ROBOT CONTROL STRATEGY

Leonard Friedman

TRW Systems Group
One Space Park
Redondo Beach, California

ABSTRACT

A beiiavioral theory of information processing based on ethological observations has been tested experimentally by programming a robot control system. The resultant software has successfully simulated robot operation. Three basic types of software modules are assumed as elements. These accept input stimuli and generate detailed strings of motor response "units" to attain design goals. A complex process of interaction between the modules permits the consideration of many alternative actions while focusing attention on a selected set of input signals. ' The organization is shown to have especially flexible characteristics adapted to robot control and permits a recursive definition of 3ome of the modules. Brief comparisons with other robot systems in being indicate some of the relationships between the different design approaches so far proposed.

INTRODUCTION

The software for integrated robot design may be functionally divided into pattern recognition programs and control systems. Control system programs analyze the environmental state data furnished by the pattern recognizers, initiate decision processes, and issue motor commands intended to arrive at goal states. Many different approaches have characterized the techniques used to arrive at decisions which generate sub-goals and produce detailed motor activity.

The Stanford Research Institute and Stanford University groups have stressed mathematical and engineering schema to solve these problems.[1,2] A team from MIT Instrumentation Labs, have simulated the hypotheses of a neurophysiological theory[3]. A control system model based on instinctive behavior production principles is described here that provides still another viewpoint of the general problem of producing intelligent behavior by machine. The rationale for relating animal behavior to this model is described elsewhere. 4 Ethology (the study of animal behavior) has provided a mass of data on instinctive or preprogrammed animal behavior. It has also introduced some theoretical concepts that indicate there may well be common features in the organization of information processing at the instinctive level through much of the animal kingdom from insects to mammals.5 In any event, the robot control model presented here has processing features which are divorced from the particular robot design inputs (the sensing elements) or the outputs (motor capabilities). This permits a common system of control organization for widely differing sensory capacities and motor repertoire.

After describing the main characteristics of the system, its operation will be compared with some of the other robot systems designs. The general strategy employed for finding a path through state space from initial state to goal will be discussed. For a certain class of problems, this strategy offers advantages.

To test the validity and usefulness of the model, a robot simulation was programmed in JOVIAL on a Philco 2000 computer. As part of the simulation, a display generated an animated cartoon in real time of the robot moving about in its environment, solving the problem situations it encountered. In fact, a movie has been made of typical robot activity. The simulation program is divided into two parts. One part simulates the constantly changing environment; the other part is the robot control program, organized according to the hypothetical decision processes assumed in the behavioral theory4. The architecture of the computer program itself, while faithfully representing the behavior theory, introduces another assumption having nothing to do with behavior; i.e., most of the information processing is carried on in identical modules or, equivalently, is assumed to be recursive. This formalization has implications for further development discussed after the principles of organization are explained.

ELEMENTS OF THE HIERARCHY

The decision and control function is performed by two groups of information processing modules called Releasing Mechanisms (RM's) and Selectors of Releasing Mechanisms (SRM's). The internal

structure of each RM and SRM is nearly identical.* They differ principally in the number and destinations of their interconnections (input/outputs). The behavioral theory assumes a hierarchy of many RM's and SRM's functioning in parallel. Correspondingly the simulation program uses a single subroutine to represent each RM and each SRM, entered recursively for each level of the hierarchy.

As the result of interactions between sensory inputs (stimuli) and these processors, a third group of processor elements is activated. These are called Motor Control Units (MCU's) and are programs controlling the purely motor functions such as turning wheels, steering or moving arms. See Figure 1. Clearly the internal structure of each MCU is dependent on the specific mechanical design of the robot function it must control.

It is the task of the executive control program to activate in proper order, sometimes even simultaneously, the MCU's, thus forming strings of MCU's in order to accomplish desired changes in the environment. The control program has to perform very much like a piano player, with the keys analogous to the MCU's and the melody and chords the desired output. For the robot simulation in this study, the list of these motor control units is naturally a simplified one. The following MCU's have been implemented:

MOVE BY ROTATING DRIVE WHEELS
CHANGE STEERING WHEEL DIRECTION
MOVE ARM
GRASP WITH "HAND-
RELEASE FROM "HAND"

INTERCONNECTIONS AND OPERATION

Each RM possesses a set of sensory inputs which is peculiar to it. These stimuli may be a few simple signals or compound events, a small or a large set. The RM's, subjected to complex and often irrelevant stimuli, must produce goal-oriented responses, the MCU strings. The RM's constitute the nodes of a set of hierarchies or tree structures (Figure 2). In order to produce an output, control must pass to an activated RM. Activation is accomplished by a separate mechanism associated with each level of every hierarchy. This is the SRM net shown in Figure 3. RM's may or may not be excited by input stimuli. If they are, the output is sent to the SRM at that level of the hierarchy. The SRM then picks the maximally excited RM for activation.

RM's are of two kinds, branching and terminating. A branching RM has control of several subordinate RM's, any of which can take control next. A terminating RM either controls an MCU directly or controls the equivalent of a subroutine. These subroutines so far have consisted of complete subassemblies of the three elements with all branches terminating in MCU's. Thus, a given hierarchy always terminates in MCU's.

Subroutines have been introduced because many functions such as "search" patterns and "go to" performance have such widespread use it would clearly be wasteful to duplicate them for different hierarchies. Besides branching and subroutines, looping, the remaining important facility of a program, is easily provided as explained later.

CONTROL PROGRAM OPERATION

Each hierarchy or tree structure of RM's is dedicated to the accomplishment of some specific major goal with the set of hierarchies united at the top into a single overall hierarchy, the instinct drive structure. Biological examples of major goals are food seeking, reproduction, and defense. The implemented robot simulation at present is concerned with only a single major goal, "nest-building."

To make responses corresponding both to the state of the environment and the task, the control program must first select an appropriate hierarchy of RM's. To accomplish this initial selection, the top or zero level SRM network (Figure 3) is organized into a PANDEMONIUM in the following manner[6]. Processed sensory signals from both the internal and external environment are sent to the top level RM's. Different sets of signals are directed to each top level RM. The RM node scans its specific inputs from the environment and outputs a signal to the SRM. An output greater than zero occurs when the pattern of inputs from the environment corresponds to the appropriate situation to which the given RM is supposed to react. That is, we have in effect compared a model situation preprogrammed into the RM and the actual environmental input. There are many RM's competing in parallel for activation by the SRM. The SRM receives the output signals from all the RM'sat the top level, selects the largest signal from this set, and

* An exception is goal selection and subgoal generation in the RM. This may be different for each RM.

activates the RM from which the largest signal came, thus selecting a hierarchy. A concrete example may clarify this process. Suppose the exciting signals for nest building in Figure 4 are "temperature below threshold" or "direct command" from a human controller,and the exciting signal for refueling is "fuel level below a threshold". In this arrangement, we would design our signal to the SRM from the "Refueling" RM to be larger than the maximum signal from "Nest Building" before a dangerously low level of fuel was reached. Then if the robot started with a full gas tank and a command to construct a nest, it would interrupt its structure building to engage in refueling when the analogue output signal to the SRM from the "Refueling" RM exceeded the corresponding signal from the "Nest Building" RM.

The RM's at the top of the hierarchy are never blocked from receiving input signals, but, as Figure 4 shows, each level of RM's below the top is blocked from receiving inputs until its superordinate RM is activated. Once inputs are unblocked, however, the selection of an RM at the new level occurs in a fashion similar to the top-level selection process, thus constituting a multi-layered PANDEMONIUM. As the activation of RM's descends, a level at a time, a new set of RM's is permitted to look for the particular patterns in the environment to which these RM's are ready to respond. If all goes well and the necessary patterns are found in the proper sequence, eventually a terminating RM is reached and one or more Motor Control Units are called which, by their operation, will change the environmental state. When this happens, the whole process of searching through the entire hierarchy from the top level down is repeated, possibly activating a different MCU and simulating parallel operation of the hierarchies.

Provision must be made for unsuccessful performance of an MCU or a failure to find the necessary environmental signals. When this occurs, it is necessary to back up in the hierarchy to attempt a pass through a different part of the tree structure. This backup is accomplished by a superordinate RM, monitoring its subordinate processors for success or failure. If a failure is detected from its subordinates after an RM is activated, the superordinate reduces its output to zero, regardless of its inputs, and a new choice is made at the superordinate level if one is possible. If no choice is available, the processing backs up still another level, clear up to the top if necessary. The top level is arranged so that it always finds a choice. An absence of external signals could constitute a signal to enter a maintenance routine, for example. An example of such a backup may be traced in Figure 4 with the inputs defined in the previous example. Suppose "Nest Building" is commanded and "Seek Nest Site" has for an exciting signal "Has a proper site been found?" If the search strategy of this RM is limited to a single sweep of the area (which is actually the case in our implementation) there might not be any site found in this time that met the stored criteria being sought. Accordingly, a failure would be reported to "Nest Building" that would cause it to reduce the output signal to its SRM to zero. The robot would then try refueling or maintenaxro depending on their relative excitations.

A similar process of backup may take place if two RM's at the same level produce maximal outputs to the SRM of exactly the same magnitude. If the outputs of the two RM's do not conflict, both may be chosen for activation by the SRM. If there is a conflict possible, both may be suppressed and an alternative activity sought in the hierarchies. This phenomenon is actually observed in animals, and the circumstances under which it arises are described in the ethological literature relating to "displacement activity".[5]

GOAL SELECTION MECHANISM OPPJRATIQN

A terminating RM, which may be at any level of the tree, calls a Motor Control Unit or a terminating sub-routine. To do so however, the RM must be able to specify all the parameters needed to serve as a goal for the output procedure. For example, if the subroutine is "GO TO designated object," and three or four objects possessing some of the required attributes are seen by the robot, some criterion must be used by the RM to select just one. The criterion might be "closest" or "leftmost." Whatever the criterion, each RM must be able to examine goal determining inputs (not necessarily the same as the exciting stimuli which caused an output to the SRM), and on the basis of these inputs, it must select the goal parameters needed to operate its subordinate data processors. This mechanism, internal to an RM, is called the Goal Selection Mechanism (GSM). The process of generating subgoals has not been standardized. Each RM may have a different routine in its GSM.

What is meant here may be illustrated by some of the goal selection processes in the GO TO subroutine. The primary goal in the GO TO subroutine, once a prime target has been selected, is a simple measure. The goal is attained if distance to target is less than a stored quantity (determined by the robots arm length).

If there are obstacles between the prime target and the robot, subgoals are generated using some fairly complex algorithms. These subgoals are directional vectors that are determined by calculating the smallest angular deviation permitting the robot to pass around or through the set of obstacles it is facing. The deviation is calculated from the prime target direction at the robots position and the algorithm selected depends on whether a "Single Obstacle" RM, "Two Obstacle" RM or "More Than Two Obstacle" RM is excited. On the other hand, the goal selection previously cited, involving choice of a prime target from a group of three or four suitable objects, can be implemented using the PANDEMONIUM apparatus with "closest" exciting the largest signal.

If the RM has been tentatively selected for activation and cannot find a suitable goal, it reports a failure and the backup process is initiated. In addition to specifying goal object, the GSM specifies a desired goal state. The GSM monitors the environment for a defined goal state of success or failure and relays the appropriate signal to the SRM when such a state is attained.

INTERNAL STRUCTURE OF THE RM AND SRM

Figure 5 is a block diagram that represents in simplified fashion the flow of information through an RM to an SRM and back. Initially, inputs are summed. These inputs originate from any place external to the given RM and are quantities with a positive or negative numerical magnitude. The inputs represent highly processed patterns; i.e., the answers to a test such as, "Has a nest site been found?" The negative magnitudes tend to inhibit an RM output. After the inputs are summed, a threshold quantity is subtracted and the resultant, representing degree of correlation with the expected situation for that RM, is transmitted to the SRM. The Maximum Select procedure of the SRM finds the RM that has sent the largest positive signal. If this is a new choice, the SRM signals to the selected RM to try to find goal parameters. If the RM can find goal parameters, it is activated and the processing descends another level in the hierarchy. If all the parameters can not be specified, the output of the tentatively selected RM is reset to zero and the SRM repeats its processing. If there is no output greater than zero from any other RM at that level, a failure signal is passed to the SRM at the next level above in the hierarchy and the SRM at this higher level sets to zero the output of the RM at its level and in control of the lower level; the SRM then reevaluates. A complete reevaluation takes place at the completion of every MCU output and can also be made under priority interrupts. This is a limited simulation of full parallel operation at the top level of the set of hierarchies.

Tracing through the information flow of a "Refuel" RM may serve to make Figure 5 clearer. Figure 6b illustrates such an RM with "Difference between Full and Present Gas Level", the exciting stimulu and "Gas Station Seen" as an input to th Goal Selection Mechanism. With such an arrangement, if the exciting stimulus exceeds some pre-set threshold, the "Refueling" RM begins to compete with "Nest Building" and "Maintenance". If this should be the largest signal, the R will be tentatively selected. However, if no gas station were in view, its output would be reset to zero and another RM selected resulting in a different activity despite strong excitation.

PROGRAMMING BY HIERARCHY ASSEMBLY

Using this system to produce a particula pattern of behavior requires the program designer to characterize expected states of the environment between initial and goal states by patterns or cues external to the given RM and discriminable by the robot. Such cues would include successful completion of a previous robot act, permitting chaining of motor activity. It is not enough merely to find any set of cues. For successful operation, ther should be included environmental signals that are highly improbable in any contex except the desired activity.

Animals that rely heavily on instinctive behavior have developed just such distinctive cues in social interaction. For example, male stickleback fishes, normally white-bellied for camouflage, show a bright red belly during the matin season. Other males and females react t this color with complex behavior even when it appears on red-bellied dummies otherwise quite unlike a fish in appearance!

A set of intermediate goal states which are highly likely to appear must be identified by the system designer. In addition, he must decide on the proper operations upon the environment to be effected by sequences of activated MCU's to pass from one intermediate goal to tlie next. A hierarchy of RM's and SRM's with the appropriate cues and calls to a succession of MCU's can then be most con veniently specified in a compact parameter notation. This formalized notatio is supplied to an instinct "operating system" which assembles each set of elements as the environment and hierarchy

selects them. Thus, the designer can specify an arbitrary hierarchy of stimulus-response elements to solve any desired problem within the constraints of the robot design capabilities for sensing and acting and he can do this with a minimum of instructions. The operating system or executive will call the activated sets according to these instructions and automatically provide the parallel multi-choice PANDEMONIUM hierarchies without further coding by the designer.

An example of the kind of system design possible may be shown by two different arrangements of the "Refuel" function as shown in Figures 6a and 6b. The arrangement of Figure 6a means that only a single top level threshold determines whether the robot will engage in refueling activities, including searching for a gas station. The arrangement of Figure 6b permits two thresholds. At the higher level of fuel state the robot will refuel only if a gas station happens to be seen while it might be doing sometiling else. If this should not happen and fuel level continues to drop, the "Seek Gas Station" RM will then preempt other activity.

Besides branching and subroutines, the other programming element, looping, is readily accomplished by an appropriate hierarchical arrangement of RM's. A superordinate RM can specify a goal state not necessarily immediately attainable by the action of its subordinate RM's and MCU's. Nevertheless, a chain of events can be specified at the subordinate level designed to accomplish part of the task at each pass and such that it is a circular chain; i.e., actions at the end of the chain return the discriminable environment to the same state it was in at the beginning of the pass. For example, if the task is to pile objects at a particular place, the superordinate exciting condition may be "less than 6 objects at (x,y)". This would cause subordinate RM's Lo activate first a seeking behavior pattern in the environment and then fetching operations to (x,y) for a first found object. Once this object was placed, the RM for initiating search would be excited again, as "less than 6 objects at (x,y)" is still true. Action would continue till the goal state of six objects placed was reached. Note that action is automatically discontinued once the robot has sensed that the goal state is attained even if the robot did not place the objects itself.

A LANGUAGE FOR HIERARCHY ASSEMBLY

The present status of the system is such that the program designer must insert RM designations in a table provided, specify which RM's are subordinate to which by other tabular values, etc. The whole process is now formal enough to permit the construction of a special hierarchy assembly language with commands expressing concepts such as:

ADD INPUT STIMULUS xx TO RM yy.
RM £ REPORTS TO RM a7
MCU bb REPORTS TO RM c.
RM dd GOAL PARAMETERS GO TO SUBROUTINE RM ee, etc.

Such a specialized language is of course possible in any computer application whenever the use justifies the effort of writing a translator, and the formal syntax can be specified. Just such a use is foreseen in adding learning capabilities to the present system.

AN APPLICATION TO LEARNING

Various forms of learning have been proposed to make programs behave intelligently. Samuel's method7 of changing weights has been the most successful to date, but it suffers from one defect common to all the really powerful artificial intelligence programs so far implemented. Namely, the features or cues from the environment to which adjusted weights are applied must be recognized and supplied by the human program designer.

Suppose a learning program devoted, among other things, to determining previously unknown cues from the environment can be designed to operate in conjunction with a substrate of "instinctive" starting capabilities. In other words, the robot has enough built in capability to give it a chance to learn. Once the learning mechanism had determined a new cue, it would have the task of modifying the instinct structure to incorporate this new input signal. If the learning mechanism had a hierarchy assembly language available and a translator in operation for executing a chosen assembly command, such facilities would permit efficient handling of this part of the total learning task.

There remains the difficult job of selecting the right command. It should not be necessary to resort to random mechanisms such as Friedberg used to do this.[8] The context in which a new cue is isolated will furnish heuristic pointers identifying with some likelihood the RM to which it might be associated. For example, let us assume that a constant association in time takes place between a previously unknown cue and one that is known because it is already used to excite an RM. Such an association makes that RM a likely candidate for adding on

the newly discovered cue. In short, constraints are sought in syntactic relationships between the unknown and various elements of the existing hierarchy as determined by interactions with the environment. Only selection of such commands will be attempted as can be reasonably constrained to a limited set of possibilities.

PRESENT STATUS

At present, minimal use is made by the instinct system of updated internal memory. On the other hand the state of the external environment itself is used as a kind of memory of what the next action should be. This feature is reflected in instinctive animal behavior and was long ago noted by the ethologists9. It obviously results in a great economy of memory.

Because no planning programs have been implemented as yet, the strategies available to the present system are sequential; i.e., the robot tackles the environmental state in its immediate neighborhood, acts on it, considers the resultant state, etc. There is almost no look-ahead. On the other hand, the system exhibits considerable flexibility and adaptability considering the rigid strategies it must now employ. For example, the robot film shows a search-fetch sequence in which ADROIT* sets out to pick up a nest-building object. When a similar object is placed directly in front of it, ADROIT changes goals, stops moving towards the previous target object and immediately picks up the newly presented one. This happens because all goal objects in the field of view are being reviewed for optimality and the closest becomes the prime target. Similarly any goal states specified in the RM hierarchy and in a "state-of-being" before motor action is initiated are detected and motor action to get to those states is suppressed accordingly. This results in "behavior" most unlike that of an ordinary machine, and of the kind normally associated with animals.

This suppression of motor action usually taken by the program is an example of "unplanned" branching which is typical of ADROIT's functioning. We may regard the subordinate RM's in the tree structure and the MCU sequences they produce as planned branches or jumps from a superordinate RM. Any time the program departs from a normal string of MCU's and instead produces an MCU string associated with a higher or lower level of the given hierarchy or a completely different hierarchy, this may be denoted as "unplanned".

* Acronym for the robot

It is characteristic of the program in operation that its action is full of such jumps. It has been our experience that once the program is debugged, these jumps appear appropriately natural and "life-like".

ASSUMPTIONS ABOUT PATTERN RECOGNITION

The most ambitious of the robot construction programs to date is the SRI project. The project has focused on actual hardware rather than simulation and has already achieved notable progress. It has developed some elaborate pattern recognition programs, particularly for the visual sense. Table 1 shows a comparison between the sensory and motor systems developed by SRI and the assumptions about pattern recognition capabilities of the simulated ADROIT systems.

The McCarthy project at Stanford University and the Minsky - Papert arm and eye project at MIT have made considerable progress in visual pattern recognition problems, as well as arm coordination sensing and motor adjustment. A greater pattern recognition capability than assumed for ADROIT's arm has been implemented by these teams.

CONVERGENCE OF TECHNIQUES

The SRI project has already made use of powerful mapping and planning techniques with great success. Nevertheless, SRI has introduced programs using unplanned sequential action techniques similar to ADROIT[1]s for the case of largely unknown environments, in order to avoid excessive computer processing associated with planning. On the other hand, the total lack of planning even when the information is available is clearly a defect of ADROIT*s. Planning will be introduced into the ADROIT system at the earliest opportunity.

The MIT Instrumentation Laboratory project which is designing a robot for unmanned exploration of Mars, has .simulated a neurophysiological theory of operation ol: the reticular formation for its decision making system. This model appears to serve the same function as choosing a major goal hierarchy in the ADROIT behavioral model.

The reticular formation simulation arrives at its choice by statistical rather than deterministic means. Although exploring in great detail how such a choice of mode is made, the published descriptions of the simulation do not discuss methods to arrive at appropriate motor action.[3] Nothing is known to us about how the project plans to achieve detailed and coordinated motor behavior.

LANGUAGE LEVELS

The ADROIT program uses a kind of operating system executive to provide the pointers for the fully recursive capability not available in JOVIAL and to assemble the hierarchy stipulated by the program designer. This executive is itself written in JOVIAL, as are all the other non-display routines, thus avoiding language incompatibility problems faced by the SRI System.

This is a somewhat misleading comparison, however, because the motor control unit routines in the simulation are merely simple algorithms determining updated position of the affected part in obeying a commanded movement, whereas in a hardware system, the rapid action required demands assembly language as implemented in the SRI system. Furthermore, nothing in the ADROIT System corresponds to the SRI plan of an english language communication system between man and robot, with subsequent robot processing by first-order predicate calculus. This has been programmed by the SRI team in LISP 1.5. It is premature as yet to make a decision about the basic language to be used for ADROIT's learning mechanisms.

SUMMARY

The effort to simulate a theory to account for the production of instinctive animal behavior has led to the development of information processing modules useful in robot control design. To create a working system, the modules are assembled into larger structures possessing the properties of any program; viz; brandling, subroutines, and looping. In this regard, the technique is similar to the biological processes of evolution whereby various standard modules, the specialized cells, are assembled into organs. In our analogy, the genetic and developmental processes leading to organ assembly take the place of the program designer and operating system executive leading to hierarchy assembly.

There are certain advantages to this form of organization. Many of the desired structural features of the program can be written in a compact notation stipulating only a few parameters such as module interconnections. An almost uniform formal structure has thus been established for accomplishing any robot control task amenable to programming. This particular structure has automatically built into it certain interactions not standard to programming. Among these are a correlating process for establishing the most appropriate branching action, given the present state of the environment. The branching action automatically works in backing up in the hierarchy to a higher level of decision when encountering failures, as well as moving down to motor action with success. There is an effective "focus of attention" action so that frequently irrelevant inputs will be ignored. At the same time, all the stimuli known to be relevant to a group of the tasks-at-hand are actively sought and processed. Goal states that normally require motor action to attain will be detected if they exist beforehand so that unnecessary motion is suppressed and correct subsequent action automatically is chosen.

A learning system at a higher level is planned, to operate in conjunction with the assembly of hierarchies constituting the actual robot control. Not the least of the advantages of a uniform formal representation of this control system is that it permits the introduction of a hierarchy assembly language and translator. The learning system may then use such a language to add to and improve on the work of the program designer in adapting the control system to particular environments.

REFERENCES

1. R. Raphael, "Programming a Robot", Proc. I.F.I.P. Congress, 1968.

2. K. K. Pringle, W. M. Wichmann, and J. A. Singer, "Computer Control of A Mechanical Arm Through Visual Input", Proc. I.F.I.P. Congress, 1968.

3. W. L. Kilmer, W. S. McCulloch, and J. S. Blum, "Some Mechanisms for A Theory of the Reticular Formation", AF-AFOSR-1023-67, Final Scientific Report. Principal Investigator: Dr. William L. Kilmer, Division of Engineering Research, Michigan State University, East Lansing, Michigan.

4. Friedman, L., "Instinctive Behavior and Its Computer Synthesis", Behavioral Science, March 1967.

5. Tinbergcn, N., The Study of Instinct, Oxford, Clarendon Press, 1951.

6. Sclfridgo, 0. G., "Pandemonium: A Paradigm for Learning" in Mechanization of Thought Processes, NPL Symposium No. 10, London: IKISO, 1959, pp. 511-526.

7. Samuel, A. L., "Some Studios in Machine Learning Using the Game of Checkers; II Recent Progress" IBM Journal of R. and D., 11, p. 601-617, 1968.

8. Friedberg, R. M., "A Learning Machine, Part I, IBM Journal of R and D, Jan. 1958, 2, pp. 2-13.

Friedberg, R. M., Dunham, B., and North, J. A., "A Learning Machine Part II", IBM Journal of R and D, June 1959, 3, pp. 282-287.

9. Thorpe, W. II., Learning and Instinct in Animals, Cambridge: Harvard University Press, 1956.

TABLE 1

	SRI rOBOT SYSTEM	ADROIT
Sensory Abilities	Develops outlines of object edges	Edge sensing assumed
	Range measured by optical ranqe finder	Range sensing assumed
	Tactile sense developed	Tactile sense assumed
	No Arm.	Distance from arm tip to target measured. Shoulder and elbow angle sensing assumed
Motor Abilities	Drive wheels	Drive wheels
	Steering wheels	Steering wheels
	No Arm.	Arm pickup ability

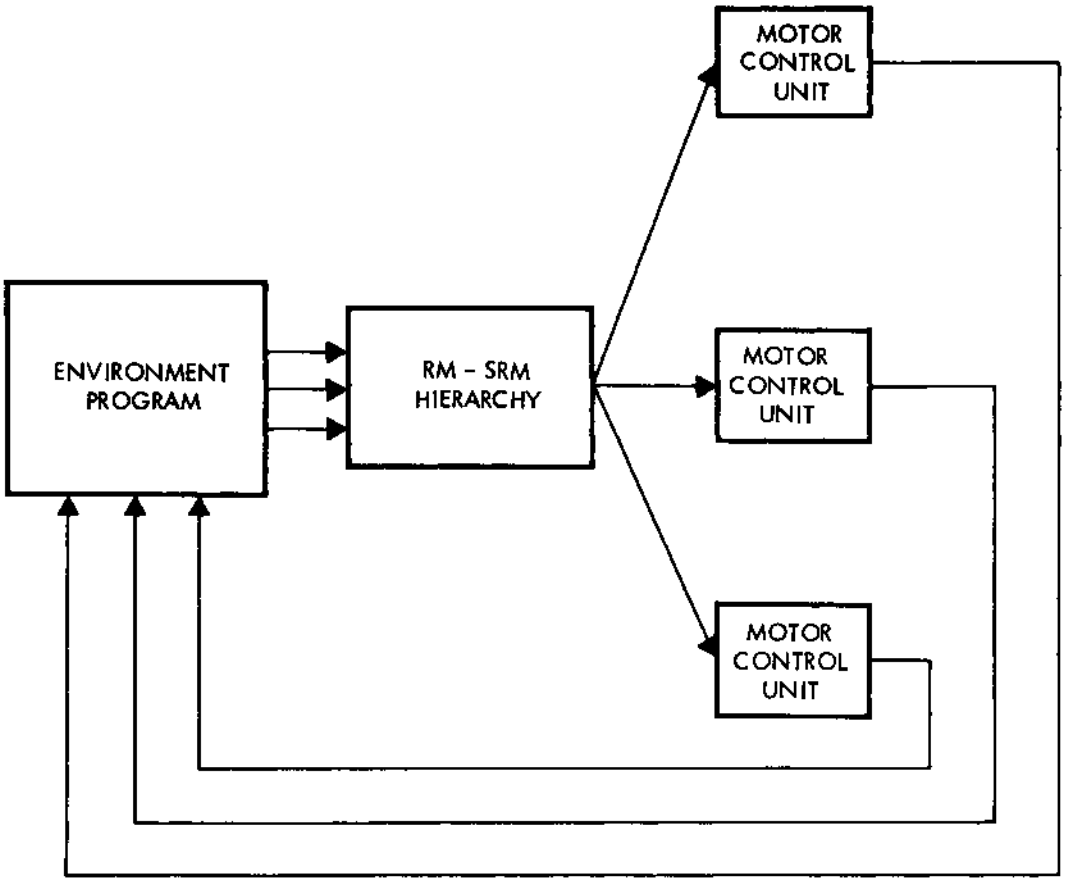

FIGURE 1. FLOW OF INFORMATION IN THE SIMULATION

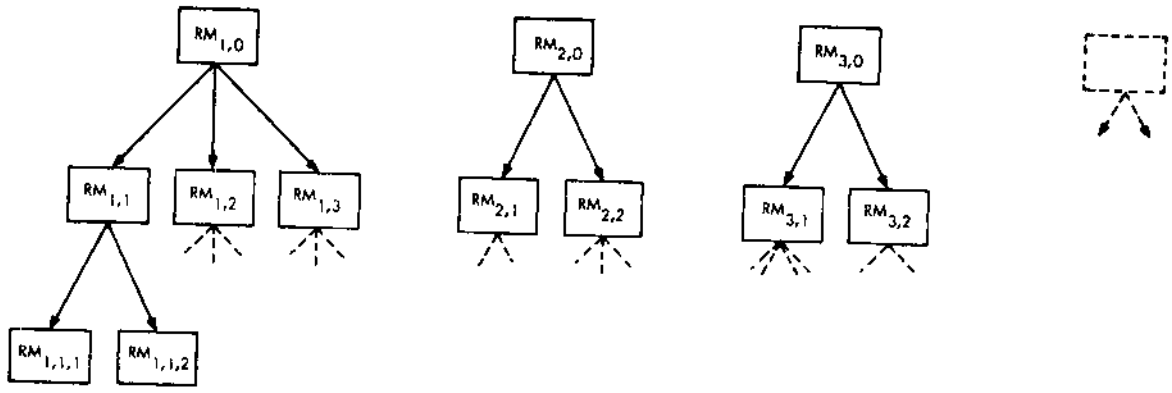

FIGURE 2. SET OF RM HIERARCHIES

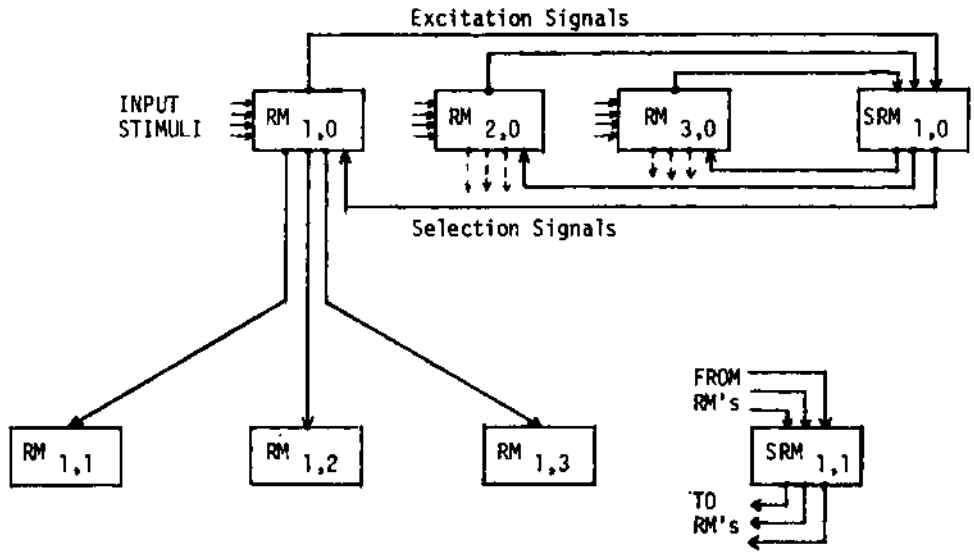

FIGURE 3. THE SRM NETWORKS

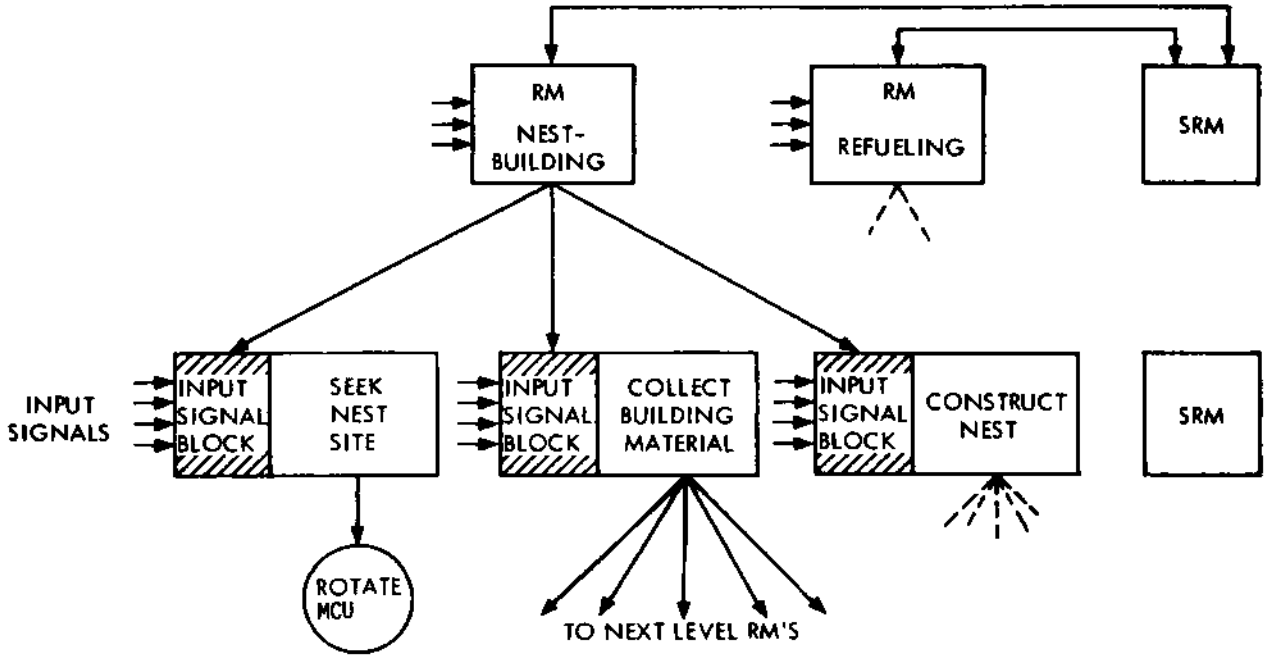

FIGURE 4. HIERARCHY FOR NEST-BUILDING

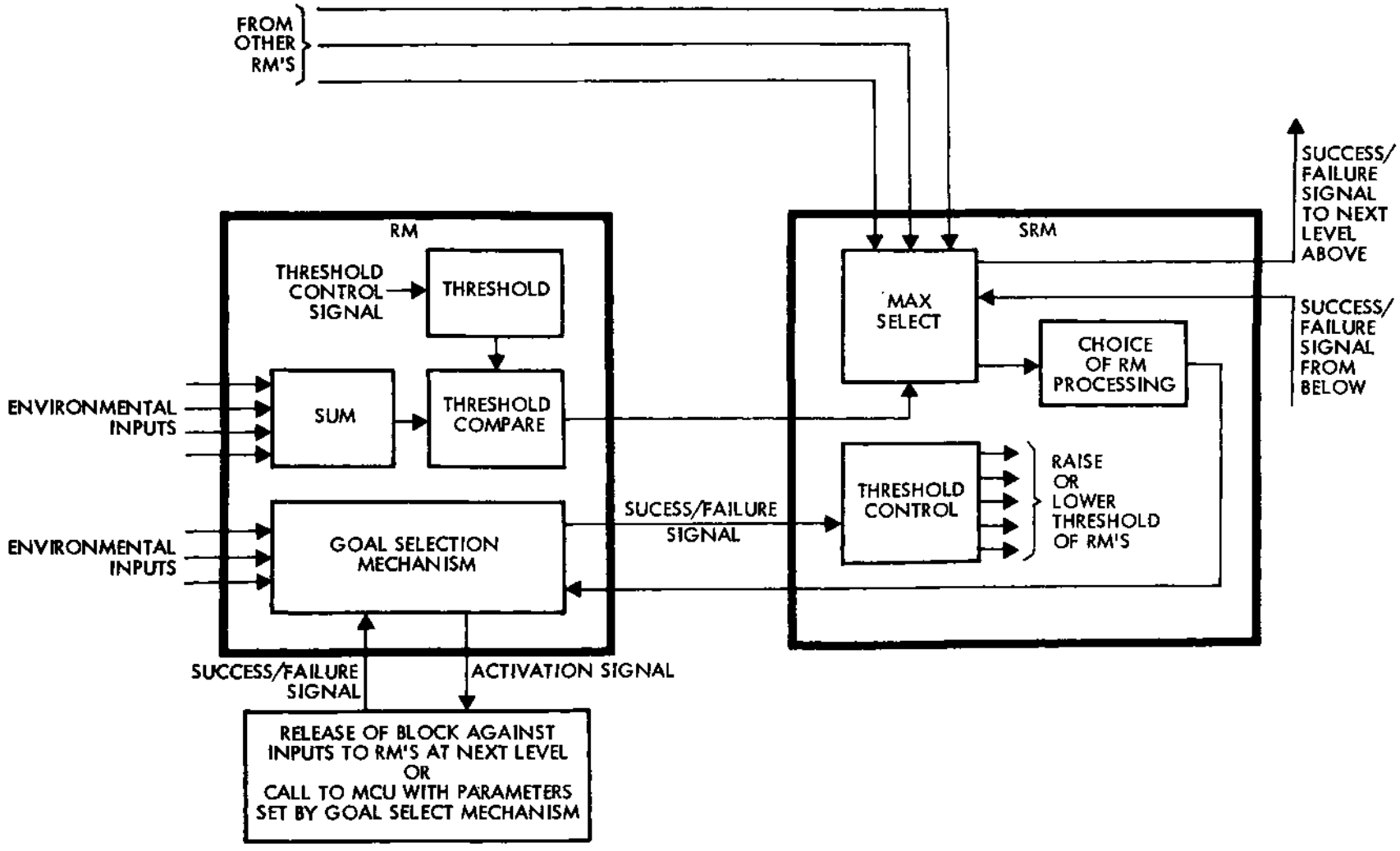

FIGURE 5. INTERNAL STRUCTURE OF THE RM AND SRM

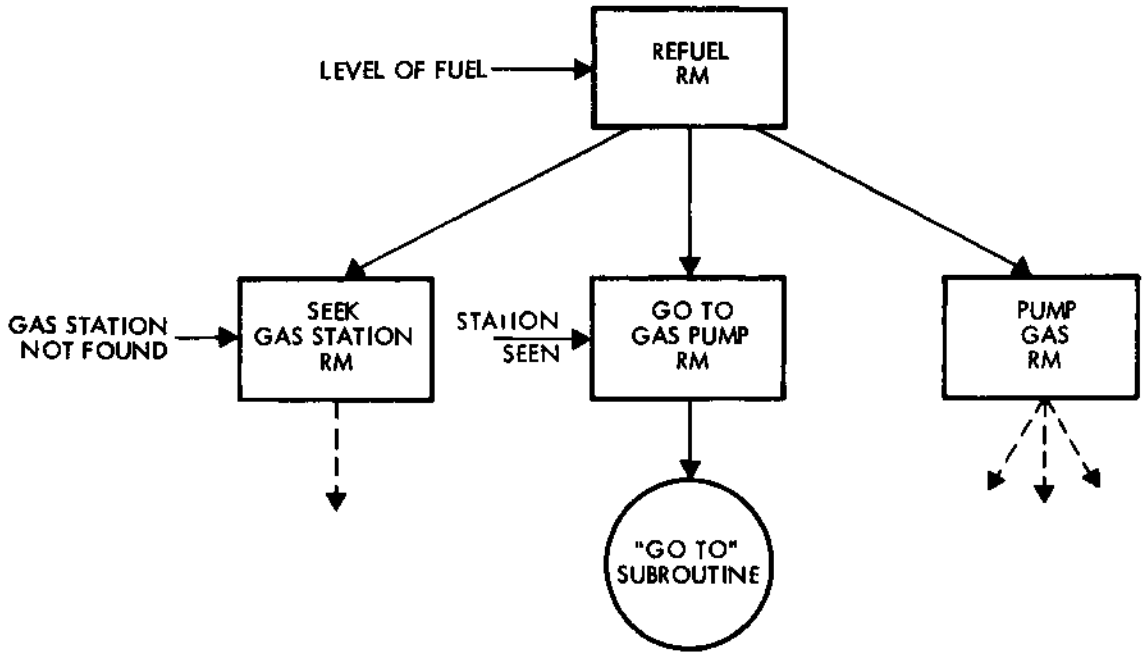

FIGURE 6A. SINGLE THRESHOLD REFUEL ASSEMBLY

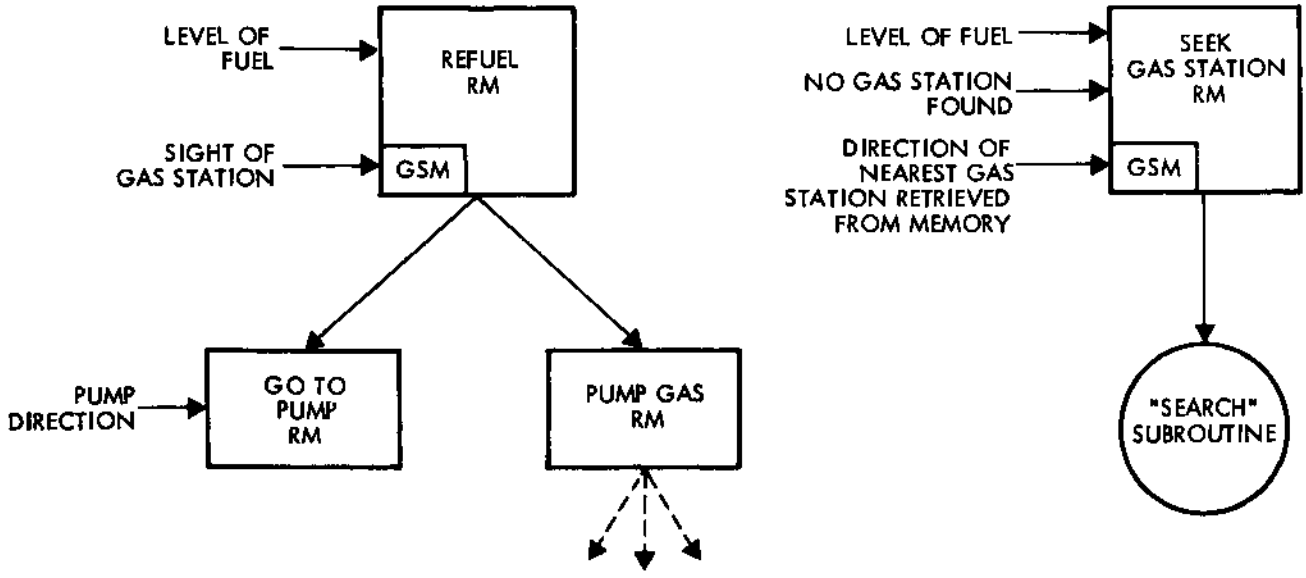

FIGURE 6B. DOUBLE THRESHOLD REFUEL ASSEMBLY