Chapter 10

HEURISTIC PROGRAMMING: ILL-STRUCTURED PROBLEMS

ALLEN NEWELL

Carnegie-Mellon University,
Pittsburgh, Pennsylvania

Contents

We observe that on occasion expressions in some language are put forward that purport to state "a problem." In response a method (or algorithm) is advanced that claims to solve the problem. That is, if input data are given that meet all the specifications of the problem statement, the method produces another expression in the language that is the solution to the problem. If there is a challenge as to whether the method actually provides a general solution to the problem (i.e., for all admissible inputs), a proof may be forthcoming that it does. If there is a challenge to whether the problem statement is well defined, additional formalization of the problem statement may occur. In the extreme this can reach back to formalization of the language used to state the problem, until a formal logical calculus is used.

We also observe that for other problems that people have and solve there seems to be no such formalized statement and formalized method. Although usually definite in some respects problems of this type seem incurably fuzzy in others. That there should be ill-defined problems around is not very surprising. That is, that there should be expressions that have some characteristics of problem statements but are fragmentary seems not surprising. However, that there should exist systems (i.e., men) that can solve these problems without the eventual intervention of formal statements and formal methods does pose an issue. Perhaps there are two domains of problems, the well structured and the ill structured. Formalization always implies the first. Men can deal with both kinds. By virtue of their capacity for working with ill-structured problems, they can transmute some of these into well-structured (or formalized) problems. But the study of formalized problems has nothing to say about the domain of ill-structured problems. In particular, there can never be a formalization of ill-structured problems, hence never a theory (in a strict sense) about them. All that is possible is the conversion of particular problems from ill structured to well structured via the one transducer that exists, namely, man.

Perhaps an analog is useful: well-structured problems are to ill-structured problems as linear systems are to nonlinear systems, or as

I wish to acknowledge the help of J. Moore in clarifying the nature of the methods of artificial intelligence and also my discussions with my colleague H. A. Simon. The research was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-146).

stable systems are to unstable systems, or as rational behavior is to non-rational behavior. In each case, it is not that the world has been neatly divided into two parts, each with a theory proper to it. Rather, one member of the pair is a very special case about which much can be said, whereas the other member consists of all the rest of the world—uncharted, lacking uniform approach, inchoate, and incoherent.

This is not the only view, of course. Alternatively, one can assert that all problems can be formulated in the same way. The formalization exists because there is some symbolic system, whether on paper or in the head, that holds the specific, quite definite information in the problem statement. The fragmentation of problem statement that occurs when an attempt is made to explicate the problem only shows that there are serious (perhaps even profound) communication problems. But it does not say that ill-structured problems are somehow different in nature.

Thus we have an issue—somewhat ill structured, to be sure—but still an issue. What are ill-structured problems and are they a breed apart from well-structured ones? This chapter is essentially an essay devoted to exploring the issue, as it stands in 1968.

We have an issue defined. What gives life to it are two concerns, one broad, one narrow. At root, there is the long-standing concern over the rationalization of human life and action. More sharply stated, this is the challenge of art by science. In terms of encounters long resolved, it is whether photography will displace painting, or whether biology and physiology can contribute to the practice of medicine. In terms of encounters now in twilight, it is whether new products come from applied science or from lone inventors. In terms of encounters still active, it is whether the holistic diagnostic judgment of the clinical psychologist is better than the judgments of a regression equation [12]. For the purpose of this essay, of course, it is to what extent management science can extend into the domain of business judgment.

When put in this context, the issue has a charge. The concern flows partly from economics, where it is now usually labeled the problem of automation. Concern also flows from a problem of identity, in which some are compelled to ask what attributes man can uniquely call his own. As has been pointed out, probably most thoroughly by Ellul [3], it makes no sense to separate hardware and software, that is, to separate machines from procedures, programs, and formalized rules. They are all expressions of the rationalization of life, in which human beings become simply the agents or carriers of a universalistic system of orderly relations of means to ends.

Thus, viewed broadly, the issue is emotionally toned. However, this

fact neither eliminates nor affects the scientific questions involved, although it provides reasons for attending to them. Our aim in this essay is essentially scientific, a fact which leads to the second, more narrow context.

Within management science the nature of rationalization has varied somewhat over time. In the early days, those of Frederick Taylor, it was expressed in explicit work procedures, but since World War II it has been expressed in the development of mathematical models and quantitative methods. In 1958 we put it as follows:

> A problem is well structured to the extent that it satisfies the following criteria:
>
> 1. It can be described in terms of numerical variables, scalar and vector quantities.
> 2. The goals to be attained can be specified in terms of a well-defined objective function—for example, the maximization of profit or the minimization of cost.
> 3. There exist computational routines (*algorithms*) that permit the solution to be found and stated in actual numerical terms. Common examples of such algorithms, which have played an important role in operations research, are maximization procedures in the calculus and calculus of variations, linear-programming algorithms like the stepping-stone and simplex methods, Monte Carlo techniques, and so on [21, pp. 4–5].

Ill-structured problems were defined, as in the introduction of this essay, in the negative: all problems that are not well structured. Now the point of the 1958 paper, and the reason that it contrasted well- and ill-structured problems, was to introduce heuristic programming as relevant to the issue:

> With recent developments in our understanding of heuristic processes and their simulation by digital computers, the way is open to deal scientifically with ill-structured problems—to make the computer co-extensive with the human mind [21, p. 9].

That is, before the development of heuristic programming (more generally, artificial intelligence) the domain of ill-structured problems had been the exclusive preserve of human problem solvers. Now we had other systems that also could work on ill-structured problems and that could be studied and used.

This 1958 paper is a convenient marker for the narrow concern of the present essay. It can symbolize the radical transformation brought by the computer to the larger, almost philosophical concern over the nature and possibilities for rationalization. The issue has become almost technical, although now it involves three terms, where before it involved only two:

- What is the nature of problems solved by formal algorithms?
- What is the nature of problems solved by computers?
- What is the nature of problems solved by men?

We have called the first well-structured problems; the last remains the residual keeper of ill-structured problems; and the middle term offers the opportunity for clarification.

Our course will be to review the 1958 paper a little more carefully, leading to a discussion of the nature of problem solving. From this will emerge an hypothesis about the nature of generality in problem solving, which will generate a corresponding hypothesis about the nature of ill-structured problems. With theses in hand, we first consider some implications of the hypotheses, proceed to explore these a little, and finally bring out some deficiencies.

The 1958 paper asserted that computers (more precisely, computers appropriately programmed) could deal with ill-structured problems, where the latter was defined negatively. The basis of this assertion was twofold. First, there had just come into existence the first successful heuristic programs, that is to say, programs that performed tasks requiring intelligence when performed by human beings. They included a theorem-proving program in logic [15], a checker-playing program [19], and a pattern recognition program [20]. These were tasks for which algorithms either did not exist or were so immensely expensive as to preclude their use. Thus, there existed some instances of programs successfully solving interesting ill-structured problems. The second basis was the connection between these programs and the nature of human problem solving [16]. Insofar as these programs reflected the same problem-solving processes as human beings used, there was additional reason to believe that the programs dealt with ill-structured problems. The data base for the assertion was fairly small, but there followed in the next few years additional heuristic programs that provided support. There was one that proved theorems in plane geometry, one that did symbolic indefinite integration, a couple of chess programs, a program for balancing assembly lines, and several pattern recognition programs [5].

The 1958 paper provided no positive characterization of ill-structured problems. Although it could be said that some ill-structured problems were being handled, these might constitute a small and particularly "well-formed" subset. This was essentially the position taken by Reitman, in one of the few existing direct contributions to the question of ill-formed problems [17, 18]. He observed, as have others, that all of the heuristic programs, although lacking well-specified algorithms, were otherwise quite

precisely defined. In particular, the test whereby one determined whether the problem was solved was well specified, as was the initial data base from which the problem started. Thus, he asserted that all existing heuristic programs were in a special class by virtue of certain aspects being well defined, and thus shed little light on the more general case.

Stating this another way, it is not enough for a problem to become ill structured only with respect to the methods of solution. It is required also to become ill structured with respect to both the initial data and the criteria for solution. To the complaint that one would not then really know what the problem was, the rejoinder is that almost all problems dealt with by human beings are ill structured in these respects. To use an example discussed by Reitman, in the problem of making a silk purse from a sow's ear, neither "silk purse" nor "sow's ear" is defined beyond cavil. To attempt really to solve such a problem, for instance, would be to search for some ways to stretch the implicit definitions to force acceptance of the criteria, for example, chemical decomposition and resynthesis.

Reitman attempted a positive characterization of problems by setting out the possible forms of uncertainty in the specification of a problem: the ways in which the givens, the sought-for transformation, or the goal could be ill defined. This course has the virtue, if successful, of defining "ill structured" independently of problem solving and thus providing a firm base on which to consider how such problems might be tackled. I will not follow him in this approach, however. It seems more fruitful here to start with the activity of problem solving.

1. THE NATURE OF PROBLEM SOLVING

A rather general diagram, shown in Fig. 10.1, will serve to convey a view of problem solving that captures a good deal of what is known, both casually and scientifically. A problem solver exists in a task environment, some small part of which is the immediate stimulus for evoking the problem and which thus serves as the initial problem statement.[1] This external representation is translated into some internal representation (a condition, if you please, for assimilation and acceptance of the problem by the problem solver). There is located within the memory of the problem solver a collection of methods. A method is some organ-

[1] Its statement form is clear when given linguistically, as in "Where do we locate the new warehouse?" Otherwise, "statement" is to be taken metaphorically as comprising those clues in the environment attended to by the problem solver that indicate to him the existence of the problem.

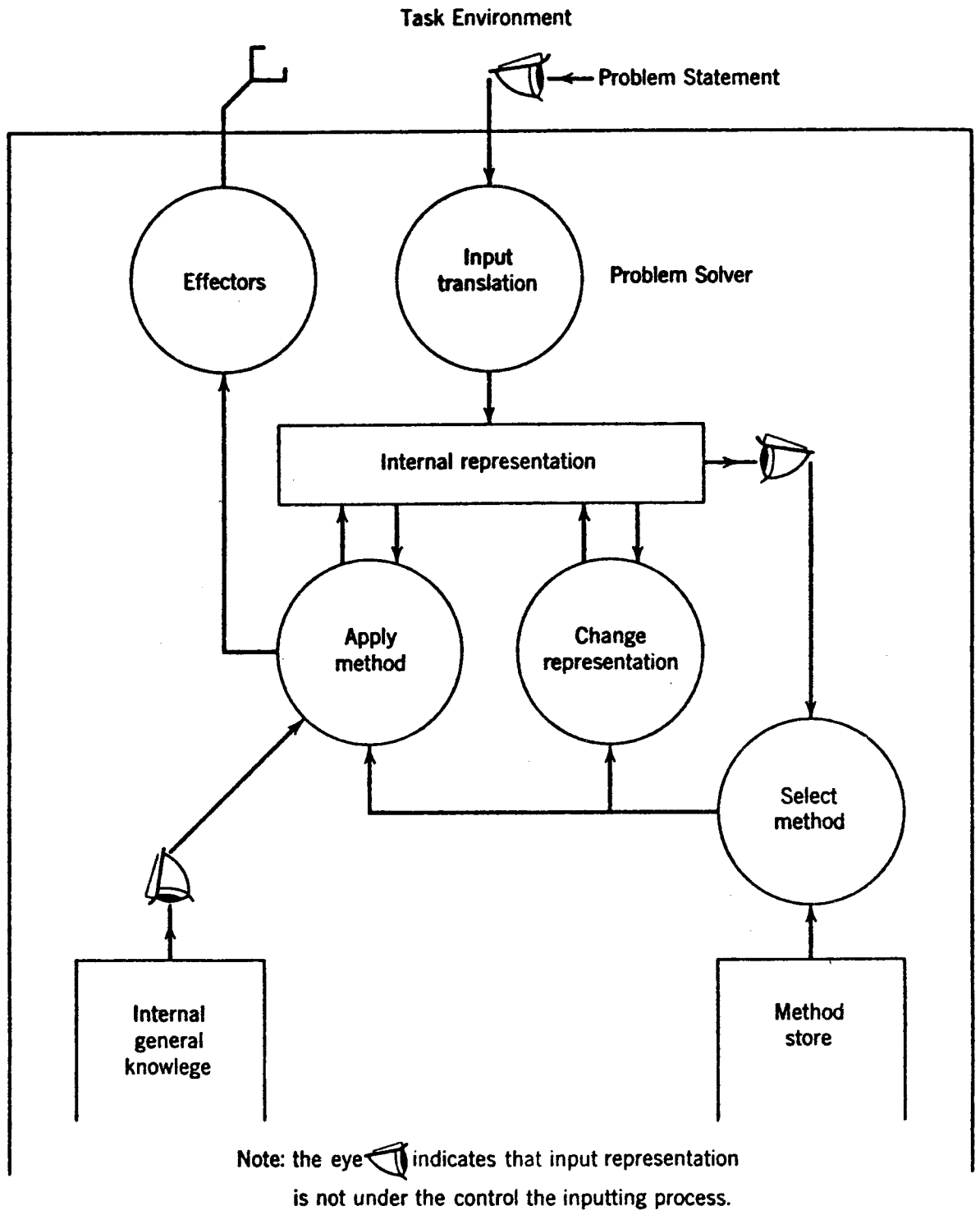

Figure 10.1. General schema of a problem solver.

ized program or plan for behavior that manipulates the internal representation in an attempt to solve the problem. For the type of problem solvers we have in mind—business men, analysts, etc.—there exist many relatively independent methods, so that the total behavior of the problem

solver is made up as an iterative cycle in which methods are selected on the basis of current information (in the internal representation) and tried with consequent modification of the internal representation, and a new method is selected.

Let us stay with this view of problem solving for a while, even though it de-emphasizes some important aspects, such as the initial determination of an internal representation, its possible change, and the search for or construction of new methods (by other methods) in the course of problem solving. What Fig. 10.1 does emphasize is the method—the discrete package of information that guides behavior in an attempt at problem solution. It prompts us to inquire about its anatomy.

1.1. The Anatomy of a Method

Let us examine some method in management science. The simplex method of linear programming will serve admirably. It is well known, important, and undoubtedly a method. It also works on well-structured problems, but we will take due account of this later. The basic linear programming problem is as follows.

Given: a set of positive real variables

$$x_j \geq 0, \qquad j = 1, \ldots, n$$

and real constants

$$a_{ij}, b_i, c_j, \qquad i = i, \ldots, m; j = 1, \ldots, n$$

maximize

$$z = \sum_j c_j x_j$$

subject to

$$\sum_j a_{ij} x_j \leqslant b_i, \qquad i = 1, \ldots, m$$

Figure 10.2 shows the standard data organization used in the simplex method and gives the procedure to be followed. We have left out the procedure for getting an initial solution, that is, we assume that the tableau of Fig. 10.2 is initially filled in. Likewise, we have ignored the detection of degeneracy and the procedure for handling it.

There are three parts to the simplex method. The first is the *statement of the problem.* This determines the entities you must be able to identify in a situation and what you must know about their properties and mutual interrelationships. Unless you can find a set of nonnegative numerical quantities, subject to linear constraints and having a linear objective function to be maximized, you do not have a LP problem, and

the method cannot help you. The second part of the method is the *procedure* that delivers the solution to the problem. It makes use only of information available in the problem statement. Indeed, these two are coupled together as night and day. Any information in the problem statement which is known not to enter into the procedure can be discarded, thereby making the problem just that much more general.

Simplex tableau

Basis	b_i	c_1 x_1	c_2 x_2	...	c_n x_n	0 x_{n+1}	...	0 x_{n+m}
x_{j_1}								
x_{j_2}								
.								
.				t_{ij}				
.								
x_{j_m}								
z_j								
$z_j - c_j$								

Procedure

1. Let j_0 = column with min $\{z_j - c_j | z_j - c_j < 0\}$;
   if no $z_j - c_j < 0$, then at maximum z, end.
2. Let i_0 = row with min $\{b_i/t_{ij_0} | b_i/t_{ij_0} > 0\}$;
   if no $b_i/t_{ij_0} > 0$, then z unbounded, end.
3. For row i_0, $t_{i_0 j} \leftarrow t_{i_0 j}/t_{i_0 j_0}$.
4. For row $i \neq i_0$, $t_{ij} \leftarrow t_{ij} - t_{i_0 j} t_{ij_0}$
   ($t_{i_0 j}$ is the value from step 3).

Define

$$x_{n+i} = b_i - \sum_j a_{ij} x_j, \qquad i = 1, \ldots, m$$

$$c_{n+i} = 0$$

$$a_{i,n+k} = 1 \text{ if } i = k, \text{ 0 otherwise}$$

Figure 10.2. Simplex method.

The third part of the method is the proof or *justification* that the procedure in fact delivers the solution to the problem (or delivers it within certain specified limits). The existence of this justification has several consequences. One, already noted, is the complete adaptation of means to ends—of the shaping of the problem statement so that it is as general as possible with respect to the procedure. Another consequence is a toleration of apparent meaninglessness in the procedure. It makes no difference that there seems to be neither rhyme nor reason to the steps of the method in Fig. 10.2. Careful analysis reveals that they are in fact just those steps necessary to the attainment of the solution. This feature is characteristic of mathematical and computational methods generally and sometimes is even viewed as a hallmark.

An additional part of the simplex method is a *rationale* that can be used to make the method understandable. The one usually used for the simplex is geometrical, with each constraint being a (potential) boundary plane of the space of feasible solutions. Then the simplex procedure is akin to climbing from vertex to vertex until the maximal one is reached. This fourth part is less essential than the other three.

The first three parts seem to be characteristic of all methods. Certainly, examples can be multiplied endlessly. The quadratic formula provides another clear one:

Problem statement:	Find x such that $ax^2 + bx + c = 0$.
Procedure:	compute $x = b/2a \pm \frac{1}{2}a\sqrt{b^2 - 4ac}$.
Justification:	(substitute formula in $ax^2 + bx + c$ and show by algebraic manipulation that 0 results).

In each case a justification is required (and forthcoming) that establishes the relation of method to problem statement. As we move toward more empirical methods, the precision of both the problem statement and the procedure declines, and concurrently the precision of the justification; in fact, justification and plausible rationale merge into one.

1.2. Generality and Power

We need to distinguish the generality of a method from its power. A method lays claim via its problem statement to being applicable to a certain set of problems, namely, to all those for which the problem statement applies. The generality of a method is determined by how large the set of problems is. Even without a well-defined domain of all problems, or any suitable measure on the sets of problems, it is still often possible to compare two problem statements and judge one to be more inclusive than another, hence one method more general than the other.

A method that is applicable only to locating warehouses is less general than one that is applicable to problems involving the location of all physical resources. But nothing interesting can be said about the relative generality of a specific method for inventory decisions versus one for production scheduling.

Within the claimed domain of a method we can inquire after its ability to deliver solutions: the higher this is, the more powerful the method. At least three somewhat independent dimensions exist along which this ability can be measured. First, the method may or may not solve every problem in the domain; and we may loosely summarize this by talking of the probability of solution. Second, there may exist a dimension of quality in the solution, such as how close an optimizing method gets to the peak. Then methods can differ on the quality of their solutions. (To obtain a simple characteristic for this requires some summarization over the applicable domain, but this feature need not concern us here.) Third, the method may be able to use varying amounts of resources. Then, judgments of probability of solution and of quality are relative to the amount of resources used. Usually the resource will be time, but it can also be amount of computation, amount of memory space, number of dollars to acquire information, and so on. For example, most iterative methods for solving systems of equations do not terminate in a finite number of iterations, but produce better solutions if run longer; the rate of convergence becomes a significant aspect of the power of such methods.

In these terms the simplex method would seem to rank as one of limited generality but high power. The restrictions to linearity, both in the constraints and the objective function, and to a situation describable by a set of real numbers, all constrict generality. But the simplex method is an algorithm within its domain and guarantees delivery of the complete solution. It is not the least general method, as is indicated by the transportation problem with its more specialized assumptions; nor is it the most powerful method for its domain, since it can be augmented with additional schemes that obtain solutions more expeditiously.

Evidently there is an inverse relationship between the generality of a method and its power. Each added condition in the problem statement is one more item that can be exploited in finding the solution, hence in increasing the power. If one takes a method, such as the simplex method, and generalizes the problem statement, the procedure no longer solves every problem in the wider domain, but only a subset of these. Thus the power diminishes. The relationship is not one-one, but more like a limiting relationship in which the amount of information in the problem statement puts bounds on how powerful the method can be. This relationship is important enough to the argument of this essay that we indicate it symbolically in Fig. 10.3. The abcissa represents increasing information in the problem statement, that is, decreasing generality. The ordinate represents increasing power. For each degree of generality an upper bound exists on the possible power of a method, though there are clearly numerous methods which do not fully exploit the information in the problem statement.

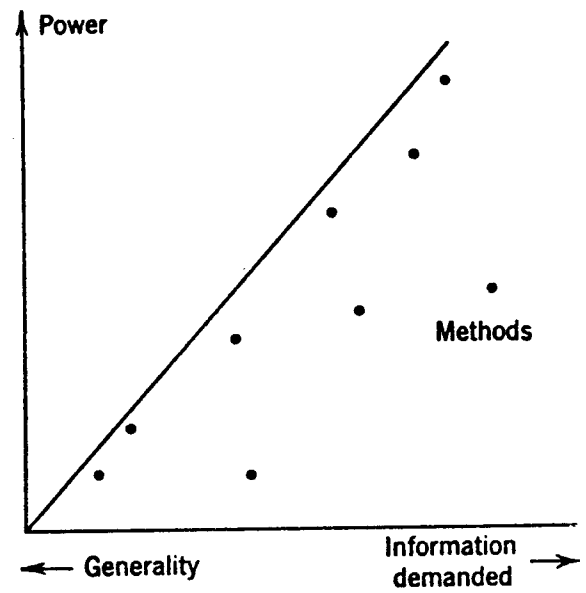

Figure 10.3. Generality versus power.

2. TWO HYPOTHESES: ON GENERALITY AND ON ILL-STRUCTURED PROBLEMS

With this view of method and problem solver we can move back toward the nature of ill-structured problems. However, we need to address one intermediate issue: the nature of a general problem solver. The first heuristic programs that were built laid claims to power, not to generality. A chess or checker program was an example of artificial intelligence because it solved a problem difficult by human standards; there was never a pretense of its being general. Today's chess programs cannot even play checkers, and vice versa.

Now this narrowness is completely consistent with our general experience with computer programs as highly special methods for restricted tasks. Consider a typical subroutine library, with its specific routines for inverting matrices, computing the sine, carrying out the simplex method, and so on. The only general "programs" are the higher-level

programming languages, and these are not problem solvers in the usual sense, but only provide means to express particular methods.[2] Thus the view has arisen that, although it may be possible to construct an artificial intelligence for any highly specific task domain, it will not prove possible to provide a general intelligence. In other words, it is the ability to be a general problem solver that marks the dividing line between human intelligence and machine intelligence.

The formulation of method and problem solver given earlier leads rather directly to a simple hypothesis about this question:

Generality Hypothesis. A general problem solver is one that has a collection of successively weaker methods that demand successively less of the environment in order to be applied. Thus a good general problem solver is simply one that has the best of the weak methods.

This hypothesis, although itself general, is not without content. (To put it the way that philosophers of science prefer, it is falsifiable.) It says that there are no special mechanisms of generality—nothing beyond the willingness to carry around specific methods that make very weak demands on the environment for information. By the relationship expressed in Fig. 10.3 magic is unlikely, so that these methods of weak demands will also be methods of low power. Having a few of them down at the very low tip in the figure gives the problem solver the ability to tackle almost any kind of problem, even if only with marginal success.

There are some ways in which this generality hypothesis is almost surely incorrect or at least incomplete, and we will come to these later; but let us remain with the main argument. There is at least one close association between generality and ill-structured problems: it is man that can cope with both. It is also true that ill-structured problems, whatever else may be the case, do not lend themselves to sharp solutions. Indeed, their lack of specificity would seem to be instrumental in prohibiting the use of precisely defined methods. Since every problem does present some array of available information—something that could meet the conditions of a problem statement of some method—the suspicion arises that lack of structure in a problem is simply another indication that there are not methods of high power for the particular array of information available. Clearly this situation does not prevail absolutely, but only with respect to a given problem solver and his collection of methods (or, equally, a population of highly similar problem solvers). We can phrase this suspicion in sharper form:

[2] The relationship of programming languages to problem solvers, especially as the languages become more problem-oriented, is unexplored territory. Although relevant to the main question of this essay, it cannot be investigated further here.

Ill-structured Problem Hypothesis. A problem solver finds a problem ill structured if the power of his methods that are applicable to the problem lies below a certain threshold.

The lack of any uniform measure of power, with the consequent lack of precision about a threshold on this power, is not of real concern: the notion of ill-structuredness is similarly vague. The hypothesis says that the problem of locating a new warehouse will look well structured to a firm that has, either by experience, analysis, or purchase, acquired a programmed procedure for locating warehouses, providing it has decided that the probability of obtaining an answer of suitable quality is high enough simply to evoke the program in the face of the new location problem. The problem will look ill structured to a firm that has only its general problem-solving abilities to fall back on. It can only have the most general faith that these procedures will discover appropriate information and use it in appropriate ways in making the decision.

My intent is not to argue either of these two hypotheses directly, but rather to examine some of their implications. First, the weak methods must be describable in more concrete terms. This we will do in some detail, since it has been the gradual evolution of such methods in artificial intelligence that suggested the hypotheses in the first place. Second, the picture of Fig. 10.3 suggests not only that there are weak methods and strong ones, but that there is continuity between them in some sense. Phrased another way, at some level the methods of artificial intelligence and those of operations research should look like members of the same family. We will also look at this implication, although somewhat more sketchily, since little work has been done in this direction. Third, we can revisit human decision makers in ill-structured situations. This we do in an even more sketchy manner, since the main thrust of this essay stems from the more formal concerns. Finally, after these (essentially positive) explications of the hypotheses, we will turn to discussion of some difficulties.

3. THE METHODS OF HEURISTIC PROGRAMMING

There has been continuous work in artificial intelligence ever since the article quoted at the beginning of this chapter [21] took note of the initial efforts. The field has had two main branches. We will concentrate on the one called heuristic programming. It is most closely identified with the programmed digital computer and with problem solving. Also, almost all the artificial intelligence efforts that touch management science are included within it. The other branch, identified with pattern

12. P. E. Meehl, *Clinical vs. Statistical Prediction,* University of Minnesota Press, Minneapolis, Minn., 1954.
13. A. Newell and G. Ernst, "The Search for Generality," in *Proc. IFIP Congress 65* (E. W. Kalenich, ed.), Spartan Books, New York, 1965, pp. 17–24.
14. A. Newell and H. A. Simon, "Programs as Theories of Higher Mental Processes," in *Computers in Biomedical Research* (R. W. Stacey and B. Waxman, eds.), Vol. 2, Academic Press, New York, 1965, pp. 141–172.
15. A. Newell, J. C. Shaw, and H. A. Simon, "Empirical Explorations of the Logic Theory Machine: a Case Study in Heuristic," *Proc. Western Joint Computer Conference,* Feb. 1957, pp. 218–230. Reprinted in Ref. [5].
16. A. Newell, J. C. Shaw, and H. A. Simon, "Elements of a Theory of Human Problem Solving," *Psychol. Rev.,* 65, No. 3, 151–166 (May 1958).
17. W. R. Reitman, "Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-Defined Problems," in *Human Judgments and Optimability* (M. W. Shelly and G. L. Bryan, eds.), Wiley, New York, 1964, pp. 282–315.
18. W. R. Reitman, *Cognition and Thought,* Wiley, New York, 1965.
19. A. L. Samuel, "Some Studies in Machine Learning, Using the Game of Checkers," *IBM J. Res. and Devel.* 3, 221–229 (July 1959). Reprinted in Ref. [5].
20. O. Selfridge, "Pattern Recognition and Modern Computers," and G. P. Dinneen, "Programming Pattern Recognition," *Proc. Western Joint Computer Conference,* 1955, pp. 91–93, 94–100.
21. H. A. Simon and A. Newell, "Heuristic Problem Solving: the Next Advance in Operations Research," *Opns. Res.,* 6, No. 1, 1–10 (Jan.–Feb. 1958).
22. H. A. Simon and K. Kotovsky, "Human Acquisition of Concepts for Sequential Patterns," *Psychol. Rev.,* 70, 534–546 (1963).
23. A. W. Tucker, "Combinatorial Algebra of Matrix Games and Linear Programs," in *Applied Combinatorial Mathematics* (E. F. Beckenbach, ed.), Wiley, New York, 1964, pp. 320–347.
24. L. A. Zadeh, "Fuzzy Sets," *Inform. and Control,* 8, 338–353 (1965).

ERRATA

Page 368 — Figure 10.1, second line under the figure should read "is not under the control of the inputting process."

Page 369 — Paragraph 1.1, line 9 should read "a_{ij}, b_i, c_j, $i = 1, \ldots, m; j = 1, \ldots, n$"

Page 371 — Third paragraph in the formula should read "Procedure: compute $x = -b/2a \pm 1/2a \sqrt{b^2 - 4ac}$."

Page 386 — Third paragraph, fifth line should read "applied to elements in the problem space produce new elements. (Operators need not"

Page 387 — Line 8 in formula should read "$q_n(q_{n-1} \ldots q_1(x_0) \ldots) = x_d$"

Page 389 — Line 10 change (m,x) to (m,t)

Page 389 — Line 4 from the bottom should read "Apply; $\xrightarrow{(m,t)}$ match $\xrightarrow{+}$ construct $\xrightarrow{x'}$

Page 395 — Paragraph 4.1., line 5 change "applicablilty" to "applicability"

Page 400 — Line 3 from the bottom change "varible" to "variable"

Page 401 — Bottom line replace (;) with (:)