STANFORD ARTIFICIAL INTELLIGENCE PROJECT
Memo No. 33

June 10, 1965

THE ADVICE TAKER AND GPS

by Barbara Huberman

Abstract: Using the formalism of the Newell-Shaw-Simon General Problem Solver to solve problems expressed in McCarthy's Advice Taker formalism is discussed. Some revisions of the formalism of can and cause described in AI memo 2 are proposed.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183).

THE ADVICE TAKER AND GPS

by Barbara Huberman

The proposal of the advice taker stipulates that it should be a program which reasons verbally. This means that it can accept a verbal description of a situation, and then answer questions about the situation by manipulating the description, possibly stopping to ask for 'advice' in the process. The verbal description of the situation consists of two different parts: descriptions of the objects in the situation, and descriptions of actions which can alter the situation. Objects are defined to be entities with various properties. Actions generally require certain conditions in order to be performed, and if these conditions are satisfied, the advice taker must be capable of performing them. The advice taker does not expect to have all the properties of the objects specified for it directly; it must be capable of deducing further properties from the information given to it. However, it is not desired that it make all possible deductions immediately, for then it would run the danger of choking itself with information about particular instances of general conditions. In fact, it should decide which deductions to make in some 'intelligent' way.

Upon inspection it becomes apparent that two different problems are involved here. First, a good formal system is needed for communication with the program. This system must convey to the program information about both objects and actions. It must be simple to use and easy for the program to manipulate. McCarthy has proposed such a system in his paper 'Situations, Actions, and Causal Laws'. I will describe an implementation of his system later.

The second problem is concerned with the way the program manipulates the system. This includes its 'intelligent' use of deduction, and its performance of actions. Not too much work has been done on this problem, and the primary purpose of this paper is to show that a variant of the general problem solver can be fruitfully applied here.

Before I show how the general problem solver can be applied I would like to give a brief description of it. All of the following quotations are from 'Report on a General Problem-Solving Program', by A. Newell, J. C. Shaw, and H. A. Simon. GPS operates within a task environment described as follows:

> "GPS operates on problems that can be formulated in terms of of objects and operators. An operator is something that can be applied to certain objects to produce different objects..... The objects can be characterized by the features they possess, and by the differences that can be observed between pairs of objects. Operators may be restricted to apply to only certain kinds of objects; and there may be operators that are applied to several objects as inputs, producing one or more objects as output."

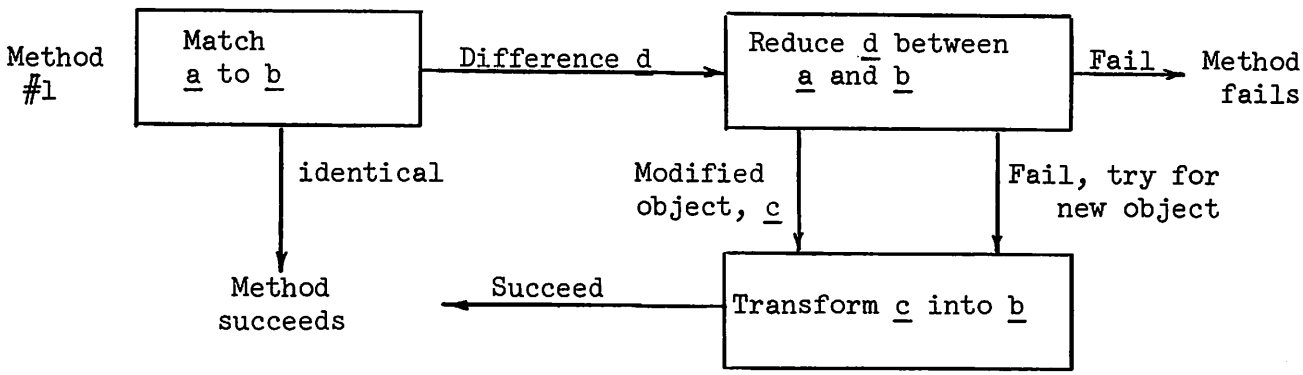
Regardless of what the task environment is, GPS attacks a problem by using an heuristic principle, the principle of subgoal reduction, and a basic heuristic system, Means-Ends analysis. The principle of subgoal reduction is:

> "Make progress by substituting for the achievement of a goal the achievement of a set of easier goals."

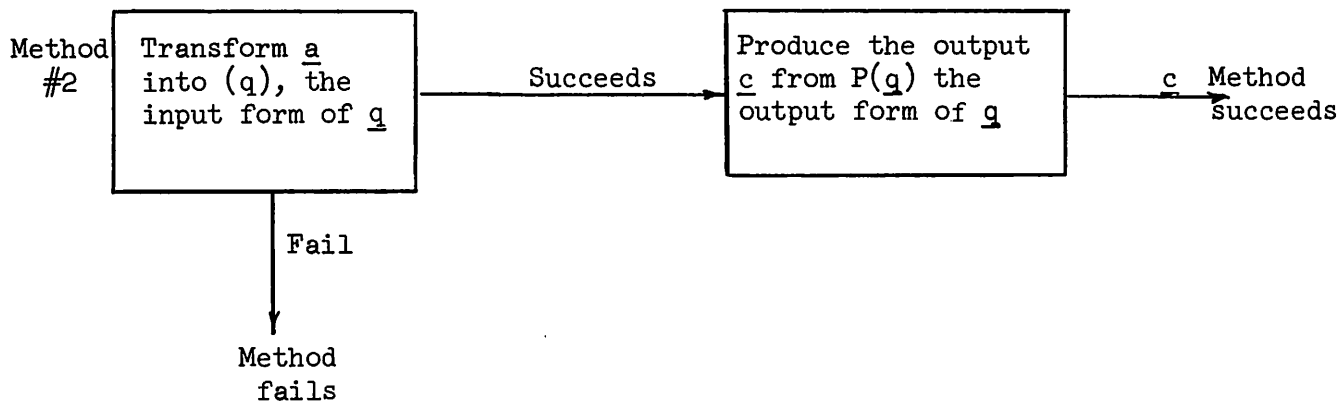
'Easier' means closer to some original object in some sense. Means-ends analysis is explained by the diagram on the next page. The analysis expects to start with a goal of type 1. Note that the analysis is not defined until a task environment is given which explains what objects, operators, differences are.

It is immediately obvious that GPS and the advice taker have at least one feature in common: the heuristics in either are separated from the formal system in which they operate. Further, the principle

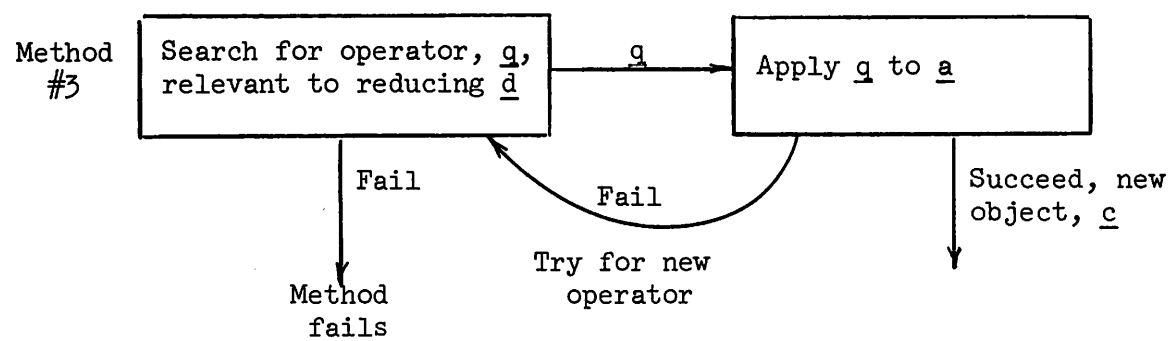
Goal type #1: Transform object a into object b



Goal type #2: Apply operator q to object a



Goal type #3: Reduce the difference, d, between object a and object b



METHODS FOR MEANS--END ANALYSIS

of subgoal reduction applies very nicely to the type of reasoning which the advice taker should be doing. By working backwards, deductions are not made until they are needed, which is a reasonably 'intelligent' way of approaching that problem. This indicates that if the description of a situation for the advice taker can be broken down into a task environment for GPS, then GPS can be used to solve the problem. Thus the question presents itself: How are the descriptions of the situations of the advice taker and the task environment of GPS related?

To answer this question I will present such a relationship. Recall that the input of the advice taker describes certain objects and actions within a certain situation.

Proposal 1: A situation is itself an object which has as properties the various objects with properties which are described to the advice taker. Let the situation consisting of all objects and properties described explicitly be called the initial situation S.

Proposal 2: The advice taker is sent into action by asking it a question about a situation. This question could be formulated as: "From initial situation S, can a situation be deduced with property P?" Let situation S* consist of all objects and properties in situation S together with the addition of the property P (this will mean adding p to the property list of some object which is a property of S).

Proposal 3: An action is described to the advice taker by telling which conditions must be satisfied before the action can be performed, and what the result of performing the action is. Thus actions can be recognized. Let ACTIONLIST be a list of

of all possible actions, each action having as a property the set of conditions which must be satisfied in order to perform the action.

Proposal 4: Let DLIST consist of all results of actions, each result having as a property the list of actions which can cause that result. A difference is defined to be any difference between two situations. DLIST thus consists of all differences which can be eliminated by performing an action.

(The appendix shows what S, S*, ACTIONLIST, and DLIST are in the case of the monkey-bananas problem.) Please note that there may be difference which can't be eliminated by taking an action. These differences come from: (a) There is insufficient information about the initial situation; or (b) The information is sufficient, but it must be deduced.

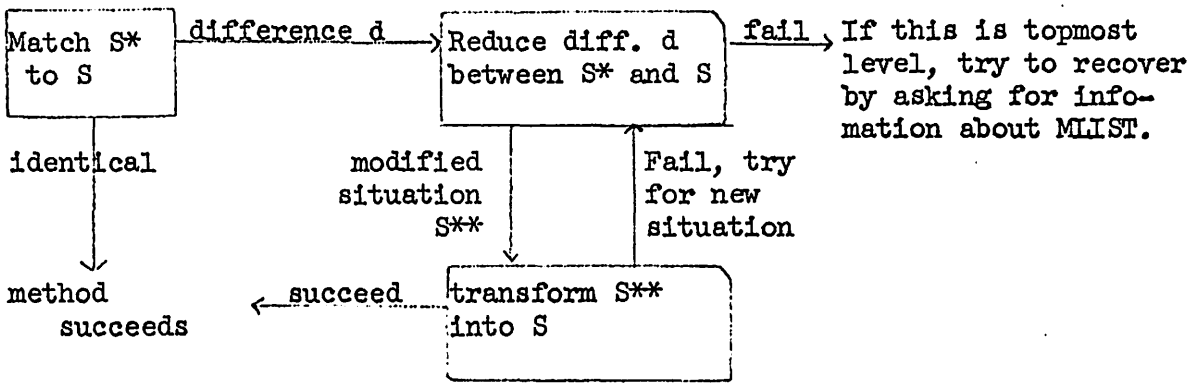
Proposal 5: For now I propose the following heuristic for handling these differences: if such a difference is encountered, assume that the program is on the wrong path, but remember what the difficulty was on MLIST. If all attempts to solve the problem fail, try to resolve the difficulty (ie., get more information) by deduction, and if that fails then ask for advice. This heuristic will probably not be useful in practice, but it will do for now.

Proposal 6: Ask the general problem solver to transform S* into S. Make DLIST and ACTIONLIST available to GPS.

The way that this adaptation of GPS works is shown by the flow-chart on the following page. It now becomes apparent that the operators of GPS are precisely the actions of the advice taker, and that GPS will actually perform an action to change a situation. The output of GPS, if success-

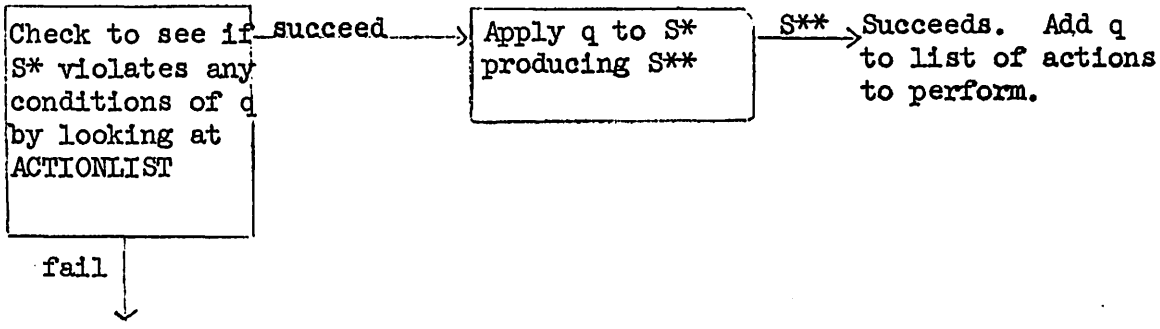
Goal Type 1. Transform situation S* into situation S.

Method 1.



Goal Type 2. Apply operator q to situation S*.

Method 2.

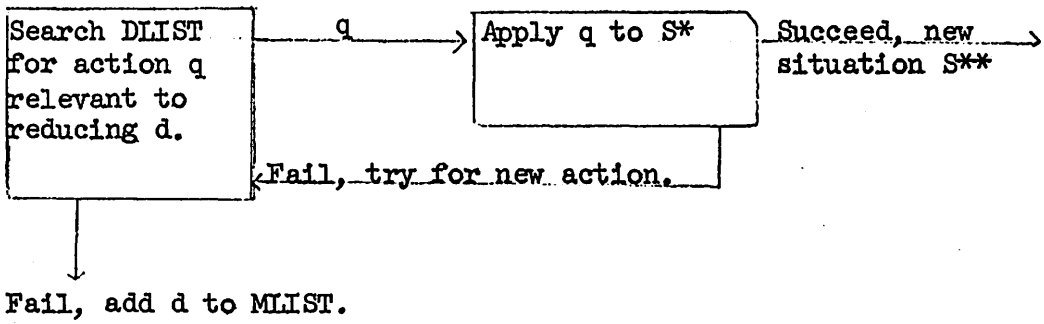


Note: 1. S* will violate a condition of q if an object of S* has a property which a condition of q says it shouldn't have.

2. S** is the same as S* except that the properties which q causes are removed, and all the conditions of q are in S**. Checking these conditions may cause a variable to be assigned a name. A record must be kept of these substitutions.

Goal type 3. Reduce difference d between S* and S by modifying S*.

Method 3.



Flow-chart of an adaptation of GPS for the advice taker.

ful, will consist of a list of the actions performed in order to convert S into S*.

I would like to turn now to the other advice taker problem: devise a good formal system for communicating with the program. I will limit my remarks to situations involving only one active party, which I will call the actor. I will take for granted the modal logic system proposed by McCarthy. However, I think McCarthy's handling of cause and can is unnecessarily complicated, and I will describe a simpler version. I will formulate no axioms, because I am not sure what the axioms should be. I will try instead to describe the meaning of cause and can.

As I understand them, cause and can are precisely the two conditions needed to define actions. Each action has certain conditions which must be satisfied before the action can be performed. This is expressed by a sentence of the form:

$$\varphi \supset can(p,\partial)$$

where φ is a set of conditions, p is the actor, and ∂ is an action. If ∂ requires no conditions, a sentence of the form:

$$can(p,\partial)$$

will express this. The advice taker program requires very little information about can. Of the axioms given for can only K1 and K3 apply to situations involving only one active party. K3, which says that if p can do ω or p can do ∂ then p can do ω or ∂, is embodied in GPS. It is used whenever more than one action is relevant to a particular difference. In this case, a choice of one action is made, while the others are held in reserve in case the choice was wrong.

Axiom K1 doesn't make sense to me, because the meaning of $\omega \supset \partial$, where ω and ∂ are both actions is unclear. If it is kept in mind that can(p,∂) has meaning only if ∂ is an action, then such axioms can be avoided.

Each action has a certain affect upon a situation. This is expressed by a sentence of the form:

$$\partial \supset \text{cause}(\varphi)$$

where ∂ is an action and φ is a set of properties. Cause only makes sense when used in this way. I will assume that performing the action ∂ while in a situation S brings about a situation S* which differs from S only as specified by φ. In other words, cause is used to define the immediate results of action; I think it is more useful to make this limitation, at least for the present. If some kind of planning is added to the advice taker at some point, then a notion of eventual causality may be useful, but that would most likely be a predicate with a different name. This eliminates axiom C1.

Another important thing about cause and can is this: they are in some sense 'meta-fluents'. What I mean is: they are fluents because they apply to situations, but they are different from all other fluents because they themselves are neither actions nor properties. They are in fact the meta-language which describes the language which the advice taker understands.

Since cause and can are neither properties nor actions, they can't apply to each other or to themselves. This eliminates axiom C2. It also seems reasonable to assume (at least for the present), that although an action may certainly have a set of results, there is no doubt about what this set is. This eliminates axiom C3. On the other hand, axiom T1 has been incorporated into the system by assuming that cause defines

immediate consequences. This becomes part of method 2 of GPS.

Finally, the advice taker is asked a question of the form: Can a situation S* in which object p has property φ be deduced from initial situation S? GPS attacks this problem as follows: If $S \neq S^*$, then look for a situation S** such that all the conditions for applying an action ∂ are satisfied by S**, and the result of applying ∂ to S** is S*. Then ask the above question about S**. This method of attack is very similar to McCarthy's axioms for canult. Eventually, the advice taker might be extended to include a question: Can a situation S* in which actor p can perform action ∂ be deduced from initial situation S. This question can be handled by entering GPS through method 2, and predicts the existence of a predicate causeult. For the present I will assume that only the first type of question is asked.

I hope it is clear by now that what I have in mind for an advice taker is a combination of a pre-processor, which handles input, and possibly asks the questions about advice, and an adaptation of GPS, as illustrated in the flow-chart. I have described the kind of input which GPS will want; the input program must produce this. The recognition of actions and properties is pretty straight-forward; the appendix gives an example of the conversion from advice taker input to GPS input. GPS is also described reasonably well in the flow-chart. I think the time has come to write the program. Of course many problems have not been faced. Among them are:

1. What is the best way to call for deduction and ask for advice? How should deductions be described?

2. How should a situation with more than one actor be handled? How should the definitions of cause and can be extended?

3. Do cause and can comprise a system which can represent all actions?

4. What additional heuristics must be incorperated in the advice taker so that it will run well? The GPS has an additional goal type concerning planning. How can this be used?

5. What kind of information may we assume the advice taker already has? How will this information be handled?

6. A situation is an object whose properties are objects which themselves have properties. Suppose these properties are objects which have properties which ... Can any use be made of this recursive structure?

Other problems will become evident as the advice taker is tried on new problems. To begin with I would like to get a program which will run the monkey-bananas problem, and then go on from there.

Appendix: The monkey-bananas problem as an example of advice taker processing.

A very simple description of the situation follows. I think it is complete but I am not sure. The description is given in list notation, as I expect it to be input to a LISP program.

1. (monkey actor)

2. (box movable standable)

3. (bananas)

4. ((under bananas) place)

Note: Sentences 3 and 4 could be replaced by (bananas onceiling) and (implies (onceiling x) (place (under x)). Then deduction would be required.

5. (implies (and (place u) (actor m) (movable b) (not (on m b)))
(can m (move m b u)))

Note: This sentence tells which conditions must be satisfied before action move can be performed. The premise (actor m) could be eliminated in a single actor situation, since (can m ∂) implies (actor m).

6. (implies (move m b u) (cause (at b u)))

Note: This sentence describes the result of performing action move.

7. (implies (and (actor m) (standable b)) (can m (climb m b)))

8. (implies (climb m b) (cause (on m b)))

9. (implies (and (at b (under x)) (on m b)) (can m (reach m x)))

10. (implies (reach m x) (cause (has m x)))

GOAL. (has monkey bananas)

The initial situation S = ((monkey actor) (box movable standable)
(bananas) ((under bananas) place))

The final situation S* = ((monkey actor (has bananas)') rest the same as S)

The difference list DLIST = (((has m x) (reach m x)) ((on m b) (climb m b))
((at b u) (move m b u)))

The action list ACTIONLIST = (((reach m x) ((at b (under x)) (on m b)))
((climb m b) ((actor m)(standable b)))
((move m b u) ((actor m) (movable b)
(place u) (not (on m b)))))

Then GPS is called via method 1, and will perform exactly as specified in the flow-chart. No deductions are required and all information is present. The possibility of the monkey climbing on the box before he moves it is prevented by the premise (not (on m b)) for move.

The output will be: (move monkey box (under bananas))
(climb monkey box)
(reach monkey bananas)

References. I looked at several papers on GPS and its uses, and several on the advice taker or pieces thereof. I will mention in particular only:

1. McCarthy, J. "Programs with Common Sense", Proceedings Symposium on Mechanization of Thought Processes, Her Majesty's Stationery Office, London, England; 1959

2. McCarthy, J. "Situations, Actions, and Causal Laws", Stanford Artificial Intelligence Project, Memo No. 2.; 1963

3. Newell, Shaw, Simon "Report on a General Problem-Solving Program", Rand Corporation Report No. P-1584; 1959

4. Newell, Simon "GPS, A Program that Simulates Human Thought", Computers and Thought, McGraw-Hill Book Company; 1963