A COST-EFFECTIVENESS BASIS FOR ROBOT PROBLEM SOLVING AND EXECUTION

by

John H. Munson

Artificial Intelligence Group

Technical Note 29

SRI Project 8259

The research reported here was supported by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NAS12-2221.
A COST-EFFECTIVENESS BASIS FOR ROBOT PROBLEM-SOLVING AND EXECUTION

Introduction

Most, if not all, of the formalized approaches to problem-solving in Artificial Intelligence and robotics to date have been planners exclusively. That is, they deal with a domain represented by an internal computer model, and they plan -- using various methods -- a strategy of actions that is supposed to achieve a desired goal. These approaches all have the inherent property that the plan that solves the problem by the criteria of the planning system has also solved the problem from the experimenter's viewpoint. In other words, the problem domain in the experimenter's mind is the same as that in the system's internal model. When the system reports a solution to a problem, the experimenter can and does check the solution by reviewing it step by step to see if it matches "sound" reasoning done in his own mind.

In most or all such systems, the effects of the operators or procedures that may be used to form a solution are entirely known. (Otherwise, "sound reasoning" becomes difficult or impossible.)

Hence, the experimenter who deals with such a system takes a problem known to him, commonly a puzzle or board game, and codifies it in a computer model that is isomorphic to the original. When the system reports its solution to the modeled program, he is happy that it has solved "his problem."

The existence of this isomorphism, however, means that one can only give the system problems that are essentially in the domain of mathematics. They are crisp, exact -- uncertain only if they impinge on Godel's incompleteness theorem. If they purport to reflect real-world, physical problems, the models to date do so only in the most trivial, idealized fashion. If an operator is intended to do something, it will get done.
The complexity and uncertainty inherent in real-world situations and actions are simply not present. This has been the case so far with all the problem formulations given to the problem solver in my group (the QA3 theorem-prover). Of the eleven problems given to the General Problem Solver (CPS) in Ernst and Newell's recent book, only one (the monkey problem) modeled a physical problem, and that in the most idealized terms.

This aspect of the current approaches may be criticized, in my view, as a serious limitation -- in fact, an overriding one -- when the application of problem-solving to robotry is considered. It is fundamental that a robot moving in physical space will be subjected to inaccuracies and uncertainties that are beyond the representational capability of the internal model. Dr. Bertram Raphael put it nicely thus: "the ultimate data base for a robot resides not in the computer but in the actual room around the robot." (The model will, in general, be inadequate in other non-physical respects as well, but this just adds weight to the argument.) Thus, the isomorphism is necessarily broken: an internal problem solution can never be guaranteed to be an external one. Instead, the proof will be in the pudding, and we demand for the solution of our external problem that the robot execute as well as plan, that it act on elements of the plan in addition to thinking them up.

Furthermore, it is not sufficient for the system to think up a plan and then simply turn the physical robot loose on it. Because the outcomes of actions cannot be known for sure, any decent system should monitor the execution of the plan, ready to interrupt if the actual sequence of events diverges from the plan and the attainment of the goal seems unlikely.
There is another requirement. As a step toward reflecting the uncertainty of the physical situation, our desired model will come to include estimates of uncertainty and probability. When this happens, the system will no longer be able to produce a proof that a given sequence of actions will solve a problem; it can only demonstrate a probable outcome. Furthermore, in a long sequence of actions, the probability of following any particular path may become low enough that further contingency planning is not worth the effort. It is more valuable to proceed along the existing portion of the plan and find out what happens before planning further. Thus, the robot must acquire the capability to act before it has completed a plan.

In summary, I have argued that any physical robot is beset by uncertainties surpassing its model, that to experiment with the behavior of such a robot we must deal with execution as well as planning, and that the system will have to decide at times to stop planning and act and at other times to stop acting and plan. To my (admittedly incomplete) knowledge of the AI literature, this topic has not yet been touched -- beyond, perhaps, being given lip service.

A new basis is needed that allows planning and execution to be put on the same footing and related within a decision-making structure. For this basis, I have adopted a broad framework: that of cost-effectiveness, or utility theory. By representing both planning actions and execution actions as elements of strategy possessing costs and effectiveness, we achieve a conceptual framework adequate for the needs noted above. We acquire harmony with the ideas of uncertainty and probability and randomness, and with the concept of progress in an incomplete proof or execution and how to deal with it. We are able to treat sensibly the problems of multiple goals and time-varying goals, and the question of when to quit trying to solve a problem.
In short, I would say that the new framework breaks out of confines imposed by purely deductive processing. GPS took a step in this direction, with its means-end analysis and non-binary difference measures. The path ahead is being explored by probability theorists, proponents of modal logic, and students of the "fuzzy set" concept of Lotfi Zadeh. Arduous though it will be, I feel this is a path we must follow in the development of Artificial Intelligence for use in the real world.

Development of the Framework

I shall now attempt to motivate and develop a cost-effectiveness framework in which to study, describe, and hopefully even implement a robot executive. This framework begins with the notions of states, operators, and transitions in the different worlds (or spaces) viewed by the experimenter and the robot system. It then introduces the idea of effectiveness (positive value or utility, which ultimately derives from the attainment of goals) and the idea of cost (negative value or utility, which has as one of its most important sources the very passage of time), and shows how effectiveness and cost propagate through the state spaces.

(A caution and plea to the reader: I am going to be putting down a fair number of symbols and expressions, most of which won't get wrapped up into tidy equations. These are meant to serve more as shorthand and memory aids than as parts of a "mathematical" treatment. One of the great advantages of the cost-effectiveness viewpoint and the idea of probable outcomes, at least for me, is the feel of what is going on. As I try to work through examples and developments, I can almost see some sort of a mind's-eye robot taking
actions and maybe ending up one way, maybe another. And it seems to help to envision being handed a certain amount of money if a goal is achieved, losing another amount if a passage of time occurs, etc. In other words, intuition and "gedanken-ing" have been my main tools in this development. I shall try to assist your sharing my intuition, through the text. If you can achieve such intuition, and "feel" what is being described, you will understand this framework and, I hope, will believe in it. Long proofs and tedious defenses won't be necessary. If, on the other hand, you don't make your own personal association of meanings with the symbols and expressions that appear, the whole thing will probably look like an exercise in symbol-pushing and you will quit in bafflement and annoyance. The ideas are intuitive; I am trying for persuasion, rather than proof; please try to 

feel the development.)

Robot World-States and Model-States

Our fundamental postulate is that the robot and its physical surroundings are not isomorphically modeled inside the robot's computer. Accordingly, we need to distinguish between \( W \), the external world or environment of the robot, and \( M \), the robot's internal model of the world. At a given point in time, \( W \) is in some state \( W_i \) and \( M \) is in some state \( M_i \).

We can associate \( W \) with the experimenter's (presumably omniscient) view of the robot and its real surroundings. For example, the SRI robot currently operates in an environment consisting of a collection of office-type rooms and corridors, largely empty except for doorways, baseboard moldings, an assortment of large, movable wooden boxes, and perhaps some office furniture.
(Incidentally, this type of environment, and tasks such as exploring it, going
to particular places, and pushing the movable boxes, will provide the
descriptive examples and terminology throughout this paper.) \( W_i \) for this
robot would consist of knowledge of the room layout, plus specification of
the identities, x-y positions and angular orientations of the various objects
including the robot. If doors were involved, their state of openness would
be included, and so on.

We use \( M \) to denote the robot's model, a certain defined body of
information inside the robot's computer that represents the robot program's
knowledge of its situation. Given the present state of the art, \( M \) will tend
to present a very simplified and stylized reflection of \( W \). (The very reason,
of course, why we and other researchers set up such clean environments for
our robots is an attempt to create worlds so simplified that our models can
even begin to represent them.)

We will take the view that the only information about the robot's
condition that the robot program can directly access is that in \( M_i \). If the
program wishes to learn something from \( W_i \), it must invoke some sensing
operator or action operator, which will cause a state transition in \( M \)-space
and possibly also in \( W \)-space. The new information about the robot's
environment that is available to the robot program is that which appears in
the new model-state \( M_j \).

Thus, the act of perception is represented by an explicit operator, and
the vagaries of perception can be treated by the probabilistic transformation
structure that we shall develop below. Handling perception thus is part and
parcel of our recognition that the world and the robot's model of it are two
different things.
In our present plans for the SRI robot, the model M will consist of ordered n-tuples of information. The following unofficial but illustrative sample should be largely self-explanatory.

\[
\begin{align*}
(X & \text{ ROBOT } 37.6) & (X & \text{ OBJ}_1 & 50.0) \\
(Y & \text{ ROBOT } -5.0) & (Y & \text{ OBJ}_1 & 20.0) \\
(\Theta & \text{ ROBOT } 47.0) & (\text{OBTYP OBJ}_1 & \text{ BOX}) \\
(IN & \text{ ROBOT OBJ}_2) & (\text{COLOR OBJ}_1 & \text{ RED}) \\
(\text{OBTYP OBJ}_2 & \text{ ROOM}) & \phantom{\text{COLOR OBJ}_1 &} \\
(\text{NAME OBJ}_2 & \text{ JOHN'S}) & (\text{DOOROF OBJ}_2 & \text{ OBJ}_3) \\
(\text{NAME OBJ}_2 & \text{ X2060}) & (\text{STATUS OBJ}_3 & \text{ OPEN}) \\
(\text{NORTHWALL OBJ}_2 & 43.3), \text{ etc.}
\end{align*}
\]

(The reader should not be dismayed if this sample seems to raise many questions of representation. The problem of representing real situations is an extremely difficult one, which can be expected to occupy AI researchers for decades. In fact, I consider this problem -- which can also be stated as that of developing a machine epistemology for AI -- to be the central and ultimate challenge of AI research. The sample shown above is presented only for the purpose of establishing some intuitive material for future examples and discussions.)
Operators, Probable Outcomes, and Estimates

The robot has available to it a certain repertoire of operators: for example, turn, move forward, and many more. An operator, depending on its type, may or may not cause a change in $W$ (in other words, a transition from some $W_i$ to some $W_j$). Similarly, an operator may or may not cause a change in $M$.

In a purely non-physical computer program, and also in human thought, the distinction between the operator and the result it achieves tends to be blurred. On a chessboard, for example, "pawn-to-King-four" names the action and the result simultaneously.

When dealing with a robot, by contrast, we must differentiate between the operator, the change it produces in $W$, and the change it produces in $M$. When we speak of an operator as "move ahead four feet" or "go to $x = 20.6$, $y = 6.7$," we are actually naming the operator according to its nominal, or desired, result. The real robot will most certainly not move ahead four feet exactly in $W$. Given a clear path, it may move ahead a random distance described by a Gaussian distribution with a mean of 3.92 feet and a standard deviation of 0.2 feet (and turn and drift sideways randomly as well). Given an obstacle in the path, the robot may stop at any point. What happens to $M$, moreover, depends not only on what happens to $W$ but on the system that feeds information from $W$ back to $M$.

In some respects (such as whether there is an obstacle in the path) we may consider that the experimenter knows exactly what will happen. In other respects (such as the random stopping distance described above) the experimenter does not know what will happen, and we will conceptually describe his (or our) state of partial ignorance with a probability density function.
(Of course, these probability functions will often be extremely complex and beyond calculation. I take the view, however, that the probability concept is both a fruitful and a philosophically valid one (two different things) for representing partial knowledge in a decision-making situation. Throughout this development there will be many such functions named and left unexamined. Finding workable approximations or equivalent methods is the task of research. This paper aims to create a framework, not fill in all the blanks.)

We can represent diagrammatically the idea that an operator, applied to a state of either \( W \) or \( M \), will give rise to different results according to some probability:

Several points may be noted. First, we have emphasized the separateness of \( W \)-space and \( M \)-space. Second, although not shown, \( O \) may represent an instance of an operator, selected by parameters (such as "move four feet"). Third, the outcomes and their probabilities will generally depend on both \( O \) and the initial state, and perhaps on other variables in the robot system through their implicit relationship with \( O \). Fourth, one will often in practice use a grouping of final states: for example,
$W_j$ or $M_m$ may be taken to represent all outcomes of a move-four-feet operation when no obstacle is encountered, and so on. Of course, these groupings are approximations, and how to handle the compounding of such approximations is an unsolved problem. Finally, it is possible to view the probabilistic branching as "playing a game with nature." One chooses an operator $O$, and nature responds with a resulting state. Some aspects of AI research in game-playing may be applicable. However, nature here plays probabilistically, not to maximize value, as is assumed in classic game theory and in most research.

Now the robot's executive program, much more than the experimenter, will be burdened with ignorance about the outcomes of operations. Thus, the program needs to estimate the probable outcomes, and its estimates can be represented by a similar diagram:

```
      M_i
       O
      / \    / \  
E_a  E_z  E_b
/     /   /   
M_o  M_b  M_z
```

This diagram is drawn only in M-space because the program has access only to M-space; it never "sees" W-space directly.

The various estimates made by various robot programs may range all the way from simple-minded assumptions that the desired result will always occur to highly sophisticated calculations involving information from the model, learning from past experiences, and so on. The estimates may be quite accurate or totally fallacious in any given situation. They may appear as probability calculations, or in some other guise. In any case, we conceptually view any assumption made by the program about the outcome of an action as a probability estimate of this form.
Goals, Payoffs, and Time

A goal for the robot (synonymous with a problem to be solved) is represented as a state, or set of states, for the robot to achieve. Often a partial state description is given, such as telling the robot to go to a certain place, with the understanding that any state satisfying the stipulation achieves the goal. For visibility, we shall often show a goal state in W-space or M-space as (6).

It should be noted that a goal can be specified to the robot system only in M-space, since the system is not directly cognizant of W-space. Overlooking this fact (tantamount to re-establishing the isomorphism between W and M) has unfortunately helped lead some to talk of M-space specifications, for example "go to Room K2060," as if they were unique problems in W. In fact, there are as many such problems as there are robots, worlds, and starting states -- in other words, contexts or "frames" for the goal specification.

Associated with each goal -- and we shall be quite happy to accommodate multiple goals -- is a payoff $U_G$ measured in units of utility. $U_G$ represents the value to be realized by the achievement of the goal.

In the simple case, $U_G$ is merely a constant. However, one could envision more complex goal specifications, containing subclasses with differing $U_G$'s depending on the route taken to the goal, resources used, etc. (We shall see that such factors are often better expressed as costs on the way to the goal.)

Most importantly, and requiring some discussion, utility is related to time. A quick solution to a problem is considered better than a slow one, and must be made to appear so to the robot system. We all know that the familiar "exhaustive solutions" that take longer than the age of the universe are not solutions at all for our purposes.
In some instances the goal might have an explicit time constraint, such as "find a red cube within five minutes," and then the payoff would be explicitly time-varying. Usually, however, it will be natural and effective to let the payoff of the goal(s) be fixed and associate a negative utility, or cost, with the passage of time.

This cost of time is not intended to reflect the expenditure of power or other resources by the robot system; these can appear later as explicit costs in the formulation. The cost is intended to reflect the basic fact that the employment of any person or machine to perform a function generally has a cost per unit time; hence, a faster system is a better system. Experimentation with a robot system that has any capacity to schedule its own behavior should reflect this fact. Even if the model of a useful robot were discarded as a reason, the value of the experimenter's own time would lead to the establishment of such a cost.

Now it is true that most existing problem-solvers do not associate any cost with time. They work on a single problem; the problem at hand is the entire world to them; they pursue it until they succeed or demonstrably fail or the experimenter cuts off the run. But how can such a system arrange intelligently to handle multiple, coexisting goals with different priorities?

Only by being able to schedule itself can such a system perform, and this requires estimating the cost, in time, of its actions and relating the cost to the utility of its goals. Our framework will provide a system that can drop one goal, or line of action, if its prospects become bleak or another more promising one is injected. Furthermore, the system can terminate its activity by deciding that a goal is no longer worth working on. (The reader may suddenly picture himself confronted with a stubborn robot that refuses to
work on a perfectly good problem that he wants worked on in any case; if
this happens, either the robot's estimates or the assigned ratios between
goal payoff and time cost are wrong. The dubious reader is invited to
ponder this for himself.)

In a later section we will continue the discussion of utility, showing
how it can be "backed up" from state to state, using the costs and probabilities
associated with operators. But first we must examine the role of planning in
the robot system and determine the space (neither W nor M) in which the
system will be considered to operate.

Planning and the Knowledge-Space S

In the discussions above, we have provided settings for the robot's
active and perceptual operations. Actions are operators that change the state
of the world W, and very likely of the world-model M; perceptual operators are
certain ones involving the physical robot but devoted primarily to updating
information in M. We have described how the non-trivial relation between
an action and its outcomes is encompassed by describing the action with pro-
babilistic state transitions, and the non-trivial nature of perception is handled
by describing their effects on M the same way.

It remains to provide a setting for the planning (or "thinking," or
cognitive) operations of the robot program. In doing so, we propose to limit
sharply the scope of the model M, to that information which directly represents
a model of the world-state at a given instant in time. Information generated
or obtained by the robot program above and beyond what is in M will be
represented in a new space, which we shall denote as S. To illustrate, the
knowledge that the robot is in Room₁ is an element of M, but the knowledge or deduction that, if the robot invokes operator O it may then be in Room₂, is outside of M and is an element of S. *

In fact, the intuitive definition of S is that it is the space of states of knowledge of the robot program. Thinking or planning activities of the robot will generally cause a change of state in S, by adding knowledge to the system. Execution actions of the type discussed previously will in general advance M in time, thus rendering some knowledge in S obsolete and pruning the knowledge tree. Planning and execution then become related as alternative operators that can cause transitions in the new space S. We will in turn be able to discuss utilities and probable outcomes in S, thereby arriving at a rational, cost-effectiveness based framework that includes and relates robot planning and execution.

An Example of a Knowledge-Space

We will illustrate the structure and the use of the knowledge-space S by means of an example drawn at the simplest possible level. Although I believe this example is authentic in spirit, I do not claim that it is a finished product nor that it truly represents any realistic robot system. It is stripped down to the bare bones, and its purpose is to illustrate a space S as plainly as possible.

Consider a robot that is in a world-state W₀ and model-state M₀, and is to achieve a goal G. (G is a state specification in M.) For our example, we will assume that the robot is at some point within a single closed rectangular room that the room contains some boxes, and that the goal is

*Note added in proof: Thus, goals such as "explore" and "visit all rooms" are inherently outside of M. We should take the viewpoint (which the paper currently does not) that goals are state specifications in S, only some of which happen to correspond directly to states in M.
to have the robot at some other specified point in the room. The reader can visualize the world-state \( W_o \) for himself, and he can take the sample model given in an earlier section, suitably completed, as representing \( M_o \). The goal specification is

\[
(X \text{ ROBOT } X_G) \\
(Y \text{ ROBOT } Y_G)
\]

where \( X_G \) and \( Y_G \) are the co-ordinates of the goal point.

We explicitly separate planning and execution. We assume that the robot program has available to it two planning operators, A and B. Planner A, if invoked while the model is a state \( M_o \), may or may not succeed in producing a plan (denoted AA) for achieving the goal. If a plan AA is produced, and if it is executed while the model is in state \( M_o \) and the world is in state \( W_o \), the plan in turn may or may not achieve the goal \( G \). Similarly, planning operator B under the same conditions may or may not produce a plan BB, which in turn may or may not achieve \( G \).

We make two simplifying assumptions. First, we assume that the execution of a plan proceeds as an unbroken unit and hence may be considered as a single operator for our purposes. Second, we assume (somewhat unrealistically) that if an execution operator fails to achieve the goal, it leaves the world in state \( W_o \) and the model in state \( M_o \). Thus, \( M_o \) and \( G \) are the only model-states involved in our example.

(Although it is not strictly necessary for the development, the reader may find it helpful to carry a mental picture such as the following. Planner A checks whether the straight-line path from the robot's position to the goal is clear in the model. If so, A generates a plan AA which consists of a
simple turn and move forward to the goal. Planner B is a more complex algorithm for route finding among obstacles. (Various algorithms, such as Moore's method and the tangent-point graph procedure, have been investigated in our group.) B will not produce a plan BB if it thinks that no through path exists. B is more "sophisticated" than A. B will generally find a plan whenever A does, but that might not be the case if B demands a greater tolerance for skirting obstacles. Given a model that accurately reflects the world, both planners will produce only successful plans; given an inaccurate model, either one might produce the higher percentage of unsuccessful plans. We simplify further by identifying modeled success with external success: if the planned moves go to completion without an unexpected bump, we assume that the goal conditions are achieved in the model and that the robot moves close enough to the physical goal in \( W \) to satisfy the experimenter. If a bump occurs, the robot retraces its path and leaves the world in state \( W_0 \) and the model in state \( M_0 \).

Considering now the beginning of our example experiment, we observe first that the execution operators potentially specified by plans AA and BB cannot be chosen by the system because it has not thought of them yet. (If this seems somewhat foreign, it is because we are conditioned to the type of system described in the introductory section, in which successful planning implies and even constitutes successful execution.) The only operators potentially capable of changing the state of the system in S-space, that are available at the outset, are A and B.

We take the view that the system always knows, at a primitive level, which planning operators are potentially applicable and which have already been tried, in any given state \( S_1 \) in knowledge space. That is, we assume that those calculations are built into the system and done without cost.
whenever needed, rather than themselves being subject to the cost-effectiveness mechanism. This point will be discussed fully later.

In our example, we may represent the starting state of knowledge $S_1$ thus:

\[
\begin{array}{c}
S_1 \\
M_0 \\
A \\
B
\end{array}
\]

This informal diagram means that, in the state $S_1$, the current state of model-space is $M_0$ and the goal is $G$. The system currently has available to it planning operators $A$ and $B$, and no execution operators.

Suppose now that the system chooses to invoke planner $A$. Invoking $A$ will cause a probabilistic state transition in the knowledge space $S$, with two possible outcomes, according to whether or not $A$ produces a plan.

Let us examine the plan that $A$ might produce. Viewed in $M$-space, the plan has the form shown in an earlier section:

\[
\text{Plan } A \equiv \{ M_0 \xrightarrow{O_{AA}} E_{AA} \}
\]

Put in English, the plan is something like this: "While in state $M_0$, invoke operator $O_{AA}$. With estimated probability $E_{AA}$, the goal $G$ will be achieved in $M$-space. Otherwise (in this example) the state of $M$ will be unchanged."

($E_{AA}$ is the program's estimate; it may, of course, not match our own "omniscient" value $P_{AA}$ for the probability of success of the operator. A simple planner may put $E_{AA} = 1$, while we know very well that the planned action will fail sometimes.)

Now let us consider the same plan from the viewpoint of $S$-space. Here it appears as a new execution operator $AA$ which can be selected by the system. Since there is something new relative to the starting state $S$, the system
must be in a new state of knowledge $S_2$. We can draw the S-space view of the plan thus:

The interpretation of this diagram is that the application of AA while in state $E_2$ is estimated, with probability $E_{AA}$, to achieve the goal state $S_G$. ($S_G$ is defined in this simple example as any state of knowledge in which the model achieves state $C$. By our previous simplifying assumptions, the problem is then solved, and nothing else matters.) If AA fails to achieve the goal state, the system will then be in a new state of knowledge, $S_3$, in which AA has been exhausted. The appearance of crossed-out operator symbols is a reminder that they are exhausted relative to the state of knowledge in which they appear.

We may now include the starting state $S_1$ and the outcomes of the planning operator A in our diagram:

With estimated probability $E_A$, planner A will produce the state of knowledge $S_2$ in which the plan AA exists. Otherwise, A is exhausted without producing an AA, yielding a state equivalent to $S_3$.

By applying the same considerations to planner B and its plan BB, we obtain the complete "three-by-three" S-space transition diagram for our (simple!) example, which is shown in Fig. 1. This diagram shows all the possible states of knowledge, and the applicable operators at each state.
In state $S_9$, both avenues A-AA and B-BB have failed, and the problem is not solvable by the system.

In this example, the execution operators AA and BB bear a one-to-one relationship to the plans described by transition diagrams in M-space. In a more complicated system, this one-to-one relation might not hold. The essential idea is that an operator in S-space is anything that changes the state of knowledge of the system, whether it modifies the plan structure by "thought" (adding, modifying, re-evaluating, or abandoning plans) or by "execution" (which will in general prune part of the planning structure and will in any case exhaust the execution operator relative to the current state).
The Analysis of Payoff in Knowledge Space

It is not possible to assign absolute utility values to the states of a graph such as that of Fig. 1, because of the existence of closed loops with non-zero cumulative costs around the loops. (Another way of looking at it is that states such as \( S_j \) can be reached at different times by different routes, hence with different time costs.) Instead, we must deal with incremental amounts of utility, namely, payoffs. Payoffs may be established for states in several ways.

First, payoffs may be assigned to terminal states, in which the experiment ends. In our example, we assign payoff \( U_g \) to the goal state \( S_g \) in S-space, and we assign a payoff of zero to the state \( S_9 \), in which the experiment must be terminated without success.

Second, payoffs may be backed up to a state by the use of two rules. Rule 1 states that the expected payoff of applying operator \( O \) in state \( S_i \), denoted \( U_i^O \), is the average of the payoffs \( U_j \) of the possible outcomes of \( O \), less the costs of the transition, weighted according to estimated probability. Thus,

\[
\text{Rule 1: } U_i^O = \sum_j P_j (U_j - C_j),
\]

where \( P_j \) represents an estimate of the probability of reaching state \( S_j \) by invoking operator \( O \) in state \( S_i \), and \( C_j \) is the cost along that branch. (If the \( C_j \)'s are all equal, they can be represented by a single \( C \), and the formula becomes \( \sum_j P_j U_j - C \).) Note that if the robot system rather than the experimenter is doing the estimating, its probability estimates \( E_j \) are used for \( P_j \).

Rule 2 states that the expected payoff of a state is the maximum, over all operators applicable in that state, of the payoffs for each operator. Thus,

\[
\text{Rule 2: } U_i = \max U_i^O, \text{ over all } O
\]
Finally, payoffs may be evaluated (either for a state or for the application of an operator to a state) by an evaluation mechanism that uses the data describing the state and/or operator in question.

The methods above are fully analogous to those of game-playing programs and game theory, with the difference (as noted earlier) that the "opponent" behaves probabilistically rather to maximize his own utility. The branch points for probable outcomes of operators are the analogs of the alternate plies in a game tree, at which the opponent moves. In fact, I believe that a variant of game theory that deals with a "probabilistic opponent" has been developed under the name of "expectamaxing," analogous to "minimaxing."

Let us see what would be required to back up payoffs throughout the state space of Figure 1 from the terminal states. Figure 2 shows the space again, with the costs and expected probabilities of outcomes listed for each operator. (We are assuming constant costs.) Using Rules 1 and 2, the various utilities are calculated as follows. (The utilities for $S_4$, $S_7$, and $S_8$ are obtained from those for $S_2$, $S_3$, and $S_6$ by interchanging A's and B's.)

\[ U_6 = U_6^{BB} = E_{BB} U_G - C_{BB} \]
\[ U_3 = U_3^{BB} = E_B U_6 - C_B \]
\[ U_5^{AA} = E_{AA} U_G + (1 - E_{AA}) U_6 - C_{AA} \]
\[ U_5^{BB} = E_{BB} U_G + (1 - E_{BB}) U_8 - C_{BB} \]
\[ U_5 = \max(U_5^{AA}, U_5^{BB}) \]
\[ U_2 = U_2^{AA} = E_{AA} U_G + (1 - E_{AA}) U_3 - C_{AA} \]
\[ U_2^{BB} = E_B U_5 + (1 - E_B) U_8 - C_B \]
\[ U_2 = \max(U_2^{AA}, U_2^{BB}) \]
\[ U_1^A = E_A U_2 + (1 - E_A) U_3 - C_A \]
\[ U_1^B = E_B U_4 + (1 - E_B) U_7 - C_B \]
\[ U_{i} = \max(U_{i}^A, U_{i}^B) \]

A complete backup of payoff would thus require knowing the costs $C$ and expected probabilities $E$ of every operator. Within the limitations inherent in the use of these quantities, complete look-ahead (as it is called when viewed from the starting state at which a decision must be made) provides an optimum rational basis for decision-making in the face of uncertainty.

In practice, of course, complete look-ahead is generally impossible. In board games, it is often feasible to look ahead exhaustively through a few levels of branching, and to do so with precision because all options of self and opponent are known. At the tips of the look-ahead tree, unless they are terminal, evaluation is employed to establish payoff utilities, which are then backed up.

In the case of the robot state diagram, look-ahead is likely to be abandoned much sooner. The introduction of costs and probabilities, together with the knowledge that we will never in practice determine most of them beyond an educated (or uneducated) guess, will undoubtedly induce us to abandon look-ahead at an early point—even at the starting state!—and rely on evaluation of the available operators.

Thus, we are led to consider the means by which operators may be evaluated. Most simply, the payoff of an operator may be taken as a constant, or, better, a constant \(<1\) times the payoff of the goal in question. For the purpose of decision-making without look-ahead, it would suffice even to rank-order the available operators. Any program that has a fixed order of application of its operators is in effect rank-ordering them. A more powerful technique is to evaluate the expected payoff of an operator in the context of the current state. (Deciding whether an operator is
applicable to a given state is an extreme example of this.) In our robot example, the evaluation of the expected payoffs of the planners might reasonably be made to depend on the distance from the robot to the goal, the count of boxes in the room, and so on.

Conceptually, the spectrum of possible evaluators reaches all the way to those that would simulate every action of the operator being evaluated. Such an evaluator would of course be worse than useless, because it would be as complex, bulky, and costly to run as the operator it copies. Generally, the idea is for a simulation to be an inexpensive approximation to the simuland. But what if, as may often be the case, there appears to be no worthy approximation to the operator simpler than the operator itself? I offer, as an interesting topic to explore, that of letting some of the operators in the system act as their own simulations. For example, if the routines that cause motors to turn, etc. on the physical robot were temporarily replaced by dummy simulations, it would be possible actually to call an execution operator in a kind of Gedanken mode, and use the outcome of this Gedanken-experiment to evaluate the operator. The evaluation would automatically occur in the context of the present state of knowledge; the current model M, and so on. Because physical motions of the robot would be avoided, the Gedanken world could run faster than real time and thus meet the necessary requirement that the evaluator be less costly than the operator being evaluated. Furthermore, once the Gedanken world were created, any higher-level operator could be run in Gedanken mode without further ado.
Hierarchical Organizations of Spaces

In our example used in the previous sections, we portrayed a two-level hierarchy of spaces, namely, the model space M and the knowledge space S. We concentrated on S, in an attempt to show how planners, plans, and executors dealing with M could be related in a coherent framework from the viewpoint of S-space. We can picture S-space as a kind of higher-level space, or meta-space, relative to M.

Our postulated monitor, or S-space program, has a cost-effective view of each of the lower-level operators it can invoke, such as a planner. Each planner, in turn, could be viewed as having a cost-effective view of the operators that it can choose in the construction of a plan. Whether or not a given planner is actually programmed in this fashion is another matter. I am claiming that the cost-effective framework is, first, a valid and all-encompassing conceptual one for treating any decision-making system, and second, a framework in which planners at any level could be coded. I am not claiming that it is desirable or practical to do so. In fact, in view of the rather tedious and abstract nature of state-space expansions, it is probable that lower-level operators will be programmed more in pragmatic and specialized fashion than as explicit cost-effectiveness calculators.

At the higher level, that of the monitor program that deals with S, the chances are better for practical realization of a cost-effective calculator. But it must be borne in mind that the monitor itself is subject to design considerations, that our simple example tended to gloss over.

Even in our example, there was a choice (which we discussed but did not make) of how much look-ahead the monitor should perform. At one extreme, the program could look ahead all the way to the terminal nodes, as we

24
ourselves did in Fig. 2. At the other extreme, the program could perform no look-ahead. Finding itself at a given state of knowledge, the monitor could simply evaluate all the available S-space operators and choose the one with the highest estimated payoff.

We also assumed, in the example, that the monitor could always enumerate the available operators. This might not be the case in practice, and we might have to describe for the monitor a strategy for choosing which operators to choose for consideration by look-ahead and evaluation.

A further assumption in our example was that the monitor uses the probability estimates E that the planners generate. This assumption is not necessary; the monitor could in fact modify the E's or make its own entirely different estimates. The monitor's behavior could then be analogous to that of a supervisor who didn't take on faith whatever his subordinates told him about the projected success of their plans.

Another possible variation of the monitor is to allow the possibility of quitting at any point. In S-space, this amounts to including, at every state, an available operator that has zero cost and that always leads to a terminal state with zero payoff. As a consequence, the monitor will never proceed past a point at which the other operators all have negative expected payoff. This would seem to be a pretty refinement in an experimental robot.

From the foregoing paragraphs, it should be clear that the design of the S-space monitor is by no means fixed. In fact, there is an infinite family of cost-effectiveness-based monitors, not to mention all the other types of "monitors" that could be used to control the use of the lower-level operators. Then, we can envision a given S-space monitor as being merely a kind of higher-level planner, and we can picture a collection of such monitors as being subject to regulation by a "meta-monitor" operating at a higher level.
The meta-monitor might or might not be expressed in cost-effectiveness terms; if it is, we can describe it as working in a higher meta-space $S'$, for which the available operators are the S-space monitors and, possibly, direct use of lower-level planners and executors. The S-space programs might continue to exist as true monitors (in that they can invoke execution operators), or as Gedanken-monitors, or as high-level planners only, with the decision to invoke executors left to the meta-monitor.

It is thus clear that, as we add additional levels to the control hierarchy, richer and richer structures occur. Furthermore, it should be evident that there is no end to the number of levels that can be added (conceptually at least), and that from structure to structure the question of the roles played by the various levels and operators is finally a matter of choice, terminology, and concept. It is the task of robotry research to develop and experiment with such structures, toward the twin goals of achieving understanding of them and creating useful systems. This paper has offered little specific guidance for this task, but it has established the necessary conceptual framework for robot systems that act as well as plan, and has suggested how operators based on the idea of cost-effectiveness could be used at various levels within the system.
Figure 1

S-space transition diagram for the example
(Probability estimates not shown)