What is a heuristic?

MARC H. J. ROMANYCIA
Information Services, Engineering and Planning, Gulf Canada, Calgary, Alta., Canada T2P 2H7
AND
FRANCIS JEFFRY PELLETIER
Departments of Philosophy, Computing Science, University of Alberta, Edmonton, Alta., Canada T6G 2E5

Received February 5, 1985
Revision accepted March 29, 1985

From the mid-1950's to the present the notion of a heuristic has played a crucial role in the AI researchers' descriptions of their work. What has not been generally noticed is that different researchers have often applied the term to rather different aspects of their programs. Things that would be called a heuristic by one researcher would not be so called by others. This is because many heuristics embody a variety of different features, and the various researchers have emphasized different ones of these features as being essential to being a heuristic. This paper steps back from any particular research program and investigates the question of what things, historically, have been thought to be central to the notion of a heuristic and which ones conflict with others. After analyzing the previous definitions and examining current usage of the term, a synthesizing definition is provided. The hope is that with this broader account of 'heuristic' in hand, researchers can benefit more fully from the insights of others, even if those insights are couched in a somewhat alien vocabulary.

Key words: heuristic, rule of thumb, algorithm, problem solving, artificial intelligence, cognitive science, philosophical implications of AI, history of AI.

Depuis le milieu des années cinquante jusqu'à nos jours, la notion d'heuristique a joué un rôle crucial dans les descriptions que faisaient les chercheurs en IA de leurs travaux. Ce qui n'a généralement pas été relevé, c'est que les différents chercheurs ont souvent appliqué ce terme à des aspects assez différents de leurs programmes. Ce qu'un chercheur particulier appellerait une heuristique sera nommé différemment par d'autres. Ceci, parce que beaucoup d'heuristiques incorporent une variété d'aspects différents, et les divers chercheurs n'ont pas mis l'accent sur les mêmes aspects comme étant essentiels à la formulation d'une heuristique. Cet article se tient à l'écart de tout programme particulier de recherche et examine la question de savoir quels éléments, historiquement, ont été considérés comme centraux dans la notion d'heuristique et lesquels sont en conflit. Après avoir analysé les définitions antérieures et examiné les usages courants du terme, nous proposons une définition synthétique. Notre espoir est que, disposant d'un compte-rendu plus complet sur la notion d'heuristique, les chercheurs pourront bénéficier plus pleinement des approches de leurs collègues, même si celles-ci sont formulées dans un vocabulaire quelque peu différent.

Mots clés: heuristique, règle ad hoc, algorithme, résolution de problème, intelligence artificielle, science cognitive, implications philosophiques de l'IA, histoire de l'IA.

[Traduit par la revue]

Comput. Intell. 1, 47–58 (1985)

Introduction

That the concept of a heuristic has been, and continues to be, central in AI is too well known to require documentation. Less well known, perhaps, is the fact that this central concept has always had a number of distinct "dimensions of meaning" associated with it, and throughout the history of its use in AI different theorists have emphasized different ones of these "dimensions," so that what was once thought to be a clear instance of a heuristic would later be seen as only a marginal instance. In this paper we canvas the history of this concept in AI with an eye to teasing out these "dimensions of meaning." We present four dimensions: uncertainty of outcome, basis in incomplete knowledge, improvement of performance, and guidance of decision making. A thorough investigation is then made of each dimension to see exactly where the concept of heuristic fits along each dimension. Finally, with the entire analysis behind us, we conclude by providing our own definition of 'heuristic', one which we believe accurately summarizes what the majority of AI theorists mean by the term.

Why is a solid definition needed, it might be asked. Haven't we been getting along fine without one? It is true that very few of the research efforts that employ heuristics actually offer any detailed analysis of the concept. Individual heuristics are discovered, tested, and modified in conjunction with a particular task or subtask, but the concept of a heuristic itself is rarely reflected upon. As a rule, definition by example is the primary method of introducing the concept to a newcomer. Even such noteworthy works as Lenat's (1982, 1983*a*,*b*) are not a careful exposition of the relevant concepts, but are rather a variegated mixture of hypothetical key ideas and speculations presented as an account of his (and his colleagues') latest reflections on the subject. This is no criticism: obviously such work is of the utmost value when addressing issues at the forefront of scientific research. But we think that an equally valuable task is to try to untangle the web of distinct pronouncements made about the concept *without specific reference to any ongoing research project*, both so that future researchers can find a basis for commonality in comparing their work with the apparently dissimilar work of others, and also so that newcomers to the field will be better able to comparatively judge the success of projects which employ (what their authors call) heuristics, and will also be better able to judge the extent to which any such success is genuinely due to the heuristics, as opposed to any other techniques.

History

heuriskein (ancient Greek) and *heuristicus* (Latin): "to find out, discover."

Heuretic: The branch of Logic which treats of the art of discovery or invention. 1838 Sir W. Hamilton *Logic* App. (1866) II. 230 That which treats of these conditions of knowledge which lie in the nature, not of the thought itself, but of that which we think about

> ... has been called Heuretic, in so far as it expounds the rules of Invention or Discovery.
> *Heuristic*: Serving to find out or discover. 1860 Whewell in *Todhunter's Acc. W.'s Wks.* (1876) II. 418 If you will not let me treat the Art of Discovery as a kind of Logic, I must take a new name for it, Heuristic, for example. 1877 E. Caird *Philos. Kant* II xix. 662 The ideas of reason are heuristic not ostensive: they enable us to ask a question, not to give the answer. [Oxford Dictionary of the English Language]

Minsky's (1961*b*) subject bibliography lists Polya (1945) as the earliest reference to *heuristic* in the AI literature. Of course, Polya was concerned with teaching students of mathematics "how to think," and his recommendations should be seen in that light. But it is undeniable that Polya has a profound influence on the early researchers in AI: Allen Newell, for instance, was a student of his and claims (1980, p. 1) that "Polya ... is recognized in AI as the person who put heuristics back on the map of intellectual concerns"; and Gelernter (1959; Feigenbaum and Feldman 1963, p. 135) advises his readers to consult Polya for a "definitive treatment of heuristics and mathematical discovery."

Polya's (1945, p. 113) explanation goes as follows (Polya capitalizes words that are separate entries in his dictionary):

> The aim of heuristic is to study the methods and rules of discovery and invention. A few traces of such study may be found in the commentators of Euclid; a passage of PAPPUS is particularly interesting in this respect. The most famous attempts to build up a system of heuristic are due to DESCARTES and to LEIBNITZ, both great mathematicians and philosophers. Bernard BOLZANO presented a notable detailed account of heuristic. The present booklet is an attempt to revive heuristic in a modern and modest form. See MODERN HEURISTIC.
>
> Heuristic, as an adjective, means "serving to discover."

Polya is quite definite in his view that heuristics are not infallible, and that they are to be contrasted with deductive reasoning.

> *Heuristic reasoning* is reasoning not regarded as final and strict but as provisional and plausible only, whose purpose is to discover the solution of the present problem. We are often obligated to use heuristic reasoning. We shall attain complete certainty when we shall have obtained the complete solution, but before obtaining certainty we must often be satisfied with a more or less plausible guess. We may need the provisional before we attain the final. We need heuristic reasoning when we construct a strict proof as we need scaffolding when we erect a building. ... Heuristic reasoning is often based on induction, or on analogy. [pp. 112, 113]
>
> Provisional, merely plausible HEURISTIC REASONING is important in discovering the solution, but you should not take it for a proof; you must guess, but also EXAMINE YOUR GUESS. [p. 132]
>
> It is also emphasized that infallible RULES OF DISCOVERY are beyond the scope of serious research. [p. 132]

So Polya sees himself as reviving "heuristic," the study of methods and rules of discovery. He wishes to do this in a "modest and modern form." To explain his modern version, he says

> *Modern heuristic* endeavors to understand the process of solving problems, especially the *mental operations typically useful* in this process.
>
> ... a list of mental operations typically useful in solving problems [includes] particular questions and suggestions [like:] ... WHAT IS UNKNOWN? IS IT POSSIBLE TO SATISFY THE CONDITION? DRAW A FIGURE ... CAN YOU USE THE RESULT? ... "Go back to definitions" ... COULD YOU RESTATE THE PROBLEM? [pp. 129–131]
>
> Heuristic discusses human behavior in the face of problems; this has been in fashion, presumably, since the beginning of human society, and the quintessence of such ancient discussion seems to be preserved in the WISDOM OF PROVERBS. [p. 132]

Hence to paraphrase Polya, heuristic is a science of problem-solving behavior that focuses on plausible, provisional, useful, but fallible, mental operations for discovering solutions.

The concept of heuristic began to appear in the early 1950's AI literature and was well known by the early 1960's. This was an era of providing definitions, where AI was struggling with the term and trying to absorb it into the then-current frameworks. Everyone who employed the term during this period seemed obliged to give his own interpretation of it. It was a correct thing to do on their part because the ordinary dictionary definition of the term "to find out, discover" was not being followed.

We shall now provide some definitions from this era. We have chosen our sample from the representative anthology of that time, Feigenbaum and Feldman (1963). We could have done otherwise, but all the strands we wish to pick up are present therein.

Newell *et al.* (Feigenbaum and Feldman 1963, p. 114; see also Newell 1980, p. 17) were the first to use *heuristic* as a noun meaning heuristic process. They claim to be using *heuristic* here according to the standard dictionary definition, "serving to discover or find out," but they also oppose its meaning to that of *algorithm*:

> The research reported here is aimed at understanding the complex processes (heuristics) that are effective in problem-solving. Hence, we are not interested in methods that guarantee solutions, but which require vast amounts of computation. Rather, we wish to understand how a mathematician, for example, is able to prove a theorem even though he does not know when he starts how, or if, he is going to succeed. [Feigenbaum and Feldman 1963, p. 109]
>
> One very special and valuable property that a generator of solutions sometimes has is a guarantee that if the problem has a solution, the generator will, sooner or later, produce it. We call a process that has this property for some problem an *algorithm* for that problem.
>
> A process that *may* solve a given problem, but offers no guarantees of doing so, is called a *heuristic* for that problem. [Feigenbaum and Feldman 1963, p. 114]

One gathers from this that they believe there are only two ways to solve a problem: one by thoughtlessly following a sure-fire algorithm; the other by employing complex processes (heuristics) that are genuinely creative in exploring paths to a solution. Prior knowledge of success or failure appears the key way of distinguishing these two problem-solving methods. Efficiency of either method does not appear to be a key concern.

In Gelernter's (1959) geometry program paper, we find a definition reminiscent of Polya:

> A heuristic method is a provisional and plausible procedure whose purpose is to discover the solution of a particular problem at hand. [Feigenbaum and Feldman 1963, p. 135]

Gelernter emphasizes that the necessity of avoiding algorithmic, exhaustive search is the rationale for introducing heuristics into a problem situation. Gelernter is also one of the first to point out that heuristics work in effect by eliminating options from an impractically large set of possibilities:

> A heuristic is, in a very real sense, a filter that is interposed between the solution generator and the solution evaluator. [Feigenbaum and Feldman 1963, p. 137]

This remark is noteworthy as an example of something that is common in AI: a researcher's program or theory of problem-solving influencing his conception of heuristic. Polya and Newell *et al.* spoke of a mathematician groping for a solution, but here we have posited a formal "solution generator" and "solution evaluator." These have actual counterparts in Gelernter's computer program, but we doubt if there are any such identifiable procedural components in a mathematician's thought processes.

In Tonge's (1960) discussion of his heuristic program for minimizing the number of workers needed on an assembly line, the nonguaranteed element plays a lesser role in the definition of heuristic and the filtering element is not present. He emphasizes efficiency and effort reduction in achieving a *satisfactory* solution. His definition also shows the tendency to abstract the meaning of *heuristic* away from "process" and towards any arbitrary "device." Often the "device" is a portion of his program with an identifiable function. He also speaks of heuristics as providing "shortcuts," and as employing "simplifications," in contrast with several of the algorithmic methods that theoretically guarantee solutions. His official definition is:

> ... by heuristics we mean ... principles or devices that contribute, on the average, to reduction of search in problem-solving activity. The admonitions "draw a diagram" in geometry, "reduce everything to sines and cosines" in proving trigonometric identities, or "always take a check – it may be a mate" in chess, are all familiar heuristics.
>
> Heuristic problem-solving procedures are procedures organized around such effort-saving devices. A heuristic program is the mechanization on a digital computer of some heuristic procedure. [Feigenbaum and Feldman 1963, p. 172]

Minsky (1961*a*) was one of the first to use *heuristic* in the context of "search" through a large "problem space." Speaking of chess, which Shannon had estimated to have 10^{120} paths through its game tree, he says (Feigenbaum and Feldman 1963, p. 408) "we need to find techniques through which the results of *incomplete analysis* can be used to make the search more efficient." His official definition, like Tonge's, emphasizes efficiency rather than an oppposition to algorithms:

> The adjective "heuristic," as used here and widely in the literature, means related to improving problem-solving performance; as a noun it is also used in regard to any method or trick used to improve the efficiency of a problem-solving system. A "heuristic program" to be considered successful, must work well on a variety of problems, and may often be excused if it fails on some. We often find it worthwhile to introduce a heuristic method which happens to cause occasional failures, if there is an over-all improvement in performance. But imperfect methods are not necessarily heuristic nor vice versa. Hence, "heuristic" should not be regarded as opposite to "foolproof"; this has caused some confusion in the literature. [Feigenbaum and Feldman 1963, p. 408]

Here Minsky is saying that a foolproof algorithm could be called a heuristic, provided it shows an improvement in efficiency over some other method. He is also emphasizing, like Polya, that a heuristic must be applicable to more than just a restricted set of problems. An effort-saving method that worked on only one problem would be more properly called a specific tool rather than a heuristic method.

Slagle's (1963) description of his program to solve integration problems in mathematics uses *heuristic* primarily to stand for any of a class of rules that transform a problem into one or more subproblems. Examples of such rules would be "try integration by parts" and "try a trigonometric substitution." He distinguishes algorithms from heuristic transformations, the latter being defined as follows:

> A transformation of a goal is called heuristic when, even though it is applicable and plausible, there is a significant risk that it is not the appropriate next step. [Feigenbaum and Feldman 1963, p. 197]

This particular usage, however, disagrees with his formal definition where the heuristic actually makes the decision as opposed to being a passive rule chosen by the executive:

> Although many authors have given many definitions, in this discussion a heuristic method (or simply a heuristic) is a method which helps in discovering a problem's solution by making plausible but fallible guesses as to what is the best thing to do next. [Feigenbaum and Feldman 1963, p. 192]

We will return to discuss this kind of confusion later.

Finally we come to the definition of Feigenbaum and Feldman (1963), the editors of *Computers and Thought:*

> A heuristic (heuristic rule, heuristic method) is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits search for solutions in large problem spaces. Heuristics do not guarantee optimal solutions; in fact, they do not guarantee any solution at all; all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time. [Feigenbaum and Feldman 1963, p. 6]

This definition combines many of the features present in the other definitions we have discussed. It contains the elements of lack of guarantee, of arbitrary device, of effort reduction, of eliminating options, and of satisfactory solution. Following their definition Feigenbaum and Feldman also bring up a new element, that of domain dependence. Some heuristics are very special purpose and domain specific, like chess heuristics, whereas others, like "means-ends analysis" and "planning," apply to a much broader class of problem domains.

This brings us to the end of what we might call "the early AI period." As we see it, in this period researchers were groping with the concept of a heuristic and felt compelled to provide their readers with definitions of the term. After this period, researchers no longer felt that it was such a novel concept that it required any special explanation or justification, except perhaps when talking to lay audiences. One can already see, just from the examples cited, how the concept of heuristic was transformed since its original introduction to the AI community via Polya. Polya used 'heuristic' primarily in the context of logic or psychology of discovery. His heuristic methods were to apply helpful reasoning processes like asking certain questions, drawing diagrams, guessing, looking at the problem from a different perspective, etc. Somehow these methods direct the mind towards seeing a solution. 'Discovery' is used here very

much in the sense of invention; it presumes a kind of groping exploration prior to the discovery. By the end of this early period in AI, however, 'heuristic' has been reshaped to the AI landscape. Rather than a vague psychological groping for a solution, we were presented with the notion of an exploration guided along paths in a formal problem-solving structure or space. For this reason 'discovery' is used less in the sense of exploring a previously untrodden solution path than in the sense of finding a successful path amongst those already explicitly or implicitly prespecified in the predefined state-space structure.

Another reemphasis is that, rather than having heuristic methods derive from general problem-solving psychology and be made applicable to specific domains like mathematics, in AI we have specific problem domains giving rise to their own brand of heuristic methods. Indeed, in AI the whole driving force for introducing heuristics and discovering new ones is to improve the performance of a program in a particular problem domain. In contrast, for Polya the reason to introduce heuristics was to have math students learn how to think, i.e., to acquire the type of psychology necessary to do good mathematics.

Of course there is a reason for this difference. In AI, heuristics are often born from dissatisfaction with an exhaustive algorithm, whereas for Polya heuristic techniques are applied at the very outset when investigating a totally unfamiliar problem, and their application may even result in discovering an algorithmic solution technique. For this reason 'algorithm' and 'heuristic' are not opposed for Polya; they are not in the same category of tools. Polya believes there simply are no algorithms for investigating totally new problems; this is the domain of heuristics. Algorithms, if there be any, come after we have seen one way to solve the problem and have analyzed the solution. The analysis and inventing of the algorithm is another job for heuristic methods.

In the 1960's, after the early AI era of definitions of *heuristic*, there was another usage of 'heuristic' introduced—as part of the phrase "heuristic search." So popular has this usage become that some authors, e.g., Barr and Feigenbaum (1981), prefer it to the mere "heuristic," and others do not use "heuristic" in any other form. e.g., Winston (1977).

According to Newell and Simon (1972, p. 888), in 1965 Ernst and Newell introduced the concept of "heuristic search, which itself was simply an attempt to formulate what seemed common to many of the early artificial intelligence programs." Later, Ernst and Newell wrote:

> HEURISTIC SEARCH. This research approaches the construction of a general problem-solver by way of a general paradigm of problem solving: heuristic search (Newell and Ernst, 1965). In simplified form the heuristic-search paradigm posits objects and operators, where an operator can be applied to an object to produce either a new object or a signal that indicates inapplicability.
>
> The operators are rules for generating objects, and thus define a tree of objects. ... A method for solving a heuristic-search problem is searching the tree, defined by the initial situation and the operators, for a path to the desired situation. [1969, pp. 247, 248]

As Barr and Feigenbaum (1981, p. 30) remark, 'heuristic' appears to play an odd role here. If heuristic search is just search through a tree, then even blind search is a form of heuristic search. Nowadays it is more common to call Ernst and Newell's heuristic search "state-space search" and to reserve 'heuristic search' for search through a state space that is based on heuristic decision processes. In other words 'heuristic search', as used nowadays, does not involve a totally new usage of 'heuristic'. (For examples of the modern usage, see Barr and Feigenbaum (1981, pp. 28–30), Winston (1977, p. 122ff), and Nilsson (1980, p. 72).)

After the early AI era, there are very few definitions given except when authors are writing for a primarily lay audience. In these cases the term is typically defined very superficially so as to include all the standard definitions. Samples of these are

> ... heuristic methods, i.e., features that improve the systems' problem-solving efficiency or range of capability. These range from *ad hoc* tricks for particular kinds of problems to very general principles of efficient administration and resource allocation. [Minsky 1968, p. 8]

> A heuristic is a rule of thumb, strategy, method, or trick used to improve the efficiency of a system which tries to discover the solutions of complex problems. [Slagle 1971, p. 3]

> ... "heuristic programming" refers to computer programs that employ procedures not necessarily [but possibly] proved to be correct, but which seem to be plausible. Most problems that have been considered by AI researchers are of the sort where no one knows any practical, completely correct procedures to solve them; therefore, a certain amount of proficiency in using hunches and partially verified search procedures is necessary to design programs that can solve them. So, by a *heuristic* is meant some rule of thumb that usually reduces the work required to obtain a solution to a problem. [Jackson 1974, p. 95]

> [Guzman's scene analysis program uses] a set of informal reasoning rules (sometimes called *heuristics*) which were derived by an empirical, experimental method. ... Although the resulting programs might not be explainable in terms of some deep underlying theory, they perform adequately in most situations and therefore in a very practical sense they solve the problem. [Raphael 1976, p. 237, 238]

> A heuristic is any stratagem for improving the performance of an artificial intelligence program. The heuristic programming approach to artificial intelligence is perhaps the most popular and productive one today. It contrasts with another major approach, ... [the] simulation of human thought. In this approach the aim is more to understand and use the features of human intelligence than to apply any technique which works. [Sampson 1976, p. 128]

> A heuristic is a method that directs thinking along the paths most likely to lead to the goal, less promising avenues being left unexplored. [Boden 1977, p. 347]

> An important distinction underlying much of the work in AI is that between two types of methods used to solve problems. One method is called algorithmic, the other, heuristic. Algorithms are commonly defined as procedures that guarantee a solution to a given kind of problem; heuristics are sets of empirical rules or strategies that operate, in effect, like a rule of thumb. [Solso 1979, p. 436]

> Heuristics, as every Aler knows, are rules of thumb and bits of knowledge, useful (though not guaranteed) for making various selections and evaluations. [Newell 1980, p. 16]

> *Heuristics* are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices. [Pearl 1984, p. 3]

But there are some novel interpretations emerging, which appear to be "second generation ideas" on what heuristics really are. Unfortunately, these are never very clearly defined and explained. For example, Hofstadter (1979) has a view of heuristic as "compressed experience":

> Of course, rules for the formulation of chess plans will necessarily involve heuristics which are, in some sense, "flattened" versions of looking ahead. That is, the equivalent of many games' experience of looking ahead is "squeezed" into another form which ostensibly doesn't involve looking ahead. In some sense this is a game of words. But if the "flattened" knowledge gives answers more efficiently than the actual look-ahead—even if it occasionally misleads—then something has been gained. [p. 604]

Another example of a cursorily presented novel interpretation comes from Albus (1981):

> Procedures for deciding which search strategies and which evaluation functions to apply in which situations are called heuristics. Heuristics are essentially a set of rules that reside one hierarchical level above the move selection and evaluation functions of the search procedure. A heuristic is a strategy for selecting rules, i.e., a higher level rule for selecting lower level rules. [p. 284; see also pp. 222, 223]

And finally, Lenat (1982) appears to have a view of heuristics similar to Hofstadter's:

> Heuristics are compiled hindsight: they are nuggets of wisdom which, if only we'd had them sooner, would have led us to our present state much faster. This means that some of the blind alleys we pursued would have been avoided, and some of the powerful discoveries would have been made sooner. [p. 223]

It is our belief that definitions like these last three are not sufficiently popular in the general AI community to warrant being included as part of a comprehensive definition of 'heuristic'. This situation may of course change with time.

So far we have just reviewed some of the assorted definitions of heuristic that have appeared over the past 40 years. We have seen that different researchers have emphasized different properties as being relevant to whether a heuristic is being employed, and we have seen a shift in emphasis in the concept. We would like now to distinguish more carefully the different (but interrelated) "dimensions of meaning" that the concept embodies. We can distinguish four dimensions along which various researchers have judged whether a process is a heuristic: uncertainty of outcome, basis in incomplete knowledge, improvement of performance, and guidance of decision making.

The role of uncertainty: heuristics vs. algorithms

We have seen how, in many of the definitions, 'heuristic' has been opposed to terms like 'algorithmic', 'guaranteed', and 'complete'. We will argue in this section that the central idea underpinning these definitions is that heuristics exist in a context of *subjective uncertainty as to the success of their application*. We will explain in what respects those, like Minsky, who think that heuristics are perfectly compatible with algorithms are correct, even though there is a genuine conflict between these two notions. In illustration, we shall give the sense in which even the brute force British Museum Algorithm is a heuristic.

Central to this key property of uncertainty and, as we saw, present at the very earliest adoptions of the concept of heuristic by AI is the notion of *algorithm*. 'Algorithm' has many meanings, although it is doubtful that the ambiguity has caused any of the disagreements over definition. If we define algorithm as merely "a set of [formally defined and uniquely interpreted] rules which tell us, moment to moment, precisely how to behave" (Minsky 1968, p. 106), then any procedure for making decisions is algorithmic, and hence all heuristics implemented on computer, or otherwise strictly formulated, are algorithmic. We use "procedure-algorithm" to mean this type of algorithm. However, when 'heuristic' has been considered opposed to 'algorithm', 'algorithm' has always had a much stronger sense which includes an element of guarantee about finding a solution. Korfhage (1976, p. 48), following normal usage, characterizes an algorithm as follows:

1. Application of the algorithm to a particular input set or problem description results in a finite sequence of actions.
2. The sequence of actions has a unique initial action.
3. Each action in the sequence has a unique successor.
4. The sequence terminates with either a solution to the problem, or a statement that the problem is insoluble.

If the last restriction is too strong we may define 'semi-algorithm' as follows: "a method that will halt in a finite number of steps if the problem posed has a solution, but will not necessarily halt if there is no solution." For some problems there is always a solution, e.g., adding two integers, and so this distinction does not apply. We call such algorithms "simple-algorithms."

'Heuristic' has often been opposed to some such notion of algorithm, although not universally by all authors. Newell *et al.* (Feigenbaum and Feldman 1963, p. 114) opposed it to semi-algorithms, while Tonge (Feigenbaum and Feldman 1963, p. 172) and Slagle (1963, p. 194) opposed it to simple-algorithms. Feigenbaum and Feldman (1963, p. 6) implied a contrast with simple-algorithms and also seemed to say that there is no issue here. We have seen that Minsky, Raphael, and Sampson denied any opposition with algorithms and that both Nilsson (1980, p. 72) and Jackson (1974, p. 95) denied that heuristics need sacrifice a guarantee of finding a solution (although neither said anything about starting out with a guarantee, or what happens if one should find that the putative heuristic *does* guarantee finding a solution). Boden (1977, pp. 347, 348) argued on the one hand that there is no opposition with simple-algorithm or with semi-algorithm, but on the other hand there is a contrast insofar as heuristic programs postpone decision making, whereas algorithms require all decisions be precisely specified beforehand.

Given this mix of conflicting claims one could simply do as Barr and Feigenbaum (1981, pp. 28, 29) do, namely, state that *heuristic* is an ambiguous term and that to keep things clear one will be using such and such a definition. This response is inadequate, however, because it ignores several reasons for believing that there *is* one correct definition. These are the single origin of the term in the AI literature, i.e., the work of Polya; the fact that AI authors have placed so much theoretical weight on this specific term; and the fact that AI authors have not given their definitions with the air of "for convenience I use the term ..." but with the impression that they have captured what is really important and common to this branch of research, namely, the use of rules that save effort, or provide satisfactory solutions, or lack a guarantee, or what have you. Therefore, we believe a proper analysis of heuristic must result in one definition, and this one definition must show adequate appreciation for all the ideas which have been linked to it, and it must be able to explain any incompatibility among such ideas.

How is this possible with all the conflicting opinions over this contrast with algorithms? To answer this we must get to the core of the idea of algorithm and see precisely where the conflicting opinions are focused.

An algorithm presumes a problem and a precise step-by-step procedure that solves the problem or shows it unsolvable. Therefore, if we have a problem and we have an algorithm for that problem, then, so to speak, we should have no problem. So why weren't Newell *et al.* satisfied with the British Museum Algorithm? Because, obviously, the *real* problem wasn't solved by it, namely, to provide a solution *within certain resource limits*. Such resources include time, space, and processor type, and, on the user's part, the effort to use and to remember how to use this algorithm. So the *real* problem was much more complicated than just that of providing proofs—it was, to provide a proof within the resource limits. And for that problem the British Museum Algorithm did not provide a solution, i.e., was not an algorithm.

Once we see that the *real* problems are the "practical" problems, we can see how to resolve the algorithm versus heuristic conflict: heuristic and algorithms are not normally opposed because they usually apply to different classes of problems. Heuristics apply to the *real* or practical problem, whereas algorithms apply to the abstract, theoretical "any solution will do" problem.

Heuristics were never meant to be distinguished from algorithms except in those cases where (a) the algorithm provides a poor solution to the practical problem and (b) the algorithm claims to guarantee solving the practical problem. A strategy like the set-of-support as employed by resolution theorem provers is therefore algorithmic for the abstract, theoretical problem of proving theorems, but, because it is better than the British Museum algorithm for solving the *real* problem of proving theorems in reasonable time, and because it does not claim infallibility at doing this, it can also qualify as a heuristic. Likewise, the A^* algorithm is heuristic because it is not known to be *practically* optimal for finding optimal paths in a state space, even though it will in time find an optimum path. Therefore Minsky and others were right all along in saying that practical algorithms can be heuristic.

There is, or course, a reason why authors naturally chose algorithms to contrast with heuristics. This is because the claim to guarantee a solution is based on the element of confident and assured decision making which is antithetical to the notion of heuristic. Slagle was right when, in his definition of *heuristic*, he emphasized the property of not knowing whether the next action is the best thing to do now. If we *did* know this—know that our program was going to take the best possible action at each step—then clearly our program would not be a heuristic one. Heuristics are, among other things, rules that offer tradeoffs: a small cost (often this is the omission of a guarantee) in the hopes of a bigger payoff. But if at each step we *know* our program is performing optimally, then tradeoffs and expressions of hope would be out of the question, so it could not be a heuristic program. Of course this does not mean that an optimal strategy cannot be a heuristic for us, only that if we *know* it to be optimal it cannot be a heuristic. If such a heuristic were later discovered to be optimal, then the only excuse for our continuing to call it heuristic would be one based on habit, since by then its entire character would have changed for us.

Our discussion has laid out a clear boundary between optimal and nonoptimal strategies as regards the use of the term *heuristic*. We believe this to be sufficient to establish a definitional property. We therefore claim that heuristics are incompatible with knowledge of optimal decision making and that this is an essential property of heuristics.

Note that this property amounts to saying that heuristics are never thought to guarantee a solution to the practical problem. However, uncertainty as to optimality is a better criterion to use than uncertainty as to solution guarantee, because the meaning of 'optimality' includes the element of practicality. Optimality forces us to assume the practical context, whereas solution guarantee risks the confusion of problem solution and practical solution. It is our claim that this confusion is responsible for the differing opinions on whether an algorithm can be a heuristic. Heuristics are only opposed to those algorithms which guarantee providing a *practical* solution to a problem; they are not opposed to algorithms which merely guarantee a solution with no guarantee that this solution is practically realizable.

All in all we can conclude that this property of heuristics, the uncertainty as to optimality, allows us to place much of the heuristics literature in perspective. We can now appreciate the tendency to oppose heuristics with algorithms. Algorithms are often associated with confident, certain decision making. If all one wants is a solution to an abstract problem, then there is no uncertainty about getting one with an algorithm. In this respect the set-of-support is a thoughtless, mechanical, nonheuristic strategy. But for a real, practical problem, the certainty might be absent and then even an algorithm can be a heuristic.

Uncertainty can also partially explain other ideas often opposed to heuristics such as "exhaustive search" and "complete analysis." If we are thorough and complete then we are certain of an answer. The epitome of thoroughness is the British Museum style algorithm which systematically but indiscriminately searches everywhere for a solution. This is one reason the British Museum type algorithms are so often contrasted with uncertain, unthorough, incomplete heuristic methods. Nonetheless, British Museum algorithms can be heuristics for real, practical problems. The lack of intelligence in these algorithms means they can often search through more possibilities in a given period of time than a heuristic method, since applying discrimination requires effort. It is conceivable that in problem domains where solutions are not sparsely distributed, the British Museum algorithm could perform quite well (see, for example, Siklóssy *et al.* 1973). Hence the British Museum algorithm can plausibly be called a successful strategy which we nonetheless do not believe to be optimal. Therefore it too can qualify as a heuristic.

Heuristics as based on incomplete knowledge

At the other extreme from confident decision making lies blind, random, and ignorant decision making. Heuristics, however, offer selectivity, guidance, plausible solutions, intelligent guesses, etc., all of which indicate at least a partial insight into the problem situation. From the a-priori knowledge that a rule is based on an understanding of some facet of the problem one can derive some confidence; hence one will give some credit, some plausibility, to this rule. However, actual performance will eventually affect this sense of plausibility, and if performance is poor the partial insight itself will be brought into question.

Partial insight is what makes heuristics of such interest to the cognitive science side of AI. If one has some information about a problem domain's structure but not enough to provide an efficient algorithm for solving all such problems, then this information can still be put to use in the form of heuristics to

improve problem-solving performance. Since so many real world problems are of this form, it is no wonder heuristics have become so popular and are so worth studying.

Lenat (1982, p. 222) has remarked similarly on the domain of heuristic applicability:

> At an earlier stage [of knowing a domain], there may have been too little known to express very many heuristics; much later, the environment may be well enough understood to be algorithmized; in between, heuristic search is a useful paradigm. Predicting eclipses has passed into this final stage of algorithmization; medical diagnosis is in the middle stage where heuristics are useful; building programs to search for new representations of knowledge is still pre-heuristic.

Thus we have a spectrum of confidence levels in decision making. At one extreme are efficient algorithms and other decision processes which we believe are optimal (whether or not they guarantee a solution), and at the other extreme we have the most inefficient algorithms and other unprofitable processes in which we place little confidence. *Heuristics fall in between: they are plausible without being certain.* The placement of a particular process along this spectrum is, however, relative to our perception of the extremes. For example, Newell *et al.* (Feigenbaum and Feldman 1963, p. 116) originally spoke of their British Museum algorithm as producing such "simple and cheap" expressions that it could not be heuristic, whereas they (Newell and Simon 1972, pp. 120, 121) later call it heuristic because its generator is only apparently "blind-trial-and-error", since by generating only theorems it is so much more selective than one that generates *all* well-formed formulas.

As a defining ingredient in heuristics, partial insight offers more than just confidence. Insight is the core of a heuristic's intelligence, its reason for being. A particular heuristic is represented by its particular insight; without a genuine grasp of some aspect of the problem a device must perforce contribute nothing to problem solving. It could only masquerade as a heuristic until its luck wore out. It is with this dimension of meaning of *heuristic* in mind that Polya, Gelernter, Slagle, and Jackson offered their definitions. Here are two simple examples of the sense in which heuristics might represent partial insight into a problem domain.

In symbolic logic we know that $\neg\neg A$ is equivalent to A. This is a piece of knowledge about how logic formulas relate to one another. We also know that theorem provers bog down with more and more complex formulas, and that a big part of theorem proving is matching for similar patterns in other formulas. We can employ all these insights to construct a heuristic that simplifies pattern matching: "under such and such circumstances eliminate excess negations." Other heuristics could make use of other equivalences (e.g., those expressed by DeMorgan's rules) to recommend the conversion of all formulas to some type of normal form.

In chess, the piece of knowledge that at one point in play one's bishop can in two moves go to more possible squares than one's rook, might allow one to generate the temporary heuristic "use the bishop on this turn."

As can be seen from these simple examples, the possibilities for generating heuristics are endless. One discovers something about the problem and constructs a device to make use of this insight. The rule will thereby be plausible, and if one does not know enough about the problem to tell if the device is optimal then it can also be a heuristic.

When we analyze insight we see that it comes in a variety of forms. There is a simple insight that can be expressed in simple terms. The example from symbolic logic, that $\neg\neg A$ is equivalent to A, is a simple insight since we can describe it simply. Then there are insights that are not easily expressible, but are nonetheless present. For example, Samuel's (Feigenbaum and Feldman 1963, pp. 71–105) checker-playing program employed a polynomial evaluation function that included features like "center control," "mobility," "number of forceable exchanges," etc. This 16-element polynomial represents an insight into checkers, but how would one express it simply? For one thing the insight is highly dependent on Samuel's particular program and his test samples. One might argue that hence it is really only an insight into how to play good checkers with this particular program. We think, however, that the insight is more universal; it tells us, among other things, that as a general rule kings in the center are more powerful than we might have expected.

These two examples also show us that some insights are known prior to their heuristics while others are discovered by examining heuristics. Hence these two forms of knowledge, the aspects of the problem (factual knowledge) and how to make use of these aspects (procedural knowledge), can exist quite independently.

Some AI researchers have reflected on the abstract nature of heuristic insight. Boden (1977, p. 351), Minsky (Feigenbaum and Feldman 1963, p. 409f), and Newell *et al.* (Feigenbaum and Feldman 1963, p. 122) speak of moving from the start state to the goal state and avoiding many fruitless paths by *sensing* whether one is getting warmer or colder. A kind of negative feedback keeps one on the right track. At each point where alternatives are presented a decision is made. Only some of these decisions need to be fruitful to keep one from going too far astray. Evaluation functions fit this description well.

Another set of reflections comes from Boden (1977, pp. 341–344), Minsky (1968, p. 425ff), Pearl (1984, pp. 113–118), and Polya (1945, pp. 37–46, 180), who speak of the power of *analogies* and *models*. Analogies may be as complicated as or even more complicated than the original problem. If sufficient parallelism between the two cases exists then they allow us to transfer both the insights and the heuristics based on these, rather than be forced to rediscover these same insights and heuristics. Models are a form of analogy. They are simplified representations which allow us to focus on, or make more salient, some of the more relevant aspects of a problem. They offer a more compact representation of a problem's essentials and are thus a form of partial insight. Their simplicity may also make it easier to discover new insights. And those discovered are likely to concern the more essential aspects of the problem.

Heuristics as performance improvers

Heuristics are often seen as improving performance, as some of our definitions above have illustrated. But how do they do it? In this section we will show what may seem obvious, but should not be, that heuristics are used to help improve the performance of a problem-solving system. In this regard they are like tools introduced to fix or enhance a system. The notion of performance improvement under consideration is that of increased efficiency, that is, receiving more benefit out for effort put in. We believe, but shall not thoroughly discuss here, that the various permutations of decreasing effort and increasing benefit explain many of the forms in which heuristics occur.

First of all, it should be clear that we would not be using heuristics in problem solving, in discovering solutions, guiding

search, etc., if we did not believe that they were useful—that they contributed something. This is so patent as to be almost not worth mentioning. However, it is not the same thing to say that a device is useful and that it improves performance. An automobile's steering wheel is useful but it does not improve performance—it is a standard fixture, already present in the very idea of a standard automobile. An electronic ignition, on the other hand, being an option that is superior to the standard electromechanical ignition, can be said to improve performance. There appear to be two distinct opinions in AI as to whether to be a heuristic a device must improve performance, or whether it need merely be useful. Minsky and Sampson explicitly included performance improvement in their definitions. In fact, for them, this is the only significant property of heuristics. Along with Minsky and Sampson are all those that express performance improvement in the form of effort reduction or search reduction. These include Barr and Feigenbaum, Chang and Lee, Feigenbaum and Feldman, Hofstadter, Hunt, Jackson, Nilsson, Raphael, Slagle, Tonge, and Winston. In the other camp we find heuristics introduced not to improve the system, but rather, there in their own right from the very start. For this group, heuristics can be standard mechanisms, not just newly introduced superior features. This camp includes Albus, Boden, Newell *et al.*, Pearl, Polya, and Solso.

It is interesting to note that the second group contains those researchers whose main interest is in the "cognitive science" aspect of AI—the use of a computer to simulate human psychology, whereas those in the first group are researchers whose main concern is in the production of programs to perform certain (traditionally human) tasks. This observation suggests that what underlies the different usages of these two camps is some sort of different emphasis: the first camp toward a practical, task-oriented kind of problem solving by computer, the second camp toward a more global man–machine theoretical kind of problem solving. This is also suggested by the fact that even someone like Minsky, who makes performance improvement the primary feature of his definition, uses 'heuristic' independent of performance improvement when discussing human heuristics (Minsky 1968, p. 27).

In sum, it appears the majority of members of the AI community employ 'heuristic' to refer to some device applied as an *addition* to some problem-solving system in expectation of performance improvement. Therefore performance improvement is a property included in the most popular usage of 'heuristic'. Nonetheless, we must acknowledge a legitimate tradition of using 'heuristic' to stand for a preexisting internal mechanism of some problem-solving system, prior to any additions being made. We personally prefer this latter usage.

In reading descriptions of programs which are said by their authors to employ heuristics, one is struck by their use of certain terms in characterizing the properties possessed by these putative heuristics. These properties all presuppose the element of adding something to a system that was not present before, and they are all commonly attributed to heuristics. These properties are reflected in the use of the following adjectives when describing heuristics: *practical* (as opposed to theoretical), *domain-specific*, *ad hoc*, and *empirical* and in the use of the following nouns in place of 'heuristic': *trick*, *patch*, and *tool*.

We believe that this usage of *heuristics* is due to the experimental research framework in which AI takes place.

Quite commonly in AI a researcher devises an elegant theory of how some class of problems is solved. When tested in the form of a computer program it turns out that this theory has failings. It cannot handle some problem formulations, takes too long on others, etc. To overcome these difficulties the author begins to bend, patch, and otherwise modify the theory so that its performance improves. In fact, this occurs so commonly in AI that a special term seems appropriate to stand for these ad hoc, empirically introduced improvements for practicality's sake. For better or worse 'heuristic' has been drafted for the role. One can see some justification for this. Heuristics lack the formal certainty and confidence given to a theoretically derived decision mechanism. Heuristics make use of partial information and small insights to help guide one to a solution. Their prime justification is the practicality they afford, not the elegance or adequacy of the theory underlying them.

Thus where Polya and Newell *et al.* would have used 'heuristic' to refer to general methods that are initially part of the problem-solver's outfitting, such as means-ends-analysis, try-and-test, analogous reasoning, and inductive reasoning, others in AI introduce heuristics as afterthoughts when a particular problem-solving theory has practical failings, and where yet it remains desirable to save the good parts of the theory. Resolution theorem proving provides some examples. The bare bones resolution strategy is elegant and shows plenty of promise since it uses only one inference principle, and so need not possess any complex logic for deciding which rule of inference to apply next. However, bare bones resolution turns out to be hopelessly inefficient for most theorems. So rather than reject it entirely we seek ways to salvage it. For example, we try ordering the clause selection by using evaluation functions, we try choosing simple clauses first, or we try to use the negated conclusion and its ancestors (set-of-support). These strategies are plausible, fallible, and, it turns out, very useful in extending the theorem-proving power of the pure theory. This is why 'heuristic' in AI has tended to acquire a sense akin to *practical* and opposed to *theoretical*. This practicality is the ground for speaking of a heuristic strategy's improvement over the theoretical strategy's performance. "Domain specific" is derivative from this opposition to theory. Part of the meaning of saying that a heuristic is domain specific is that it responds to the peculiarities of the problem. And usually we only bother with peculiarities if we want to actually solve practical problems. Theoretical strategies tend to apply more generally over several problem domains.

Another research framework within AI in which a type of heuristic-based performance improvement occurs can be illustrated as follows. A skeletal program schema is written to handle heuristics for some problem domain. Heuristics are then tossed in whenever the researcher sees fit, as he acquires experience with the problem, his program, and the behavior of its heuristics. Therefore heuristics in AI are often called *ad hoc* and *empirical*, and this is not viewed negatively, but rather positively, as part of their general property of being performance improvers. Numerous AI systems adopt this same sort of skeletal framework for attaching heuristics. Virtually all of the so-called "expert systems" are designed to facilitate this experimental additive performance improvement. They are built so that human expertise can be readily transferred to them, and often the expertise is in the form of heuristics. For example, Douglas Lenat's mathematical concept discovery program, AM, at one point had some 250 heuristics coded as production rules. Examples of such rules are

> If f is an interesting relation, then look at its inverse. [Lenat and Harris 1978, p. 30]

> If concept G is now very interesting, and G was created as a generalization of some earlier concept C, give extra consideration to generalizing G, and to generalizing C in other ways. [Lenat and Harris 1978, p. 43]

He designed his system to facilitate the addition of new rules and he hoped to add more in time (cf. Lenat 1982, 1983*a*,*b*). Again each new rule is seen as potentially improving the discovery abilities of the program. Lenat also experimented with AM; he appears to have added rules in a try-and-test fashion as various ideas for enhancing AM's performance occurred to him (Lenat 1982, pp. 205–207).

We have just seen how performance improvement is a popular activity in AI and how heuristics are associated with this activity by patching impractical theories or by being incrementally added to a general problem-solving schema. These are ways of expressing the basic benefit-greater-than-cost intuition and show that a number of properties that are ascribed to heuristics can be derived from it. For instance, Gelernter's (Feigenbaum and Feldman 1963, p. 137) idea of heuristics as "sufficiently nonporous *filters*" and the popular notion of "selective *pruning* of decision/game trees" both focus on our desire to eliminate from consideration more useless items than valuable ones. We also have Tonge's (1960) "shortcuts," "simplifications," and "adequate solutions." These are attempts to keep the costs (of having to perform detailed analyses) down but the benefits (quality of solutions) sufficiently high. Abstractions or generalizations of decision devices, insofar as they reduce the number of detailed devices that need to be memorized and also reduce the need to consider each one every time a decision is required (but do not grossly mishandle too many of the exceptional cases), are also candidates for being heuristic. Or, more generally, any area where we can trade off resource utilization for a slight loss of number of solvable problems, or of quality of solutions, is an area open to performance improvement by heuristic methods. Conversely, if by whatever means we can marginally increase resource utilization (time, memory, tool, etc.) costs, but recoup a dramatic increase in solvable problems, or a significant increase in quality of some solutions, then this too can qualify as heuristic performance improvement. Expert systems are good sources for finding such effort-increasing heuristics since we typically add rules to them, which implies occupying more space and spending more time considering extra rules. A consequence of the existence of this last class of heuristics is that all those definitions of heuristic that use the phrases "effort reduction" or "search reduction" are misleading. "Performance improvement" is the more accurate phrase since it covers all the cases of relative cost–benefit improvement.

Heuristics as decision guiders

Heuristics have been variously presented in the form of proverbs, maxims, hints, suggestions, advice, principles, rules of thumb, criteria, production rules, programs, procedures, methods, strategies, simplifications, option "filters," goal transformers, and no doubt there are others. (See the History section, given earlier, for an example of each.) What is common to all these forms? In this last section on properties we hope to show that heuristics always try to help the problem solver by *guiding his decisions* during the course of moving from initial to solution state. Since this is not really a contentious point with anyone, we will not belabour it. Nonetheless, because it is a key property it deserves a clear statement. In the end we will discover a few new things about decision guidance; in particular we hope to clear up the issue of whether heuristics can be passive options presented to an executive decision maker or whether they must be the higher-order decision rules guiding the search for a solution.

To show that decision guiding is the primary function of heuristics, we first show that the element of choice is always present when heuristics are discussed, and that heuristics as a group do not consistently influence any other element of a problem solver or his situation. For example, they are not devices that consistently influence memory, clarity of vision, creativity, thoroughness, or any other feature of problem solving. To phrase it differently, we claim that the use of 'heuristic' always presumes the existence of a decision mechanism and that the heuristic's effect is to lead this mechanism down one path as opposed to another. The influence may be direct, i.e., the heuristic actually decides where to go. For example, evaluation functions are direct. Or the influence may be indirect, i.e., the heuristic simply changes some aspect of the problem situation. For example, "eliminate complex theorems from the subproblem list." There is no sharp line dividing these two types of influence.

By way of illustration, we bring forth a representative sample of usages and definitions to support the claim about the universality of decision guidance. We can start with Polya, whose usage we recall was rather different from what is prevalent in AI today. For Polya, any behavioral method considered useful while problem solving could be a heuristic method. This includes asking oneself certain key questions, drawing a diagram, or trying to rephrase a problem. Since Polya did not use the paradigm of search when describing mathematical problem solving, these behavioral methods need not affect any decision making. They could influence some unconscious processes which suddenly inspire the solver to see a solution. Nevertheless, Polya only speaks of his methods as being chosen by a solver. The student should try this, think of that, ask himself this question, etc.

In the early AI period, the paradigm of heuristic use is one of guiding search through a problem space. This applies to every author covered above. Like Polya, Newell *et al.* officially leave open the possibility of heuristics being arbitrary useful processes applied during problem solving. Yet in fact they solely use them to influence the order of development of the solution path along the subproblem tree. The value of the heuristics is explained by their effect on movement through the subproblem tree, and all their heuristics are clearly decision guiding. The four primary methods in the Logic Theorist directly choose some of the paths to be followed, while the "similarity test" acts as a filter, screening some theorems prior to matching and hence indirectly guiding the course of search.

Gelernter employed a similar tree search paradigm and uses his heuristics to filter out less promising decision options. Tonge used heuristics to simplify wherever possible the entire pattern of activity used to balance assembly lines. Slagle uses the Logic Theorist framework where heuristics both decide what problem transformations to apply next, as well as transform the problems themselves. Minsky introduces heuristics in a context of search where they guide the solver gradually to a solution. (He gives "hill climbing" as a typical example.) Feigenbaum and Feldman mention state-space search reduction in their definition and give assorted rules of thumb as examples. For these the solver is portrayed as trying one thing rather than another and is thereby led down a different problem-solving path.

Following the early definitional era the state-space search-guiding paradigm remained the dominant framework for talking about heuristics; we see it in virtually all game-playing applications. In these, the heuristics decide on which of the assortment of legal moves to perform next. Likewise for theorem-proving applications. Which formula in the expanding list of formulas should the system examine next, resolve next; which of a set of simplification rules should it try next; etc.?

When we come to expert systems the search paradigm is mentioned less often but is no less strong. Typically such a system works in conjunction with a human expert. It may ask him for more input, ask for certain tests to be performed, or explain why it favors a certain hypothesis. Its heuristics can thus be viewed as guiding the problem-solving decisions made by itself and its users as they focus in on a satisfactory diagnosis (MYCIN), molecular structure (DENDRAL), or geological analysis (PROSPECTOR).

Along with this list of usages we can bring forth all the key words used in defining 'heuristic' as evidence that heuristics exist to influence problem-solvers' choices. Proverbs, maxims, hints, suggestions, and advice are clearly meant to influence decision making. "Principles," "criteria," "rules of thumb," and "rules" properly all exist to govern conduct and in the case of problem solving, one's conduct is typically consciously selected. Furthermore, having chosen to follow a rule, one's subsequent decisions are often altered by the new face the problem now presents. *Programs, procedures, methods*, and *strategies* are all organized sets of rules which, however complex, are in effect single rules themselves. Each is but a rule which summarizes a variety of conduct for assorted circumstances which may arise over a period of time. Hence they too exist to govern conduct, and the problem solver decides to follow them or not. Finally, the other things which some heuristics have variously been called, "*filters*," "*simplifiers*," "*transformers*," and such, seem always to have as their purpose the restructuring of the problem situation so that one has a different set of options from which to choose.

From cited key-word definitions and from usage, it is clear that decision guidance has always been seen as the basic function of a heuristic device. But there have been some confusions regarding this property, so we will now set about exposing and resolving these. We first will describe how, with respect to decision making, there are two distinct ways *heuristic* occurs in the literature. It is because of their failure to recognize this fact that some authors have given erroneous definitions of *heuristic*.

With regard to the executive's function, which determines the overall direction of activity, a heuristic may be used *actively* to decide which of several rules, pieces of advice, game moves, or solutions to select, or it may be referred to *passively*, as one of the rules, pieces of advice, etc. which is being offered for selection. These two categories are not mutually exclusive, nor need a heuristic belong to at least one category. For example, in the case of the Logic Theorist the four "methods" are passive heuristics selected by the nonheuristic executive, but also they are active heuristics when they decide which of the theorems to consider next. An example of a heuristic that is in neither category can also be found in the Logic Theorist. The "similarity test" is not part of the executive since all it does is change the problem environment, not decide the course of problem solving; and on the other hand neither is it selected from among alternative activities to perform since it is always applied and it has no competitors.

It is hard to say whether the one category of usage is more common than the other. Many heuristics do not make executive decisions, such as "castle early" or "try rephrasing the problem." But on the other hand many heuristics are not chosen from a list of possible things to do at this stage of problem solving. They are constantly working features of the system—filters are a good example here. Again, many other heuristics do actively direct the search—all game-playing and theorem-proving programs that employ heuristic evaluation functions do this. Likewise, many other heuristics occur with competitors—most expert systems or production systems have long tables of heuristics which the executive must scan to decide which to currently employ. Therefore, all in all, we must conclude that both these usages are genuine and that neither dominates.

Having distinguished these two ways that heuristics can be involved in a decision situation, we have completed the groundwork for discussing nebulous problems like the hierarchical organization of problem-solving systems, the layers of decision making, the locus of intelligence—in executive or subordinate, or perhaps the difference between high-order strategies and low-order tactical decision making, etc. All of these could be analyzed in a context of some heuristic, some perfect, and some random decision devices. However, we can do none of this here. All we would like to do with this insight regarding executive and subordinate heuristics is square away some problematic statements made by Slagle and by Albus.

As we saw earlier, Slagle's official definition requires that all heuristics be active. However, we have just seen that it is definitely not true that all heuristics are active, i.e., part of the executive; they do not all decide what should be done next.

Along a similar vein is Albus's claim that "A heuristic is a strategy for selecting rules, i.e., a higher level rule for selecting lower level rules" (Albus 1981, p. 284). So again heuristics are portrayed as part of the executive, i.e., in control of what is done next. Elsewhere he makes similar remarks:

> In most cases, the search space is much too large to permit exhaustive search of all possible plans, or even any substantial fraction of them. The set of rules for deciding which hypotheses to evaluate, and in which order, are called *heuristics* ... [Heuristics have a] recursive nature. A heuristic is a procedure for finding a procedure. [1981, p. 222]

These remarks suggest the source of his belief that heuristics must be in the executive: he believes that heuristics only occur in a context that fits the state-space search paradigm. Elsewhere he actually describes all problem solving as state-space search (Albus 1981, pp. 281–285). When one has a formal state-space network defined, it is easy to imagine that all decisions can be reduced to answering "what path shall I follow?" or "what strategy shall I follow for moving down a path?" Heuristics become the strategies, and the strategies for selecting the strategies, which tell us where to go next.

But it is implausible that all problems can be made to fit the state-space scheme. As Boden (1977, p. 350) observes, it is hard to define solution states and intermediate states for problems like "shall I marry him?" and "how can I write a detective story?" And even within this scheme, heuristics can be applied to numerous background duties as opposed to making direct choices of what option to choose next. As an example we mentioned above the similarity test of the Logic Theorist. Another example is Lenat's (Lenat and Harris 1978, pp. 30–33) heuristics in AM, many of which contribute incrementally to

prioritizing projects on the "agenda" of things to do next, without individually choosing what exactly is done next. Indeed, Lenat's executive is very simple and runs without any heuristics at all. Likewise, all declarative (as opposed to procedural) expressions of heuristic knowledge about a domain, such as MYCIN's "if evidence *E* then assert *A* with confidence factor CF," are heuristics that do not choose what to do next. In the case of MYCIN, a modified depth-first algorithm makes these choices (Barr and Feigenbaum 1982, pp. 187–191).

Conclusion

We set out to define *heuristic* against a historical backdrop of conflicting definitions. What emerged from our survey of definitions was that *heuristic* could refer to any device used in problem solving, be it a program, a data structure, a proverb, a strategy, or a piece of knowledge. But not just any such device. There had to be an element of "rule of thumbishness" about the device; it had to be useful but need not guarantee success. This lack of guarantee, however, applies to the entire, *real* practical picture of supplying a solution. A heuristic device *can* guarantee supplying a solution, but if it is also provably the optimal device for arriving at a solution, then it is not a heuristic. As for its utility, this is derived from the heuristic's having captured some fact, some insight, about the problem domain. All in all, therefore, heuristics fit on a spectrum of devices between those that are random and uninspired and those that are applied automatically because they never fail to please, or if they do fail then we resign ourselves to this because we have a proof that there can be no better device.

Although these two properties should be sufficient to eliminate the majority of nonheuristic devices, most AIers use *heuristic* more restrictively still. They reserve the term for just those devices they have added to their experimental system in hopes of improving its performance. Although we suspect they would relinquish this property upon a little reflection, this restricted usage is nonetheless prevalent. For instance, it is to be found in the majority of definitions given by AIers themselves. Therefore we must admit this property if we are to give *an AIer's definition* of 'heuristic'. As for what "performance improvement" means, we found that, contrary to many authors, it did not mean search or effort reduction, that this was only half of the equation, the other half being the possibility of improvement in solution quality in exchange for a modest increase in search effort.

With the addition of performance improvement we have all the properties needed to restrict the set of problem-solving devices to those that AIers call heuristic. Technically this would suffice as a definition. Yet when we examine all the remarks made about heuristics in the literature we find that there is a popular theme not covered by fallibility, plausibility, and performance improvement, namely, the function of heuristics in problem solving. They work by guiding search, suggesting behavior, making decisions, or transforming the problem so that different courses of action are open. These properties are reflected in the choice of words used to make heuristics concrete: rules, advice, procedures, filters, etc. We suggested that the search guidance characterization is so popular because of the popularity in AI of the state-space framework for describing problems. Of course, the state-space framework is so popular in AI because, as scientists, AIers can benefit by analyzing a problem's essentials into paths, option nodes, states, etc. Heuristics in this framework naturally affect the decisions as to which paths to follow.

Having described all this we concluded the discussion of decision guidance by establishing that heuristics could be involved in direct active decision making, or merely passively as options to execute, and that therefore some authors were incorrect in thinking that all heuristics chose what course problem solving would follow next.

Concisely put, *a heuristic in AI is any device, be it a program, rule, piece of knowledge, etc., which one is not entirely confident will be useful in providing a* ***practical*** *solution, but which one has reason to believe will be useful, and which is added to a problem-solving system in expectation that on average the performance will improve.*

ALBUS, J. S. 1981. Brains, behavior, and robotics. Byte Publications Inc., Peterborough, NH.

BARR, A., AND FEIGENBAUM, E. A. (*Editors*). 1981. The handbook of artificial intelligence. Vol. 1. William Kaufmann, Inc., Los Altos, CA.

——— 1982. The handbook of artificial intelligence. Vol. 2. William Kaufmann, Inc., Los Altos, CA.

BLEDSOE, W. W. 1971. Splitting and reduction heuristics in automatic theorem proving. Artificial Intelligence, **2**, pp. 55–77.

BODEN, M. A. 1977. Artificial intelligence and natural man. Basic Books, Inc., New York.

CHANG, C.-L., and LEE, R. C. 1973. Symbolic logic and mechanical theorem proving. Academic Press, Inc., New York.

COHEN, P. R., and FEIGENBAUM, E. A. (*Editors*). 1982. The handbook of artificial intelligence. Vol. 3. William Kaufmann, Inc., Los Altos, CA.

ERNST, G. W., and NEWELL, A. 1969. GPS: A case study in generality and problem solving. Academic Press, New York.

FEIGENBAUM, E. A., and FELDMAN, J. (*Editors*). 1963. Computers and thought. McGraw-Hill Inc., New York.

FINDLER, N. V. 1976. Heuristics. *In* Encyclopedia of computer science. *Edited by* A. Ralston. Van Nostrand Reinhold & Co., New York, pp. 606, 607.

GELERNTER, H. 1959. Realization of a geometry-theorem proving machine. *In* Proceedings of an International Conference on Information Processing, Unesco House, Paris, pp. 273–282. (Reprinted in Feigenbaum and Feldman (1963), pp. 134–152.)

GRONER, R., GRONER, M., and BISCHOOF, W. F. (*Editors*). 1983. Methods of heuristics. Lawrence Erlbaum, Hillsdale, NJ.

HOFSTADTER, D. R. 1979. Gödel, Escher, Bach. Basic Books, New York.

HUNT, E. B. 1975. Artificial intelligence. Academic Press, Inc., New York.

JACKSON, P. C., JR. 1974. Introduction to artificial intelligence. Petrocelli Books, New York.

KORFHAGE, R. R. 1976. Algorithm. *In* Encyclopedia of computer science. *Edited by* A. Ralston. Van Nostrand Reinhold & Co., New York, pp. 47–50.

LENAT, D. 1977. The ubiquity of discovery. Artificial Intelligence, **9**, pp. 257–287.

——— 1979. On automated scientific theory formation: A case study using the AM program. *In* Machine intelligence. Vol. 9. *Edited by* J. E. Hayes, Donald Michie, and L. I. Mikulich. Ellis Horwood Ltd., West Sussex, England, pp. 251–283.

——— 1982. The nature of heuristics. Artificial Intelligence, **19**, pp. 189–249.

——— 1983*a*. Theory formation by heuristic search. The nature of heuristics II: Background and examples. Artificial Intelligence, **21**, pp. 31–59.

——— 1983*b*. EURISKO: A program that learns new heuristics and domain concepts. The nature of heuristics III: Program design and results. Artificial Intelligence, **21**, pp. 61–98.

LENAT, D., and HARRIS, G. 1978. Designing a rule system that searches for scientific discoveries. *In* Pattern directed inference

systems. *Edited by* D. A. Waterman and Frederick Hayes-Roth. Academic Press, New York, pp. 25–51.

MINSKY, M. L. 1961*a*. Steps toward artificial intelligence. Proceedings of the Institute of Radio Engineers, **9**, pp. 8–30. (Reprinted in Feigenbaum and Feldman (1963), pp. 406–450.)

——— 1961*b*. A selected descriptor-indexed bibliography to the literature on artificial intelligence. IRE Transactions on Human Factors in Electronics, **2**, pp. 39–55. (Reprinted in Feigenbaum and Feldman (1963), pp. 453–475).

——— 1965. Matter, mind, and models. *In* Proceedings of the International Federation of Information Processing Congress 1965, Vol. 1, pp. 45–49. (Reprinted in Minsky (1968), pp. 425–432.)

——— 1967. Computation: Finite and infinite machines. Prentice Hall Inc., New York.

——— (*Editor*). 1968. Semantic information processing. M.I.T. Press, Cambridge, MA.

NEWELL, A. 1980. The heuristic of George Polya and its relation to artificial intelligence. A paper given at The International Symposium on the Methods of Heuristic, University of Bern, Switzerland, Sept. 15–18, 1980. (Published in Groner *et al.* (1983), pp. 195–244.)

NEWELL, A., and SIMON, H. A. 1972. Human problem solving. Prentice Hall, Englewood Cliffs, NJ.

NEWELL, A., SHAW, J. C., and SIMON, H. A. 1957. Empirical explorations with the logic theory machine. *In* Proceedings of the Western Joint Computer Conference, Vol. 15, pp. 218–239. (Reprinted in Feigenbaum and Feldman (1963), pp. 109–133.)

NILSSON, N. J. 1971. Problem-solving methods in artificial intelligence. McGraw-Hill, Inc., New York.

——— 1979. A production system for automatic deduction. *In* Machine intelligence. Vol. 9. *Edited by* J. E. Hayes, Donald Michie, and L. I. Mikulich. Ellis Horwood Ltd., West Sussex, England, pp. 101–126.

——— 1980. Principles of artificial intelligence. Tioga Publishing Co., Palo Alto, CA.

PEARL, J. 1984. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Publ. Co., London.

POLYA, G. 1945. How to solve it. Page references to 1957, rev. 2nd ed. Doubleday Anchor, New York.

RALSTON, A. (*Editor*). 1976. Encyclopedia of computer science. Van Nostrand Reinhold & Co., New York.

RAPHAEL, B. 1976. The thinking computer. W. H. Freeman & Co., San Francisco, CA.

ROMANYCIA, M. H. J. 1983. The concept of heuristic as used in the artificial intelligence community. M.A. Thesis (Philosophy), University of Alberta, Edmonton, Alta.

SAMPSON, J. R. 1976. Adaptive information processing. Springer-Verlag Inc., New York.

SAMUEL, A. L. 1959. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, **3**, pp. 221–229. (Reprinted in Feigenbaum and Feldman (1963), pp. 71–105.)

SIKLÓSSY, L., RICH, A., and MARINOV, V. 1973. Breadth-first search: Some surprising results. Artificial Intelligence, **4**, pp. 1–27.

SLAGLE, J. R. 1963. A heuristic program that solves symbolic integration problems in freshman calculus. *In* Computers and thought. *Edited by* E. A. Feigenbaum and J. Feldman. McGraw-Hill Inc., New York, pp. 191–203.

——— 1971. Artificial intelligence: The heuristic programming approach. McGraw-Hill, Inc., New York.

SOLSO, R. L. 1979. Cognitive psychology. Harcourt Brace Jovanovich, Inc., New York.

TONGE, F. M. 1960. Summary of a heuristic line balancing procedure. Management Science, **7**, pp. 21–42. (Reprinted in Feigenbaum and Feldman (1963), pp. 168–190.)

WINOGRAD, T. 1973. A procedural model of language understanding. *In* Computer models of thought and language. *Edited by* Roger C. Schank and Kenneth Mark Colby. W. H. Freeman and Company, San Francisco, CA, pp. 152–186.

WINSTON, H. P. 1977. Artificial intelligence. Addison-Wesley Publishing Co. Inc., Philippines.