point. For example, our finite-state machine above becomes understandable if we assert that after emitting a B, it reads a symbol off of a tape; if the symbol is 1, it emits an X, if it is 0, the machine emits a Y. Thus, we identify the influence of *information* with (apparent) *choice*. This is, of course, a fundamental intuition of formal information theory; we see here that it is just as fundamental in understanding the organization of behavior.

Sometimes it is the apparent seeking of information that leads us to postulate a branch point, rather than the other way around. For example, when we see a cat carefully scanning a ledge before jumping onto it, we assume that he is deciding precisely how he can execute the jump, if at all.

Although the postulation of branch points does not force a hierarchical organization (as the finite-state representation demonstrates), the two are very importantly related. One simple way of seeing this is to think of a behavioral "parsing tree" such as the following, for the sequence on the previous page:



(The arc indicates an "AND" node; the lower node is an "OR" node.) It is extremely convenient to imagine that there is some entity, some decision process, associated at each branch point, and that this entity "supervises" the activities that are found below it. In the case of an "OR" branching, the supervisor of course makes the decision of which branch should be taken. In the case of an "AND" branching, the supervisor at least decides when one phase should end and the next commence (which is sometimes a non-trivial problem in complex systems like our robot).

We suspect that such postulated decision processes or supervisors are the essence of behavioral representation. Certainly our remaining sections will all revolve about this concept.

D. Spheres of Influence

Once we have postulated a hierarchy of supervisors, it is natural to think of them in terms of the managerial structure of a human organization. While there are a number of inadequacies to this metaphor, it can be quite instructive. We think of a human supervisor as having a certain "sphere of influence." This includes the agents "below" him whose work he controls, and the administrators above him who specify and evaluate his own work. It is important to note that the supervisor's world, that is, his sources of information, are *local*, being restricted to the nearby realms above and below him. Of course, there is no precise definition of "local"; what is important is that some information is harder for the supervisor to come by than other information.

To give an important example, let us consider the case of a man sitting in his living room watching t.v. who suddenly desires a can of beer. At some peripherally conscious level, he realizes that he must get up, go into the kitchen, and open the refrigerator in order to get a can of beer. In order to get up, he calls upon a skilled activity involving placing both feet on the floor, bending forward at the waist, placing his hands on the arms of the chair, etc. In order to place a foot on the floor, perhaps specific neural circuits are used, containing internal feedback loops to ensure smooth control of the muscles. Now, what interests us is that the near-conscious supervisor has not the slightest idea of how the muscles are moved, while the muscular circuits have not the slightest idea of the desirability of beer. (By a valid analogy, a corporation president and a laborer for the corporation have no idea of each other's tasks.) Putting this in terms of information and decisions, we can say that the near-conscious planner is not capable of making any decisions on the basis of signals from individual muscles, and the muscular control circuits are not capable of making any choices based on needs or knowledge involving beer.

Many of the hardest problems in designing the robot control system arise from precisely such disjoint spheres of influence. At one level the robot decides to look at a particular building, but the eye was already being moved in the other direction for a different reason, and besides the building in question is too far behind the robot to be seen any more. Such problems of coordination are basic to any behavioral system which is sufficiently ramified to contain supervisors with non-intersecting spheres of influence. We will return to the matter of coordination after examining one more fundamental notion.

E. Goals

Perhaps the most tenuous concept involved in the description of behavior is that of "goal". Even more than the other notions that we have discussed, the idea of a "goal" is clearly a descriptive artifact. The desk calculator clunks along perfectly well with no goals driving any of its gears or pinions. We have found no single answer to the question of the proper role for the concept of goals, but we are beginning to have some ideas as to where it fits into the scheme of things.

If we consider our hierarchy of supervisors or executives, we realize that the *administrative* tasks performed by these entities (tasks such as keeping track of which subordinates are doing what) are distinct from the overall task of the system. That is, the manager of a steel mill pushes papers, but his ultimate responsibility is to produce steel. We may suggest that the notion of "goal" arises precisely when we have such a separation between an ultimate responsibility and the administrative work required to meet that responsibility. In straightforward behaving systems, where there is no such separation, we do not need to postulate goals. For example, the engine of an automobile drives the wheels, period -- we do not need to say that it has the goal of driving the wheels.

Of course, the designer of the automobile had the goal of making the wheels go around, which is why he supplied the car with an engine. For this reason, it does not sound nonsensical to say that the goal of the engine is to drive the wheels, but in saying so we are merely including the *human* into the system that we are describing. This would be made clearer by a careful linguistic distinction: We should say that the *purpose* of the engine is to drive the wheels; of itself, the engine has no goals.

To take an example at the opposite end of the spectrum, suppose that a man decides to discover a cure for cancer by next February. Here we have the ultimate separation between the end product of a system and the procedure for obtaining it, namely there is no known procedure for obtaining it. In this case, the only useful description of the man's behavior is in terms of a goal.

We see, then, that the notion of goal is a function of the way in which a behavior is described. We should be very careful about postulating goals as a *mechanism* of the behavior itself. This comment applies specifically to the new goal-oriented programming languages, and to some of our own programming on the robot simulation.

It is common to talk of goals in terms of *states*. Even in terms of the cancer example, such a notion seems artificial: the man's goal is to *do something*, namely discover a cure, not to be in the state of having discovered a cure. Also, we may think of organisms whose behavior is commonly described in terms of tropisms: the worm's goal is to move toward water, away from light. Here we may salvage the notion of state by speaking in terms of gradients, but we should be aware that we are embalming time- or space- derivatives in what is supposedly a static description. Thus, it is unduly restrictive to think of goals only in terms of states.

Goals, too, are things that are desirable. What does this mean? Perhaps it means that what a system wants, or what it wants to do, defines its goals? A certain amount of programming experience or philosophical reflection will show that such an analysis is tautological. We must admit that what a system *does* is *identical with* our (often *post hoc*) imputation of goals. However, this identity does not render the concept of "desirability" meaningless. We suspect that this concept can be usefully related to that of *expenditure of resources*. Suppose that on a Sunday a man has to choose between going fishing or mowing the lawn. We observe him to be packing up his fishing gear. We then say that he has selected the *goal* of doing some fishing, this being (therefore) the more desirable alternative. If it had been possible for the man to do both activities at the same time, the description in terms of goals would have been much less useful. Thus, ultimately the notion of goal brings us right back to the notion of branching, of decision.

F. Resource Conflicts

As the foregoing discussion indicated, there is a close connection between decisions and limitations of resources. If a system had unlimited resources of all sorts, it would still have to make decisions involving coordination (see the next two sections), but many of its organizational problems would disappear. This is strikingly clear in the case of our robot simulation, where much of the subtlety arises from the fact that the robot is capable of entertaining many simultaneous hypotheses about the world, but it must check them out serially because of the focal nature of visual attention. (This is not to imply that focal attention is informationally inefficient; on the contrary, it is rich in informational benefits, but these come at a high organizational cost to the system that employs focal mechanisms.)

The resource limitation which is most familiar to computer programmers is that of "processing power", i.e. the enforced serial nature of most of our machines. When a process has "AND-ed" subprocesses, we tend to think of them as sequential steps; when a process has "OR-ed" subprocesses, we worry about the order in which they should be tried until one of them succeeds. It is important to note that these primary concerns of the programmer are in fact artifacts of serial processing in our computers (and perhaps of serial analysis in our conscious thought). In human managerial systems, and in biological nervous systems, there is ample opportunity for simultaneous activity among processes at the same level. In such cases, the notions of "AND-ed" or "OR-ed" subprocesses merge into each other, and we must find new bases for describing the activity of the supervisor. The next two sections will suggest some principles that may be useful.

An important fact about resource conflict is that it may cut across the sphere-of-influence boundaries of individual local supervisors. For example, in our robot simulation, no matter what is the hierarchical relationship of various processes that may wish to move the eye, there is only one eye, and all must compete for it. It follows that the entity which allocates such a resource cannot have *its* sphere of influence confined to any sub-locality; therefore, it must become a *global* decision-maker. This seems to us to be an extraordinarily powerful conclusion. It seems to mean that a system, no matter how homogeneous its elements (e.g. a nervous system), cannot have a homogeneous *behavioral* structure if it contains conflict over resources. There must be some mechanism which allows the attainment and enforcement of a global decision as to the allocation of the resource. We might even suggest that, according to this argument, the appearance of a unitary "mind" is unavoidable (albeit at the level of behavioral description) in any system with a high ratio of potential behaviors to bodily resources.

G. Condition Conflicts

More general than resource conflicts are the problems that arise when two supervisors of independent subprocesses have incompatible requirements as to the state of the world. To air condition your house, the windows must be closed; to ventilate it, they must be open; therefore you cannot do both at once.

In an algorithmically-behaving system, especially a sequential one, the initial design of the system assures in advance that the preconditions for a given subprocess will be met at the time that the process is called for. The more adaptive a system becomes, the more its organization must explicitly cope with the meeting of preconditions before a subprocess can be unleashed. Perhaps the ultimate of such organization is a collection of independent "demons", which are subprocesses that themselves actively "monitor" their preconditions, and autonomously commence their activity as soon as their conditions are met. This "pandemonium" organization is powerful because of its inherent parallelism, but in most cases it must be combined with some sort of executive mechanism which will provide the requisite administrative (global) control. In order to see how such hybrid organizations function, we must gain an understanding of some of the more basic elements of the condition conflict problem.

We often think of "conditions" in terms of predicates which are either true or false. There are a number of reasons why this conception is inadequate. Many conditions (such as spatial position) take on a range of values, which may well be continuous. In many cases it is worthwhile to consider both the value of some measurable quantity (e.g. intensity of a stimulus) and its time-derivative; this complicates the specification of a "condition" involving such a quantity. Often in real systems, the value of a condition can be obtained only to some degree of certainty less than 1.0 ; in such cases there must be a balance between the overhead of ascertaining the condition and the chance of making an erroneous decision. Even worse is the problem of the possible variation in a condition over time. That is, the system cannot afford to monitor all conditions at all times, but conditions may have changed in the interval since they were last observed (with some conditions being more likely to change than others). This latter has come to be known as the "frame problem"; clearly it implies that conditions must be assigned "expected truth values", rather than being represented as predicates which are either true or false.

These kinds of problems are compounded whenever the system takes any overt actions, because then it produces some not-wholly-predictable change in the world. In general, the possibility that any one subprocess will change the preconditions for any other (either favorably or unfavorably) can be computed only in terms of expected probability, since a system has only a partial knowledge of the world, and only limited time to spend predicting the consequences of its actions. Of course, it is precisely this sort of uncertainty which underlies the importance of *sensory feedback*. If you want to know whether or not your elbow is resting in your coffee cup, don't figure it out -- take a look. Or, even better, have "passive" sensors which can *interrupt* an action if it results in the placement of your elbow in the coffee.

The notion of interrupt relates back to the idea of a "demon" silently watching until a certain condition is met, but it further implies the power of one subprocess to halt or at least influence another. Once this vital concept is allowed, our intuitive ability to comprehend the control organization of a complex behavioral system goes from poor to abysmal. This is just the point at which we would like to have a workable mathematical representation, but at the moment we must be content with an informal examination of the concepts that such a mathematics must represent.

H. Temporal Organization

The problems of condition conflict can be looked at from a temporal as well as from a logical point of view. In a sequential system, each subprocess is invoked only when the previous one is complete, at the behest of the administrating superprocess. In a pandemonium system, the temporal interaction is more complex, with the demons "waiting" in some kind of limbo status until they get an opportunity to perform, perhaps interrupting some other demons in the process. In all of this there is still one element lacking: What sets the pace, what determines the global temporal organization of events? This can be made into a fairly deep question.

In many computer programs, the question of pace is totally irrelevant. For example, suppose we are given the mathematical relation $X = (Y + 2*Z)^2$. This relation is inherently atemporal. Now consider a sequential program for computing X in terms of Y and Z:

```
(1)  X ← Z
(2)  X ← 2*X
(3)  X ← Y + X
(4)  X ← X*X
```

It does not matter how fast this program is run. All that matters is that the steps be performed in order; this is what determines the equivalence between the program and the formula.

The situation is of course entirely different for a system that must interact with the real world. If any one subprocess has a temporal extension, then the others must be placed in some temporal relationship to it. We can think of several ways of achieving such temporal coordination, each with its advantages and disadvantages. It is possible to define a global time-scale, "clock time", against which all activities are mapped out. It is possible to specify events in *relative* time; e.g., B happens five seconds after A, but C is temporally independent. It is possible to control a process in terms of the *rate* at which it proceeds. And it is possible to regard time as one of the preconditions to the commencement or branching of a process: e.g. one subprocess could take a certain branch if another subprocess had run for such-and-such a period, or if the clock time were such-and-such. No one of these devices is adequate for all purposes, and certainly all are used in effecting the time-coordination of human affairs.

We feel that time is less understood relative to its importance than any other aspect of behavioral organization. This is especially true in regard to *simultaneous* processes, which are just beginning to receive formal study. For example, the notion of *monitoring*, and of the *supervision* of one process by another are most clearly exemplified when the supervisor and the supervisee are functioning at the same time. Clearly this and similar concepts are crucial to the organization of process control.

I. Executive Bookkeeping

One of the functions of a supervisor is to keep track of what is going on among its subordinate processes. In current programming systems, subprocesses are usually run sequentially, they terminate of their own accord, and their success or failure is evaluated only after they terminate. Even in so straightforward a case, the supervisor may require considerable bookkeeping in order to keep track of what has and what has not been done. The problem grows very complicated if the supervisor is to gain anything from attempts which fail. There is the problem of computing which portion of the acquired hard-knock experience was a function of the particular approach that was tried, and which experience is relevant to any further approach that might be tried. Ultimately, this is a form of the frame problem, solvable only by estimation.

The notions of "success" and "failure" should be treated gingerly, since we would like to distinguish between goals which are explicit to the supervisory process, versus those which are implicit in the organization of the system (e.g., in our little program for computing $(Y + 2*Z)^2$, all goals are implicit, and the supervisor has only to make sure that the steps are executed one after the other, since they automatically "succeed" and "terminate"). Of course, it is even harder to define when a process is "succeeding" or "failing", in terms of a measure of *progress*, yet this must be done in any system where processes cannot be expected to terminate themselves automatically (e.g. the search for an item in a huge memory store).

The notion of "backtracking" in case of a failure is subject to complexities, even in case failure is well-defined, and even disregarding the problem of learning something from the failure.

It presupposes that there is in fact a record somewhere of what was being tried (this is not automatically the case in a pandemonium-like system). Also, the problem of diagnosing where to place responsibility for the failure may be effectively insoluble in cases where the chain of command passes through several disjoint spheres of influence. For example, if our beer-seeking t.v. watcher finds that he cannot move his foot (perhaps it is asleep), the analysis of the situation and corrective action must be made at a very much higher level than that at which the failure actually occurred. Thus, recovering from a failure may be a challenging exercise both in bookkeeping and decision-making finesse.

J. Executive Decision-Making

Given that the supervisor can keep track of what its subordinates are doing, in most systems it must allocate "processing power" or some other resource to them on a merit basis. Presumably the most meritorious course of action is that which will produce the best or most results with the lowest expenditure of resource (including time). Of course, the question is how the supervisor is to know ahead of time, in a non-algorithm-like system, how to estimate the effectiveness and expense of the various alternatives that are presented to it. It is tricky to define how a supervisor can predict or estimate the behavior of a subprocess without of course carrying out the actual execution of the subprocess.

We should also mention that the very generation of alternative subprocesses may be a task that consumes non-negligible resources. For example, if the robot (or an animal) is confronted with a visual scene, it must match that scene with long-term memory in order to draw out hypotheses by which it may recognize parts of the scene. This match-search of memory is a major part of the recognition process, and the system obviously cannot afford to draw all possible hypotheses out of memory before testing any of them. Thus, part of the executive responsibility is to generate new potential subprocesses in a manner which is efficient, as well as efficiently managing the subprocesses which have already been proposed.

This sort of executive decision-making is perhaps the crux of efficient behavior. At the same time, it is relatively simple conceptually (if stated as a choice among alternatives), and relatively well-studied by traditional means (e.g. statistical decision theory). Therefore we will go no further into the mechanisms of decision-making here, since our object is to consider the structure of the behavioral system as a whole. And while it might be relatively simple to enumerate the criteria for any individual decision, it is usually not so simple to specify how such decisions should interact, how supervisors should coordinate and decide priority among themselves, what spheres of influence should be open to each supervisor, and so forth.

K. Deciding the Overall Organization: Statistical Information

The question of overall organization, as we asserted in Section VI.A, is one of *optimizing* with respect to certain goals, whether one is describing a given behavioral system or designing a new one. Thus, in many cases the finding of a *good* formalization is substantially a different problem from the finding of any one that will work at all. The optimization must take into account the system's behavior over a large class of similar inputs; that is, it is essentially an inductive process. For example, suppose that you are introduced to a person, and that he reacts moodily to your attempts to talk about football. From this one event, you have no idea whether the problem is that he hates football, or that he hates introductions, or that he was having a bad day, or that he is generally a surly person. These possibilities can be distinguished only by observing him in a number of similar situations. It is easy to see that the same sort of procedure is necessary for arriving at the proper description of any complex behavioral system.

We would like to emphasize that the information gathered in such experimentation with a behavioral system is *statistical* in nature, and that therefore the selection of an optimum model of a behavioral system is closely connected to the statistics of its responses to typical inputs. This fact has been implicit in everything we have said about alternative organizations of systems. For example, if subprocess B is *always* both desirable and possible after the execution of subprocess A, then the best organization is to make them sequential steps under some larger process. If the applicability of B depends on some particular set of conditions, it might be best to provide a test of those conditions, with the execution of B being dependent on this test. If B is only rarely applicable, or if the circumstances of its applicability are not readily predictable from tests, it might be best to establish B as a "demon" which independently waits and watches for its opportunity to proceed.

Thus, the proper organization of a particular behavior is entirely dependent on the particular statistical peculiarities of the task at hand. This is true of the global organization, and of the details of control throughout the system. Furthermore, there are some problems, such as the handling of the "frame problem" mentioned in Section VI.G., which have solutions *only* in terms of a statistical conformity of the system to its informational environment. Perhaps this ubiquitous influence of the statistical properties of the task is the most important general principle that can be stated about the organization of behavioral systems.