ROBOT PIANNING, EXECUTION, AND MONITORING IN AN UNCERTAIN ENVIRONMENT

John H. Munson
Stanford Research Institute
Menlo Park, California, U.S.A.

ABSTRACT

An intelligent robot, operating in an external environment that cannot be fully modeled in the robot's software, must be able to monitor the success of its execution of a previously generated plan. This paper outlines a unifled iormalism for describing and relating the various functions oi a robot operating in such an environment. After exploring the distinetion between the external world and the robot's internal model of it, and the distinction between actions that interact with the world and the robot's descriptions of those actions, we formalize the concepts of a plan and of its execution. Current developments at Stanford Research Institute, and the benchmark idea oi an ultimate rational robot, are both analyzed in this framework.

INTRODUCTION AND MOTIVATION

We can describe robotry as that subfield of Artificial Intelligence (AI) in which the intelligent system in the computer deals directly with a real, external environment. As a part of AI, robotry potentlally partakes of all the problem areas oi AI: We want to develop robots capable oi problem solving, pattern rocognition, 1anguage comprehension, and so on. However, interaction with an external environment that cannot be fully modeled in the computer emphasizes a new set of problems 1argely unique to robotry. These problems center on the robot's execution, in an uncertain environment, of previously generaled pians.

In any AI problem tormulation there is an iniormation structure within the computer that constitutes a model of the problem domain. Given the present state oi the art, the models tend to be reasonably simple. Puzzles and board games have been very popular problem domains for AI because the domains can be fully represented by relatively simple and unambiguous models, freeing the experimenter to concentrate on the problem-solving issues Other domains, which reflect real-world problems, are typically abstracted and limited to simple models that serve as vehicles for problem-solving studies. An excellent example ot this approach is the monkey-and-bananas problem.(1,2) The states of the model are iew, and the actions of the operators that can affect the model are considered to be unequivocal.

In such an approach, the test for successful operation of the problem-solving system is inherently based on the model itself. If the system finds what it reports to be a solution to a problem (and if the system is logically sound), the experimenter is satistled.

In robotry, however, a different situation obtains. The system must interact wlth the real environment, or world, which is represented internally by the robot's model. In general' for a number of reasons, this representation will be neither comprehensive nor exact:

(1) Real-valued quantities cannot bo measured, nor represented, with infmite precision;

(2) Many physical objects and situations do not admit of complete description (for example, a human, or a complex piece of equipment);

(3) Sensory or perceptual activities, used to update the model in accordance with the world, are subject to accuracy limitations and also to gross errors,

(4) Effector activlties that affect the world are subject to inaccuracies (e.g,, distance moved) and also to gross failures,

(5) The state-of-the-art may not permit a model large enough or sophisticated enough to represent fully the pertinent aspects of the world, even ignoring the other difficulties listed.*

Ry assumption, the model constitutes the sum total of knowledge about its environment on the basis of which the robot must make its plans. The acid test of the plans occurs, however, when they are executed by the physical robot acting in the world. This distinetion between the internal and external environments introduces new issues of <u>execution</u> and <u>monitoring</u> that are characteristic of robotry.

This paper deals, then, with the beginnings of a theory relating robot planning, execution, and monitoring in an uncertain environment. We are concerned with formalizing the robot's uncertainty about world-states and the consequences of its actions, and its ability to deal with a planning tree whose branches have measures of probability or uncertainty associated with them.

It is too early to say what directions the teasible implementations o1 such theory will take, since the terntory is a part of AI that is largely unexplored. Because of the almost universal occurrence of uncertainty in realistic environments, research in robotry Is likely to be led ultimately to new iormulations incorporating the ideas of

We recognize that this fifth point could apply, and the considerations of this paper be applicable, to cases in which a computer deals with an external environment" that is not physical—for example, another computer program or human belief structure. For simplicity, however, and in keeping with the initial motivation of this work at SRI, we shall continue to refer to the domain of our inquiry as robotry.

probability theory, Tuzzy sets,(3,4) or modal logic,(5) in contrast to the two-valued deductive logic characterist ic of current work in problem solving and theorem proving.

ROBOT WORD STATES AND MODEL STATES

As a general framework, we adopt the iamiliar terminology of state spaces and transitions induced t herein by various operators . We define W, the robot's world space, as the collection of possible states of the environment of the robot, W - $(w_1\}$ • At a given point in time, the world is in some state w_1. We associate w_i with the experimenter's (presumably omniscient) view of the robot and its surroundings, so that w_1 includes all there is to

know " about the environment. Clearly, given the present st atus of our capabilities in informat ion representation, we cannot reduce W to an explicit formulation .

We define M, the robot's model space, as the set $\{m_i\}$ of possl bio states of a dist inguished data structure in the robot's computer, i.e., the model. By assumption, the model comprises all of the robot's information about the current status of itself and its surroundings. At a given point in time, the model is in some statr. m_1 .*

In keeping with the reasons listed in the introduction, there is no simple relationship between the elements of W and the elements oi M. In particular, there is no unique functional mapping in either direct ion. Present-day models are necessarily very simple and crude relative to the worlds they represent, so that a given stale of the model will represent many states of the world. Conversely, when the world is in a given state, the model may be in any state. Intuitively, we feel that some state or states of the model are correct descriptions of a given world state, whereas others are incorrect. Formally, one can postulate a modeling relation RWM, which maps world states into the model states that correctly (or best) represent them.

In general, the modeling relation R cannot bo a function in the mathematical sense, uniquely defined at every point in W, If, for example, the world consists of a single doorway, there will be not only states of W that clearly map into mopen and inclosed, but also marginal states for which the correct state of M is ambiguous. Furthermore, the marginal region is context-sens!tive:

*The states of the robot's model, denoted by m_i herein, are the same as the states denoted by S, S_i, etc., in the paper describing the STRPS planner. (6) Also, our model is called the world model therein, and there are other minor differences in notation.

Although the experimenter's "omniscient view" may include knowledge of the robot's model and its program, these are not contained in w_i ; W_i includes only the external, or physical robot and surround ings .

it diters depending whether we are modeling the ability of the doorway to let the robot pass, to shut out a draft of air, etc.

One approach to this difficulty is to detine R as a partial function, defined only whore the mapping is unambi guous, but this prohibi ts the modeling of marginal states . A better alternati ve is to include both mappings w_m m_{open} and w_m— nicinsed , where w_m is a marginal state. One might further try to refine the mapping by attaching probability or conlidence assignments to both branches of the mapping, but it is questionable whether the idea of probability captures the desired spirit in this situation. Perhaps a more appealing approach is the introduction of fuzzy sets and iuzzv functions developed by Zadeh,(3) Chang,(4) and others, in which various mathematical concepts (e.g., set membership) are broadened to include nonbinary alternati ves. It is beyond the scope ot this paper to explore this issue further. We merely point out that this is an unsolved problem that arises at every turn in the modeling of real environments.

ACTIONS AND OPERATORS

Included in the robot system is a set Q - $\{Q_j\}$ of actions, through which the robot 1ntoract s with i ts world, causlng changes in i ts environment and/or gathering perceptual informat ion therefrom. We may think of each action as being embodied in an actlon routlne in the robot's software, which can bo invoked as desired by the robot's overall executive routine. From our viewpoint (that of the omniscient experimenter) , the ant i cipated outcome of the application ol Q_j when the world is in some state w_x and the model is in some state m_1 is a change to new states In both the world and the model, thus, the action may be described by a functional relation mapping the world-and-model Cart es i an product space into i tself .

$$Q_1 : W X M \quad W X M$$

We distinguish sharply between the robot's actions and its operators, 0 - [0] . Whereas an action Q is a routine the robot can execute in order to ihteract with the world, the corresponding operator 0 is the description of the expected results of that action that is available within the robot system. We might call the operator the robot's model ol the action .

Because the robot's software (except for the action routines) can only deal with the model, and not the world, an operator can only be a relation among states of the model,

$$O_J : M \rightarrow M \quad .$$

We can think of each operator as being embodied in an operator description, a routine (or data for driving an. interpretive routine) that yields the desired functional transformation when applied to the model. We shall use the terms operator' and operator description interchangeably, and the symbol 0. to refer to both.

Actions and their Possible Outcomes

In developing and describing actions and operators for a robot, we tend to think of them according to their desired outcome: roll ahead *x* feet, go to the next room, plan a route, and so on. We must pay heed, however, to the fact that various outcomes are possible and that the experimenter's estimates of the possible outcomes of actions and their likelihoods differ from the estimates made by the robot (i.e., contained in the operators).

An omniscient experimenter might report the behavior of a robot like the SRI robot, executing the action roll ahead x feet, as follows: Usually, the full roll is completed. Experience shows that, for a given x, the actual distance rolled is described by a Gaussian distribution with mean 0.98 x and standard deviation 0.04 *x*. The robot's position coordinates in the model will be incremented by x cos θ and x sin θ where 8 is the current angular position of the robot in the model (not in the world). If, however, there is an obstacle in the robot's path, it will bump that obstacle, stop, update the model with the new robot position and also with the entry of a new object, and terminate the action. On the other hand, we have programmed the robot to check the model before moving, and if it finds a modeled obstacle in the path (whether there in reality or not) it will terminate the action without moving and will report the cause of its failure.

Several points may be made about this description. Although it describes one of the robot's primitive actions, it is already somewhat complex. Even so, it is far from being comprehensive: The experimenter has neglected to describe additional "failure modes of the action that may occur in reality, such as slippage of the robot's wheels.

Even among the modes he has described, the experimenter cannot predict the exact outcome of a motion, and he has quantified his degree of ignorance among the infinity of possible outcomes with a probabilistic relation Q_j containing a (Gaussian) probabillty density function. In other cases, a "fuzzy" or modal form of Q_j might be deemed to best express the human's manner of estimation.

In practice, one models the outcome of a completed robot motion with a single outcome specification. One would like to dispense with the tedious mechanics of error analysis, but it is a fact that such motions lead to cumulative error that must be dealt with ultimately. This seems to be a basic problem of robotry in a physical environment. The natural solution is to use perceptual feedback on some sort of periodic basis, perhaps governed by accumulated anticipated error, to correct the errors in the robot's dead reckoning.

In the illustrative action description given above, we observe that the final states of the world and the model depend in a significant way on the initial states of both. That is, factoring Q_j into its components, we have

$$Q_j(w_1, m_1) = (w_f, m_f) = (Q_{j,w}(w_i, m_i), Q(w_1, m_1)),$$

in which the dependence of q on m_i and ol Q_{jm} on w_1 cannot in general be ignored.

We further observe that what we have called a single action Qj is in fact a (theoretically infinite) family or schema of actions, generated by the parameter *x*, the nominal distance the robot is to move. We loosely refer to such a parameterized family as a single action, using Q j as a shorthand notation for the family $Q_{J;Z}$, where z is the set of parameters defining the tamily. Note that the functional dependence of $Q_{j;z}$ on z together with Wi and m_i may bo arbitrarily complex. In an actual robot system, of course, a single action routlne implements such a family oi actions, receiving the parameters as arguments when called.

Operators and their Posslble Outcomes

We have just seen that there are three possible sources of variation in the final states of M and W resulting from an action Qj: the implicit dependence of Q_J on a parameter, its functional dependence on' the initial states, and the possibility that Q_J is a probabilistic relation, rather than a single-valued function. Correspondingly, an operator O_J has the form of a family of functional relations from M to M, generated by a set of parameters z. It is reasonable to take the parameter set for an operator as being identical to those for the corresponding action.

We distinguish different types oi operators according to the number and nature of their specified outcomes. A simple operator is single-valued, i.e., a function: $O_1(m_1) = m_f$. A compound operator is a multiple-valued relation over part or all of its domain: $O_J(m_1) = \{m_{f1l}, m_{f2}, \ldots\ldots\ldots, m_{fn}\}$. A compound operator expresses the robot system's anticipation that, when action Q is applied with the model in state m_i, the resulting state will be mf1 or mf2 or ... or mfn, without attaching any measures of likelihood to the alternate outcomes. A complex operator is a multiple-valued relation for which likelihood estimates have been attached to the alternate outcomes, using a probabilistic (or fuzzy or modal) formalism.

Furthermore, the operator functions and relations may be partial functions and relations, defined over proper subsets of M. The domain over which a given operator is defined represents the set of states of the model in which the robot considers that operator to be applicable. We shall subsequently observe the use of such criteria of applicability in the planning process.*

The domain-defining formulas are called preconditions in lief. 6.

Just as there is no simple relationship inherent between states of the world and states of the model, so there is none between the functional form of an action and that of its associated operator. On one hand, operators will tend to be quite crude in representing actions, given the present state of the art, just as models will be crude in representing the world. On the other hand, by way *oi* example, consider an action that might be represented as a simple function (if we do not scrutinize it too closely)--for example, an action go straight to location L. " In the experimenter's view, this action has a fully predictable outcome: it will succeed for states of the world in which there is no obstacle on the straight-line path from the robot 's current location L_c to L, and will fail when there is an obs tacle. We can consider several possible operator forms, any of which might reasonably be used in practice to represent tins action:

(1) A simple operator, whose outcome merely specifles that the robot js at L (thus, it always predicts success) ;

(2) A compound operator, specify i ng that the robot ends up either at L or at some unknown location U;*

(3) A simple operator, more complicated than (1) above, that places the robot at L or at U according to whether there is an obstacle on the path in the model,

('1) A complex operator that estimates the likolihood of the robot's encountering an obstacle. This est imate might be based on both information from the model and a priori estimates of the likelihood of surprises. For example, if the model indicates no obstacle, the operat or might place the robot at L with 90% probability and at U with 10% probability.

Clear!y, an important question is that of the fidellty of an operator in representing its associated action. Ideally, we want the transit lons in M generated by the operator to mirror the trans!tions in W generated by the action. This idea can be expressed formally, as follows. Let us make the simplifying assumption that the action' s effects on the world do not depend on the model--i.e., we consider that $Q_{J,W}$ is equi valent to a function jt^w

$$Q'_{J,W} : W \rightarrow W$$

Then the modeling relation $R : W \longrightarrow M$, together with the relation $Qj_{,w}$ in W, induces a relation in M. To the extent that 0 agrees with this induced relation—i.e.i to the extent that $Oj = RQ_{j,w}R^{-1}$ throughout M—we may consider that 0. faithfully represents Q_j in the context of the modeling relation R. Periect agreement would mean that Oj tells the robot as best it can, confined to the language of M, what Qj will do in W.*

(The foregoing is merely the nucleus of a formal theory of computer representation of actions, and we have skipped over the details. To properly develop such a theory would require the treatment of several topics, including the proper definition oi the inverse of R, the establishment of measures and metrics in W, and the extension oi all the pertinent concepts to the probabilistIC—or fuzzy or modal--case. We suggest that the development of such a theory of representations of the world may be an interesting and rewarding endeavor. To carry It further here would be beyond the scope and aims of this paper) ,

The Form of an Operator Description

The foregoing discussion of an operator as a mathemati cal relation possesses full generallty. It fails, however, to take into account the practicali ties of computer implementation of operators. To this end, we redescribe the operator description in more convenient operational terms.

It is convenient to break the operator description into the following components :

- The name of the operator (and, synonymously, of its associated action);
- Its parameters, if any (in which case the operator is actually a schema),
- Specification of the domain of applicability of the operator;
- Speciflcation of the value of the operator at each point (state) in the domain. 11 the operator has multiple outcomes, this becomes a multiple specification, with appropria te measures of probabillty (in the case of a complex operator) attached to each branch .

The output specification(s) may have an explicit functional dependence on the domain, and both of them may depend explicitly on the paramet ers.

In the present development of the SRI robot system, the model is an uns tructured collect ion of relatively simple entri es—namel y , axioms i n the first-order predicate calculus. The specif*i*cation of the domain of an operator takes the form of a statement in the predicate cnlculus, which we call the precondition(s). The domain oi the

The operator description might specify that U is constrained to be of the form $[\alpha L_c + (1 - \alpha) L]$, $0 \leq \alpha \leq 1$,, if the robot system can handle such infor mation.

Ideally, in addition, the effect Q of the action on the model would also equal $RQ_{J,W} R^{-1}$. That is, at execution time the action would update the model to keep It correctly descrlbing the world, exac tly as predicted by the ideal operator.

operator then consists of all states (axiom sets) in the model space in which the precondition state" ment is provable as a theorem.

An (individual) outcome of an operator is expressed as a set of changes to be made in the model, in the form of an add list and a delete list, describing the additions to and deletions irom the model. The reader is referred to Ref. 6 for examples of model entries, precondition expressions, and add- and delete-expressions.

PLANS AND PLANNERS

A planner is a robot system component that, in its normal mode of operation, takes three inputs:

- An initial state of the model, m_a (often the current state of the robot^{f}s model),
- A set of operator descriptions;
- A goal specification, g_o.

The goal specification defines or induces a set Mg of model states, the goal states, in which the specification g_O is valid. (For example, the goal specification may be a formula in the predicate calculus, and M_g is the set of states--].e., axiom sets—in which g_O is derivable as a theorem.)

The output of a planner is a plan. Intuitively, we think of a plan as a sequence of operators $(O_1, \ldots, O_\Omega)$ with instantiated parameters, causing state transitions in the model space M leading irom the initial state m_a to a goal state:

$$\begin{tikzcd} m_a \arrow[r, "O_1"] & m_b & \cdots & m_y \arrow[r, "O_\Omega"] & m_z \in M_g \end{tikzcd}$$

However, our actual definition of a plan generalizes this intuitive concept in several ways: a plan need not begin in the specified initial state, it may not succeed in reaching a goal state; it may consist of a tree or a more complicated directed-graph structure, it may inciude operators with multiple outcomes; and the nodes of a plan are not single states of the model, but subsets of M.

The definition that follows is assumed to be taken in the context of a given model space, M, and a givon set o1 operators, 0.

A plan is a colored, directed graph that satisfies the following four conditions.

(1) Each arc oi the graph is colored (labeled) with an operator $O_J \in$ O,or a parameterized operator schema.

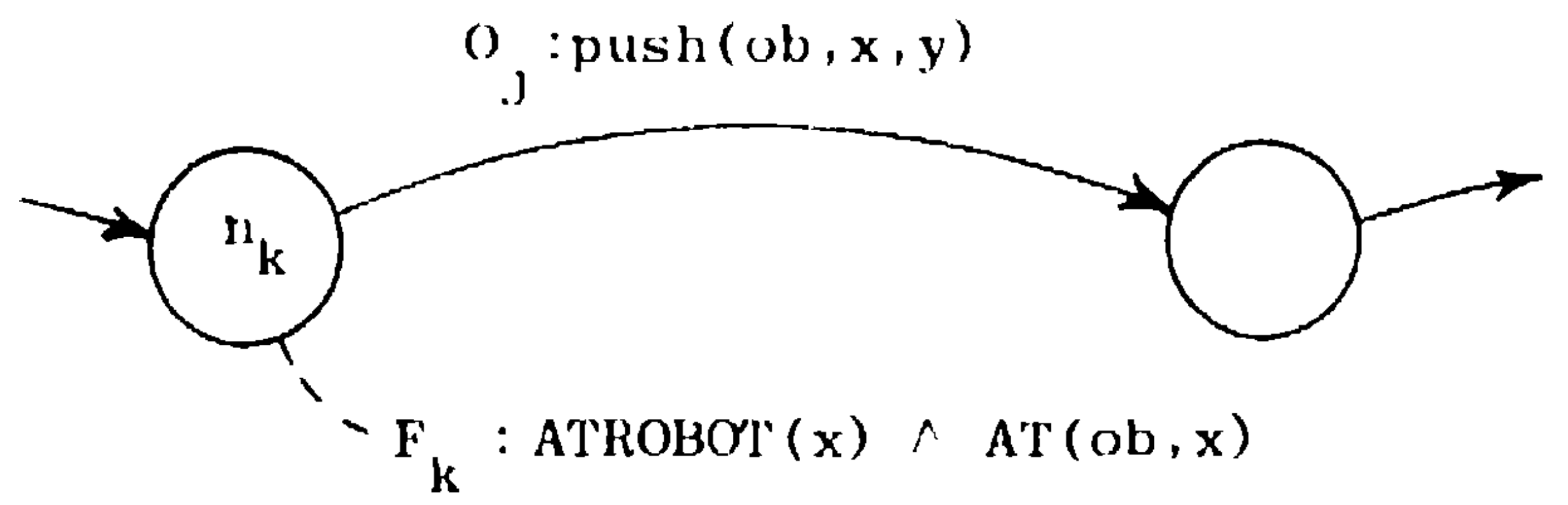
(2) To each node nk of the graph is attached a formula F_k which in turn specifies a subset M_k of the model space M.

(3) Only arcs of a single color emanate from a single node.

(4) The state set M, at a node is contained in the domain of the operator 0, coloring the arc(s) emanating from the node; or, equivalently, F, implies the preconditions of 0, .

Condition (1) allows steps of a plan (arcs) to be fully specified or to have free variables, which may reflect either don't-care conditions or goneralizations of an instantiated plan. Plan generalization is a fundamental and important process for learning in a robot system. We hope to give the SRI robot the ability, once it has generated a plan for a specific situation, to generalize the plan to refer to arbitrary objects, locations, etc. and to store the generalized plan in the form of a new meta-action routine and meta-operator with an appropriate operator description.

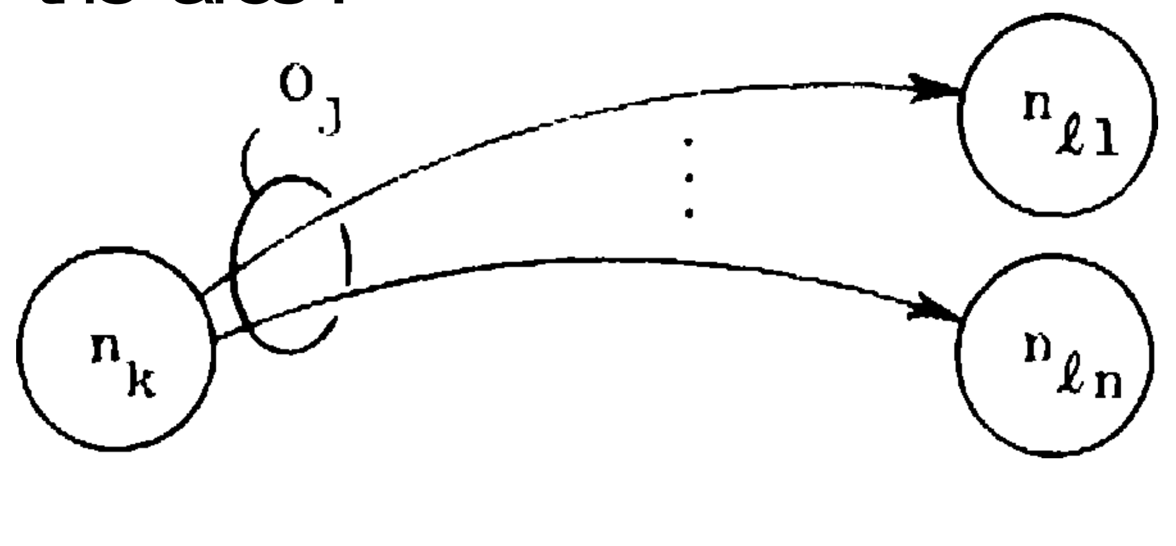
As an illustration of Conditions (1) and (2), consider a plan for the SRI robot that includes the fragment



where the operator schema 0 indicates the robot pushing any object ob from any location x to any location y. Then the predicate-calculus formula Fk induces a set of states M_k in M; namely, those states (i.e., sets oi axioms) in which an instance of F_k can be deduced. These are just the states in which the robot and some object are at the same place. Note that the state set M_k is generalized from a single state in two important ways. First, F_k has parameters (ob, x) corresponding to the parameters of Oj, so that the plan is generalized and can be applied to any object at any location. Second, the bulk of the state-delining inlormation is treated as don't-care information: the applicability of the plan does not depend on whether the robot's TV camera is on, etc. Thus, in general, M, is an (infinite) family of states reflecting the expansion of all the don't-care conditions.

A directed graph is a collection of nodes (vertices), connected by arcs (edges) each of which can only be traversed in one direction, defined as "forward". If a label from a set of labels (here, the robot operators) is attached to each arc, we call the graph colored and call the labels the colors. Proceeding in the forward direction, we say an arc emanates from its predecessor node and points to its successor node. Node n_s in the directed graph is accessible from node n_r if there exists a connected path of forward traversals along arcs leading from n_r to n_s.

Condition (3) means that at any point (node) in a plan, the plan will unambiguously specify to a robot executive what action to invoke next. If an operator is parameterized—i.e., an operator schema it is assumed that the parameters will be bound to specific values in the model at the time of execution. Multiple arcs emanating from a node signify multiple possible outcomes of the operator that labels the arcs:



These characterize compound and complex plans, described below.

Condltion (4) constitutes a basic check on the semantics of the plan. If the robot's model is in a state m that is a member of a state set Mk , so that we could say the robot is "at" node nk, in the plan, this condjtion guarantees that the opera tor O1 emanat ing from node nk is applicable to the state m.

Translated into execution-time terms, this means the following: If a robot exocutive is at point nk in the execution of n plan, and if the state of the robot model at that time is a member of M_k, then insoiar as the robot can tell, it should be proper to invoke the action routine Qj corresponding to Oj . More precisely, assuming that the model state correctly represents the world state, and assuming that the operator description fai thfully represents the action routine, then the conditions for successful application of the action routine should be met.

The robot's executive can thus monitor the execution ot a plan by comparing the robot's model after each action against the appropriate state set(s) Mk in the plan. For this reason, the Mk's are called monitor sets, and the F_k's, monitor iormulas.

Complete and Incomplete Plans

The foregoing definition of a plan has been made quite broad, in anticipation of the day when a robot might maintain large, complicated plans, of which only fragments might be required in specific instances at execution time. Our main interest, however, is in the use of a plan (or an appropriate fragment of a larger plan)* to carry the robot from a specific initial state to a specific goal. We say that P is a complete plan from state m_a to goal go if P contains a subgraph P such that:

(1) P' is a plan,

(2) All of P' is accessible from a node n_r such that $m_a \in M_r$;

(3) For at least one node n_s of P , Fs implies g

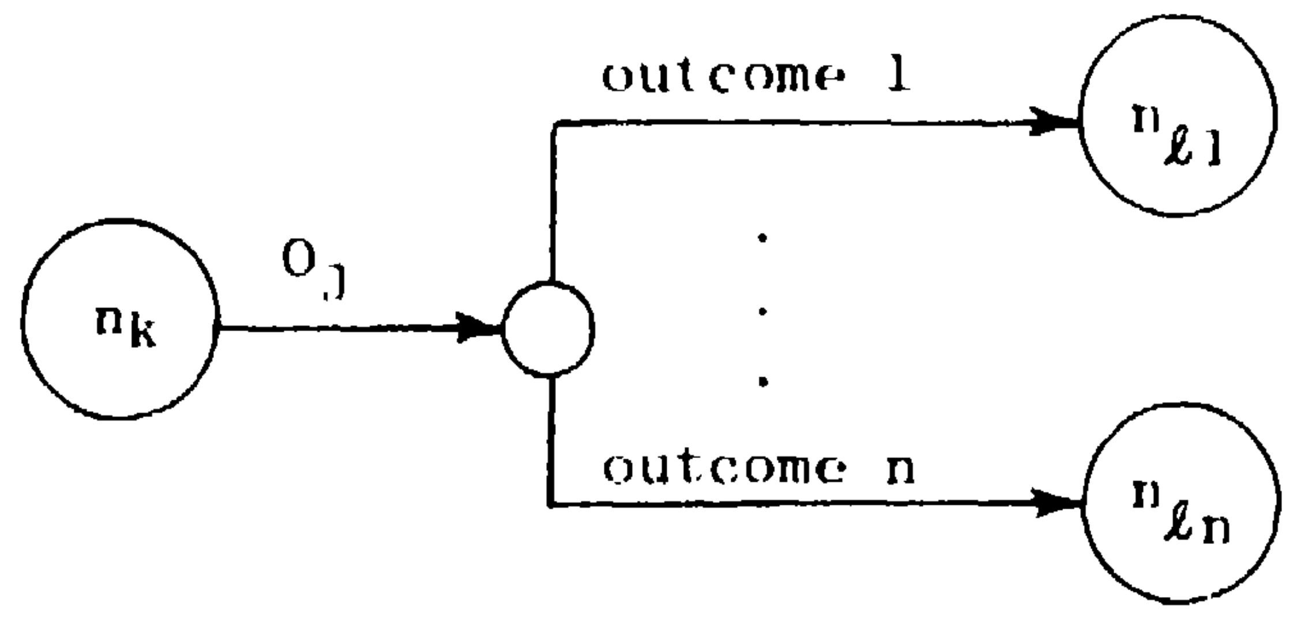
A plan P that does not satisfy those conditions is incomplete (in the context of a given initial state and goal).

Typically, a planner is provided with an initial state and a goal, and its objectlve is to create a complete plan. The output of the planner, however, may fail to satisfy either or both of (2) and (3) above. Such an incomplete plan may still have value; in particular, the robot executive may be able to proceed with an incomplete plan that has a node n_r that includes the initial state. This topic is resumed in a subsequent section on the robot executive.

Simple Compound, and Complex Plans

We define a simple plan as one containing only simple operators–hence, a graph with only a single arc emanating from each nonterminal node. A compound pi an is one including compound operators (and, possibly, simple operators) , hence, multiple arcs of a single color may emanate from a node. Finally, a complex plan is one that also includes complex operators, in which estimates ot like]ihood are attached to the arcs representing multiple outcomes .

(In diagramming a plan, it may be more convenient to int roduce auxillary nod es on t he compound and complex operators, so that a single arc of a given color emanates from each state node:



The state-transit!on graph then takes on the appearance of a game graph, in which chance or the unknown makes a play at the auxiliary nodes.)

In most problem-solving work to date, the task presented to the planner has been to produce a simple, complete plan . The QA3 theorem prover(2, 7) at SRI, when used as a robot pianner, reports success only when it has produced a completo plan--i.e., proved the theorem representing the goal. QA3 does have a rudimentary capability to act as a compound planner by using operator-description axioms of the form (0, applied to m_i implies mf or mf') and to proceed from both resultant states to the goal.

The STRIPS Planner for the SRI Robot

The STRIPS planner(6) is currently the cornerstone of our software implementation efforts for

Note that a well-formed subgraph of a plan is itself a plan.

the SRI robot. STRIPS works in a model space of the type described herein, using a GPS-like strategy. (1) Given operator descriptions, an initial model m_a, and a goal statement g_0, STRIPS uses theorem-proving methods(2,7) to find differences between g and m_Q. Selecting an operator that may be relevant to reducing such a difference, STRIPS attempts to show (again using theorem-proving methods) that part or all of the difference can be eliminated by the application of the operator. Success in this endeavor allows STRIPS to postulate the preconditions of the operator as a subgoal to be achieved from tho initial state.

STRIPS iterates the foregoing process, dealing at any time with a problem composed of a goal (the original goal or a subgoal) together with a model state, and maintaining a planning tree of such subproblems. New model states appear in the planning tree when STRIPS finds that a preconditions subgoal is realized in a state. STRIPS applies the associated operator to that state, generating a new state which (together with the next-most-recent subgoal established along that branch of the planning tree) constitutes a new subproblem. (Space does not permit a fuller description of STRIPS here; the reader is referred to Ref. 6 for a description and examples.)

If STRIPS succeeds in advancing the initial state, by successive application of operators, to a state satisfying the goal, it has achieved a complete plan. The arcs of the plan are labeled with the operators that were applied. The nodes of the plan, however, are not the successive states calculated along the solution path. Rather, associ ated with each node is a monitor formula F_k generated as follows.

Remember that an operator Oj appears in the plan only when Its preconditions have been proved from the axiom set representing a particular state. Let Sj denote the <u>support</u> of the preconditions of Qj--I.e.i the conjunction of all the axioms actually required in the proof. Then Sj is included in F_k, the monitor formula for the node from which 0 emanates.

The foregoing is a minimal prescription for F_k, since it merely satisfies Condition (4) in the definition of a plan. An F_k can be made much stronger (more restrictive) by backing up axioms from other Sj 's downstream in the plan and conjoining them to the F_k in question. The developers of STRIPS plan to incorporate such a procedure. It is described in detail in Ref. 8. Basically, every axiom appearing in every Sj is backed up, node by node, toward the beginning of the plan as long as the axiom was not added to the model by the operator (arc) being traversed.

The result of this accumulation of axioms is to create Fk's that guarantee not merely the applicability of the forthcoming operator, but the applicability of all subsequent operators and the attainment of the goal, as long as the intervening operators (or actions) make the predicted changes to the model. In this sense, the augmented F. 's (called <u>kernels</u> in Ref. 8) serve as comprehensive tests for the applicability of a plan, beginning at any node. When a plan is stored away as a meta-operator, the Fk for the initial node (or for each node considered as a potential entry point) can act as its preconditions.

STRIPS currently runs until it produces complete plans (or until it has exhausted all possibilities, or is cut off). It should not be difficult, however, to introduce termination criteria enabling STRIPS to produce incomplete plans compatible with various executive structures, such as are discussed below.

THE EXECUTIVE

We now consider robot systems in which tho top level of the control hierarchy is represented by a system component called the <u>executive</u>. The executive can call on the planner, as a subroutine, the executive can also execute a plan by calling on actions (corresponding to operators in the plan) that cause the robot to act in the world .

In practice, the executive may communicate in various ways with an experimenter at a console attached to the robot system. For the present, however, we assume simply that the executive has had presented to it a goal statement in a problem language (e.g., the predicate calculus) that it shares with tho planner. We shall distinguish several levels of executive capability according to the sophistication with which the executive monitors the behavior of the robot and chooses between planning and action.

<u>A Classification of Executives</u>

We may catalog the various classes of robot systems according to the nature of the plans that the executive can accept from the plannei—i.e., whether complete or incomplete, and whether simple, compound, or complex—and according to whether the executive checks the model for feedback ' after each step of execution. This categorization is shown in the following table.

	Complete plans <u>only</u>	Complete or incomplete <u>plans</u>
<u>No feedback</u> (Simple plans only)		B
<u>Feedback</u>		
Simple	C	D
Compound	E	F
Complex	G	H

The letter identifiers are used in the subsequent discussion. We shall refer to a Type A system, etc., or to a Type A executive, although the basis for the distinction often lies as much in the planner as in the executive itself. Generally speaking, the systems increase in complexity as their identifiers advance through the alphabet.

The Type A executive receives a simple, complete plan and acts on it with no feedback from the actual operation of the robot. More precisely, remembering that a (simple) plan is a sequence of instantiated operators interleaved with monitor formulas, we can define a Type A executive as one that ignores the monitor formulas and blindly executes the actions corresponding to the operators in sequence. Assuming only that each action terminates within a finite amount of time and returns control to the executive, the executive will run through the entire list, implicitly assuming that the result of each action leaves the world ready for the next one. As far as a Type A executive can tell, it has successfully carried out the plan.

It should be noted that we have not prohibited an action routine working under control of a Type A executive from employing feedback in its own internal workings• An action routine may cause the robot to move, may take pictures or utilize other sensory inputs for navigation, etc., and may update the model with any amount of information that it has acquired from the world. Our definition merely prescribes that at the end of an action the Type A executive does not access the updated model to determine whether the conditions for applying the next action are met.

Although Type A executives may seem needlessly crude within the framework of this paper, they occur naturally in robot research programs because they are the easiest to implement. Having created a planner (e.g., QA3), one merely needs to make the instantiated operator list comprising a plan available to a minimal executive routine that will call the associated action routines in order. This allows the experimenter to see the robot in action without developing the more complex intercommunications demanded by more sophisticated executives. Early experiments with the SRI robot were carried out in this fashion.

We define a Type B executive as one that can accept incomplete plans from the planner but which, like a Type A executive, executes each accepted plan without feedback. A Type B system is interesting because, with a very simple executive, it can achieve a crude monitoring ability by relying on the abilities of the planner. The planner can present an incomplete plan that only attempts to achieve a portion of the original goal (for example, a single clause in a formula representing the goal). Alternatively, the plan might only specify the first operation to be applied to the initial state, or the operations up to and including the first one with a multiple outcome. After blindly executing the actions corresponding to the incomplete plan, the executive returns to the planner, presenting the new current state of the model and the initial goal as a new problem.

Assuming that the planner takes the new problem and solves it from scratch, it will effectively reestablish the validity of any unused portion of its former planning tree that it has to regenerate. If, on the other hand, unplanned outcomes of the executed actions have affected the model so as to invalidate the previous work, the planner will automatically have a revised problem to work on. In the extreme case, in which the incomplete plans on which the executive acts have only one step each, the planner has in effect taken over the job of checking the monitor sets at each step during actual execution. This mode of operation is conservative, in that the robot does not plan to execute actions that are not properly applicable. It is grossly inefficient, in that the planner will typically be redoing much of its previous computation at every step. (However, the Type B executive, like Typo A, can be a worthwhile experimental approach, in view of the human labor involved in setting up more complex executives.)

It is apparent that one could remove the major inefficiency of a Type B system by allowing the planner to retain the planning trees from its previous attempts and reestablish the validity of unused segments. This would have the same effect as some of the more sophisticated executives to be described below. In general, it may be an arbitrary matter whether a particular calculation or decision is described as being performed in the planner or in the executive, since they communicate directly with each other.

A Type C executive receives a complete, simple pi an, and proceeds to execute the plan step by step. After each execution step, however, the executive stops to see whether the monitor formula for the next step is satisfied before proceeding with it. (As suggested above, the executive might actually call upon the deductive machinery of the planner to perform this check.) As long as the checks are satisfied, the execution of the plan proceeds. If the plan is completed, the executive checks tor satisfaction of the goal condition and, if it is satisfied, reports success.

If at any point the monitor check fails, it implies that the execution of an action resulted in an unplanned model state not prescribed by the simple plan. The simplest Type C executive would merely start afresh with the now-current state and the initial goal as a new problem to be solved by the pianner, and execute the resultant plan. It is evident, however, that some port i on of the planner 's previous work may still be valuable. In particular, the successive monitor formulas of the former plan serve as inviting target goals for getting back on the track oi the plan, since it is plausible on heuristic grounds that the new state of the robot may be quite close to satisfying at least one of the monitor formulas. Thus, the planner could be called again and given an initial planning tree whose nodes contain the former monitor formulas as subgoals. If the planner can make a plan from the new state to any monitor formula, the old plan will carry the rest of the way to the goal. A procedure of this sort, for replanning with the kernels (monitor formulas) produced by STRIPS, is described in Ref. 8.

An executive that checks the model after performing each action is able to deal with alternate outcomes, hence with compound or complex plans. Type G (and Type E) executives are such; after each execution step, a Type G executive refers to the model and the plan. As long as the new model state agrees with the momtor formula at any of the successor nodes of the arc just executed, execution proceeds. If this check fails, the executive behaves as described under Type C.

Because executives of Types C, E, and G require complete plans by definition, the question of when to plan and when to act is simple for these classes (as we have seen). Since a complete plan promises to carry the robot all the way to its goal—at least, assuming the right outcomes occur when several are possible--there is little point in iurther planning as long as the checks at every step show that the next action can be applied along a path to a goal state.

Executives Acting with Incomplete Plans

If, however, we allow incomplete plans that do not extend all the way to the goal,* appropriate to executives of Types D, F, and H, there is a very real question at times whether it is more efficacious to act on an incomplete plan or to continue planning. There is a risk either way. The risk involved in further planning (which will tend to emphasize the extension of the existing incomplete plan) is that, if unanticipated outcomes occurring during subsequent execution render the plan invalid, the effort is wasted. The risk in execution is that it may be leading up a blind alley in terms of attainment of the final goal. Further planning might have exposed the futility of the incomplete plan.

Thus, we are led to a formulation for Type D, F, and II executives in which planning and execution are competing activities that can be engaged in by the executive. It is assumed that both actIVIties have associated, imite costs. Planning costs real time or computer time (if it were free, the robot would of course plan everything all the time). In fact, in the present state of the art, the planning of a step may often take considerably longer than its execution. Execution also takes time, and in addition may make irreversible, undesirable changes in the world and the model. Thus, at a given point in time, the executive needs to make a cost-effective decision between planning and acting.

In the section that follows, we sketch an abstract cost-effectiveness formulation for treating robot planning and execution. It appears that this formulation is general enough to describe a broad class of robot executive systems.

A COST-EFFECTIVENESS FORMULATION FOR ROBOT EXECUTIVES

Game theory, which descends from the work of Von Neumann and Morgenstern,(9) may be characterized as the study of rational decision-making under conditions of uncertainty. Given the "philosophical" assumptions that the benefits and costs of an entity[1] possible actions relative to an environment can be quantified into units of a common measure (utility), and that the iormalism of probability theory is an adequate vehicle for representing uncertainties about the environment and the outcomes of actions, game theory provides a formalism for investigating optimum strategies for the entity (human, organization, or robot). A major goal of game-theoretic reasoning can be summarized in the concept of cost-effectiveness: choosing strategies that yield the largest expectation value of effectiveness (positive benefits) minus cost, when the two are related to each other through the measure of utility.

It we make the same philosophical assumptions about the operation of a robot in an uncertain (i.e., imperfectly modeled) environment, game theory is the natural vehjcle for finding and descnibing, in abstract terms at least, the optimum or ulli-mate" behavior strategies for a robot executive. It must be quickly admitted that, because of our primitive capabilities in quantifying real environments, there seems little hope of using the elegant abstractions of game theory as a guide to the construction of practical executives. The game-theoretic viewpoint, however, does provide a landmark and a conceptual viewpoint for comparing specific executive structures with the ultimate rational executive.

Let us sketch a cost-effectiveness-based robot executive. As usual, we distinguish between the world and the model, and between actions and operators. We still view the model as an assemblage of explicit entries, but observe that some entries may express the level of confidence or degree of uncertainty of others. For example, the robot may model not only its location, x, but the error in its location Ax, which may be incremented each time the robot moves. The presence of objects, the status of doors, etc., may be assigned levels of confidence that are increased through observations or decreased with the passage ol time.

A plan must always begin with a monitor formula satisfied in the current model state, if executing the first step of the plan is to be a realistic current option.

This is in marked contrast to most human activities, in which the comparable "planning" is so rapid and so automatic that it is often carried out on the subconscious level. At most, the human may devote conscious effort to estimating the likelihood of success along various branches of 'spontaneously generated complex plans.

A goal Is still a specification in terms of states of the model, but now a specification of the goal's utility is added. The utility of a goal must also include the concept ot timeliness; achieving a goal in a minute is clearly preferable, in general, to achieving it in an hour. The time dependence of utility may be explicit in some cases, e.g., This goal is worthless if not achieved by one o'clock. In general, however, it appears appropriate to attach a cost to the passage of time in all phases of the robot's activities. This cost of the robot' s (and the experimenter's) time could be calculated automatically by the robot's executive in considering the cost-effectiveness of every action.

Among a family of states constituting a goal, the utility may vary irom state to state, reflecting the concept that some states are better at tainment s oi the goal than others. The executive can thus judge when a goal is attained well enough to dispense with further effort. A human often does this—for example, when maneuvering a car into a parking place.

The introduction of utility for goals, with time dependence, provides the mechanism for the treatment of multiple goals. priori ties among goals, urgency, etc. The experimenter can then give the robot goal commands equivalent to Drop what you're doing and perform this task right away, you have nothing else to do, explore the environment, and so on.

In our terminology, the operators for a costel tec tive executive are complex, specifying multiple outcomes with probabilitv estimates nttached to each . In addJtion, each operator must be supplied with an estimate (or a priori estimates must be generated) ot the time that the corresponding ac tion is expected to consume . Ot her, explicit measures of the cost of the action might be provided, if, for example, the action consumed n valunble resource, such as electrical power in the case oi a self-contained robot .

Armed with such goal descr i ptions and operator descriptions, a planner in a robot system performs a search that is governed by utility, not merely probability. One can envision a STRIPS-Ukc planner, for example, maint a)ning a complex planning tree but terminating any branch of the tree for which t he accumulated costs of the actions exceed the utility of the goal. More generally, the calculation <>i expected utility for the portion of plan extant at each node in the planning t ree becomes the guidi ng measure for search . This calcula t i on must combine the costliness of the partial plan thus tar generated (i.e., the expected cost of execut ing it) with an estimate ol the costliness ot the remainder oi the plan needed to make it complete (see the discussion of the "A" search algorithm in Refs. 10 and 11.) Thus, the cost-effective planner must have some mechanism, however crude, 1 or estimating the nearness ot a plan to completeness and thus estimating 11 s progress at any point .

This leads us to the major conceptual step in the formulation of a cost-effective executive. If the progress of a planner in generating a plan can, however crudely, be estimated and dealt with as a measurable quantity, just as the progress of the robot across the room can, why not treat the planner as a kind of action routine that affects, not the external environment, but a quasi-environment whose states are plans? Furthermore, just as an action routine has an operator description that models it, so we can provide the executive with a quasi-operator description for the planner, that estimates (for a given current model state, goal, and existing accumulation of plan) the outcome of a call to the planner. By using this description to estimate the cost-effeetIveness of a call to the planner, and by comparing that with the estimated cost-effectiveness attached to any exist ing complete or incomplete plans, the executive can make a rational decision whether to plan or to act.

We note that the executive may be able to buy differing amounts of effort fom t he planner, specified in various ways, e.g., Proceed to a complete plan (with some time limit), Plan until you encounter the first application oi a complex operator, Plan only along branches wi th cumtil ative probability of occurrence greater than 0.2, Cut off all branches oi the plan whose anticipated utility falls below a certain value, and so on. Thus, a planner may be considered as a family of quasi-actions, parameterized by the condltions governing its effort when called, and the planner description should (however crudely) reflect this variability. *

It we consider the formal space in which a costef lective executive operates, It is apparent that it is at least the product ot the space of model states and the space of plans. Actions (not including the planner) cause transitions in this space by changing the state oi the model and also by trimming down the plan, i .e., obsoleting that part of the plan that is no longer relevant to the new model state. The planner, on the other hand, generally augments the plan or creates a plan where none exists, while not affectlng the st ate oi the model.

Formally, t herofore, both the actions and the planner can be described at the mota-level by

In any cost-etiective robot executive likely to be implemented in the foreseeable luture, 1 he decision when to plan and when to net will probably be implemented by an ad hoc routine that calculates the utilitles of both sides and makes a simple decision. Abstractly, however, given a complicated planner descrlption or descriptions for a family ot planners, the problem oi when and how to call a planner J n preference to ac t ing could become difficult enough to require the services of a meta-planner in the executive.

functional relations in a space whose states incorporate the model state and the state of the plan. We call this space the knowledge space of the robot, K. K has states of the form {mi ;P;C}, where *mi* is a state of the model, P is a state of the plan, and C is a set of additional control information required to specify fully the state of the executive.

The need for the control information C can be illustrated by the following examples. Suppose that the executive calls a planner, and the planner fails to create any new plan. Again, suppose that a planner is called and yields a plan, which is executed but which leaves the model in its initial state (or, perhaps, by some measure, no closer to the goal than the initial state was). In either of these cases, both the model and the state of the plan end up where they were originally. Clearly, to avoid endless looping, the executive must have some information about its recent history; we define this as the control information, C.

In practice, the control information is likely to be buried in the executive routines in the form of program control flags, or even embodied in the progress of the program's location counter. For example, a simple executive might have a flow chart that works as follows: If there is no plan for the current state, call the planner. If there is still no plan, exit. Otherwise, execute the plan. I± the goal is achieved, exit with success. Otherwise, return to the top." From the lofty viewpoint of the knowledge-space abstraction, we can discern in the flow of control an extremely simple form of control information. More important, we have a conceptual framework in which to relate this to other executive structures.

SUMMARY: PRESENT STATUS AND PROBLEMS FOR THE FUTURE

In this paper, we have presented a general formulation that we believe is appropriate to the planning, execution, and monitoring functions of a robot working in an uncertain (i.e., imperfectly modeled) environment. We feel that this formulation is general enough to encompass the ultimate rational robot executive, working under the game-theoretic doctrine of cost-effectiveness. At the same time, it has provided a specific framework within which to view the development of the STRIPS problem solver at SRI, and we feel that it will continue to serve as a guide and a check on further efforts.

STRIPS and the formulation herein grew out of a common impetus, namely, the observed inadequacies of problem solvers confined to the first-order predicate calculus for work in a dynamic problem environment. This issue is discussed by the creators of STRIPS in Ref. 6. Our response in both developments has been to choose the formalism of states and operators as the basic problem representation, leading to the use (in STRIPS) of GPS-like search methods, while retaining the calculus as a deductive mechanism within states.

It is the author's opinion, however, that any system confined to crisp, black-and-white reasoning (i.e., logical formalisms in which all formulas ultimately map into the two truth values TRUE and FALSE) will turn out to be inadequate for intelligent behavior in realistic environments. Even in the most menial tasks, the human operates in a perennially inexact and potentially uncertain environment, in which probably and maybe and might and sort of are constant companions. A robot performing similar tasks must be prepared to handle comparable contingencies, unless it works in an environment fully sanitized by the expensive process of engineering away all the uncertainties. This view motivates the several references herein to probability theory and other logics that are not two-valued, and the portrayal of a cost-effectivenoss-based robot executive (originally developed in Ref. 12).

Still, we must walk before we can run. The first implementation of STRIPS, which exists at the time of this writing (early 1971), utilizes two-valued logic. In fact, STRIPS is currently confined to producing simple plans, rather than compound ones. Nonetheless, in combining formal deductive methods with a state-space formalism, STRIPS represents a major advance in our problem-solving capability.

In the future, an attempt may be made to incorporate compound operators and probabilistic outcomes in STRIPS. It appears that the most immediate effort, though, will be devoted to further understanding the relationship between the monitor formulas and the plan structure, and using this knowledge to control the revision and generalization of plans.

At this writing, the executive to work with STRIPS and operate the SRI robot has not been coded. We may expect it to be a Type C or D or F executive, in terms of the classification discussed earlier, and to build on the ideas presented in Ref. 8. A major portion of the effort in creating an executive for the SRI robot actually lies in establishing communication with an experimenter at a computer console for the transmission of state-defining axioms, goals, responses, and miscellaneous system instructions, including the use of a limited subset of natural English.(13) A large, corollary effort is involved in programming the action routines for the robot's navigation and perception, and devising their operator descriptions. Many of these action routines call lower-level routines that control the robot vehicle, and the action routines have to have some problem-solving capability in their own right. We are trying to relate the plans or flow charts for these action routines to the concepts developed in Ref. 6 and herein.

We have alluded to certain problem areas that have been skirted in the current implementation efforts. One of these is the role of uncertainty. A second is the problem of representing or modeling complicated environments. A third is the extremely complex issue of human use of language and concepts, and their reflection in the formalisms and routines used in the robot system. (For example, our robot should perhaps " go to" an object to push it, and go to it to observe it visually, in quite different ways.) Compounded together—because they interact strongly—these three problem areas might be considered to form the hub of the study *of* *in*telli gent behavior, which will attract the efforts of AI workers for many years to come.

REFERENCES

(1) G. Ernst and A. Newell, GPS: A Case Study in Generality and Problem Solving, ACM Monograph Series (Academic Press, 1969).

(2) C. C. Green, The Application of Theorem-Proving to Question-Answering Systems," Ph.D. Dissertation, Stanford University, Electrical Engineering Department (June 1969).

(3) L. A. Zadeh, Fuzzy Sets, Information and Control, pp. 338-353 (June 1965).

(*A)* R. C. T. Lee and C. L. Chang, Some Properties of Fuzzy Logic, " Department of Health, Education, and Welfare, Division of Computer Research and Technology, National Institutes of Health, Bethesda, Maryland (1969).

(5) R. Feys, Modal Logics, Coll. de Logique Math., Series 13, Louvain (1965).

(6) R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, presented at the Second International Joint Conference on Artificial Intelligence, London, England, September 1-3, 1971.

(7) C. C. Green, Theorem-Proving by Resolution as a Basis for Question-Answering Systems, Machine Intelligence 4 , B. Meltzer et al. (Eds.) (American Elsevier, New York, 1969).

(8) R. E. Fikes, M onitored Execution of Robot Plans Produced by STRIPS," presented at TFIP Congress 71, Ljubljana, Yugoslavia, August 23-28, 1971.

(9) J. Von Neumann and 0. Morgenstern, Theory ot Games and Economic Behavior (John Wiley and Sons, New York, Science Editions, 3rd Ed., paper, 1964) .

(10) P. E. Hart et al., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Trans, on Systems Science and Cybernetics, Vol. SSC-4, No. 2, pp. 100-107 (July 1968).

(11) N. J. Nilsson, Problem-Solving Methods in Artificial Intelligence (McGraw-Hill, New York, 1971) .

(12) J. H. Munson, A Cost-Effectiveness Basis for Robot Problem Solving and Execution," Artificial Intelligence Group Technical Note 29, Stanford Research Institute, Menlo Park, California (January 1970). Available from the author.

(13) L. S. Coles, "Talking with a Robot in English," Proc. First International Joint Conf. on Artificial Intelligence, Washington, D.C., May 7-9, 1969, Donald Walker (Ed.) (The MITRE Corporation, Bedford, Massachusetts, 1969).

ACKNOWLEDGMENT

The research reported herein was sponsored by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NAS12-2221.