1981

The heuristic of George Polya and its relation to artificial intelligence

Allen Newell
Carnegie Mellon University

CMU-CS-81-133

THE HEURISTIC OF GEORGE POLYA AND ITS RELATION TO ARTIFICIAL INTELLIGENCE

Allen Newell

July 1981

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Given at the International Symposium on the Methods of Heuristic,
University of Bern, Bern Switzerland
15-18 Sep 1980

To appear in R. Groner, M. Groner and W. F. Bischoof (Eds.),
Methods of Heuristics, Hillsdale, NJ: Lawrence Erlbaum (forthcoming)

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551.

Abstract

Polya's fundamental work in heuristic is well known and well regarded in artificial intelligence. However, no one has built seriously on his work, eg, by constructing programs that make use of his heuristic. This paper attempts to understand why this might be the case. First, an attempt is made to characterize the nature of Polya's heuristic. Then six theses are put forward that might account for the failure of his work to have a major impact. Three are easily discarded, but three are serious candidates: that the essential heuristic knowledge is not captured in Polya's work; that the emphasis on learning in Polya's heuristic is beyond the current art in artificial intelligence; and that the use of auxiliary problems is beyond the current art. This last thesis is explored in detail in the remainder of the paper. Some interesting concepts emerge, particularly the notion of *object-centered* problem space and the contrast between *tame* and *wild* subproblems.

Table of Contents

List of Figures

The Heuristic of George Polya and its Relation to Artificial Intelligence[1]

1. Introduction

1.1. Polya Revered and Polya Ignored

This paper is an inquiry into the relationship of George Polya's work on heuristic to the field of artificial intelligence (hereafter, AI). A neat phrasing of its theme would be *Polya revered and Polya ignored.* Polya revered, because he is recognized in AI as the person who put heuristic back on the map of intellectual concerns. But Polya ignored, because no one in AI has seriously built upon his work. In the coin of the AI realm, no one has built an AI system to realize the schemes investigated in Polya's works.

Everyone in AI, at least that part within hailing distance of problem solving and general reasoning, knows about Polya. They take his ideas as provocative and wise: "And everyone should know the work of Polya on how to solve problems" (Minsky, 1961). But they also see his work as being too informal to build upon: "Analogical reasoning is potentially a very powerful heuristic device. In fact, Polya (1954) devoted one entire volume of his two volume work to the discussion of the use of analogy and induction in mathematics. Unfortunately, he presents ad hoc examples but no general rules" (Hunt, 1975 p221).

Since it is clear to all (no doubt to Polya most of all) that his work on heuristic was not fashioned to aid in creating machines to solve problems, the issue could be laid to rest as no issue at all. The occasional explicit references to Polya's work are exactly suitable -- nods to the precursors of the more detailed work that AI needs to do. All is as it should be.

It does not seem so to me. Not that anything is *wrong*, but that there is more to be considered than what lies on the surface.

1.2. Why Take Up this Task?

On the surface, there is good reason to take up this task. The topic of this conference is heuristic. Polya has written more about heuristic per se than anyone else -- six volumes, each short, but full to overflowing with concrete analysis of heuristic. Furthermore, this conference is not limited at all to AI, but is meant to range wherever heuristic will take it.

---

[1] I thank Jon Bentley, who shares an abiding interest in Polya's work (Bentley, 1976), for many discussions on the topic of this paper. I also thank Jaime Carbonell and John McDermott for comments on an earlier draft.

However, there is another, more personal, reason: I am, in an odd way, a student of Polya's. Perhaps a bit of early history will be permitted.

I entered Stanford as a freshman in 1945 for one quarter, before a short World War II stint in the Navy, returning there in 1946 to do my undergraduate work in physics. I had some mathematical leanings, and had taken some calculus when in high school, a novelty in those days. In that first returning quarter I took an odd course entitled *Mathematical methods in physical science* -- odd, in not being the standard freshman fare of physics, chemistry, western civilization and French. I can recall no reason. Perhaps, it was the usual course catalog shopping that students always do, perhaps the oft noted returning veteran's independence (though I was still only 19).

The course turned out to be Polya's *How to Solve It* course. The little book of that title must have been just out, although I have no recollection of it in the course. We did go through all its pieces and parts. I remember diagramming in detail the progress through a problem -- remarkably like the sequence on the inside cover of volume II of *Mathematical Discovery*. I do not remember becoming a better problem solver, I'm afraid -- though who am I to say? I certainly graduated and went on to become a professor. Among academics, at least, that should not diminish the credibility of the hypothesis that I learned something. But I did become fascinated with Polya and his teaching. It was a neat course.

I have incontrovertible evidence of my fascination. I took every course of Polya's open to an undergraduate. I think I took them all, though no doubt I am mistaken. In any event, I took his courses on *Theory of Probability, Interpolation and Numerical Integration* and *Differential Geometry*, all before the end of my freshman year. Why, I ask you, would a young freshman physics major take a course in differential geometry, or interpolation, unless he was fascinated by the teacher? Even then it seemed a bit arcane. I was clearly majoring in Polya. On the other hand, I had no personal relationship with Polya at all. I was just an undergraduate and the culture of our great universities makes it quite normal to pass through them with only the course as the channel between student and professor.

When I look over my own early papers, many explicitly devoted to heuristics and problem solving, I find they are totally silent about the work of Polya.[2] Thus, the question posed about the field is a question posed of myself. I certainly knew Polya's work. I had assimilated it at a stage when learning is as easy and natural as getting wet in the rain. I acquired all his books, as soon as I came across them, long before starting research in what was to be AI. Polya surely influenced me in all my early work. No explicit evidence of that exists, so far as I know, but it is surely true.

---

[2]The first reference is a brief footnote in (Newell, Shaw & Simon, 1962, written 1958), but due to Simon, my co-author.

1.3. The Nature of the Paper

This paper is not to be an inquiry into my own personal scientific development. The focus is to be on the substance of Polya's work and its relationship with the enterprise of AI. The first task is to characterize the body of heuristic that Polya has built up. Interestingly, no such characterization exists in the literature, as far as I know. It can be taken as just one more indication that Polya is revered but ignored -- he is to be *appreciated* but not *analyzed.* Then we will consider several alternatives that might explain why the impact of this work on AI has not been greater. One particular thesis will be examined in detail, though all the others will not thereby be disposed of.

Before launching into the main enterprise, it is important to be clear that Polya's work in heuristic is aimed explicitly at teaching the young to be better problem solvers and teaching their teachers how to help them. Polya never aspired to have any influence or relevance to work in AI. From certain remarks in his book, it seems he had real doubts about the feasibility of mechanizing reasoning. He never refers to any of the work in AI, even though his last book, *Mathematical Discovery II*, was published in 1965, several years after the early work in heuristic search (starting in the mid fifties), had become quite well known. Indeed, he has indicated recently that he is unacquainted with AI, except for its name.[3] Be that as it may, both Polya's practical aim and overarching view are different from those of the present investigation. A neat phrasing of this difference might be *wisdom* versus *science.* Polya is concerned with giving wisdom to problem solvers. We are here concerned with a science of intelligence and with the construction of computer programs that exhibit intelligence.

[3] In a letter commenting on an earlier draft of this paper.

2. Polya's Heuristic

Polya's heuristic is contained in a corpus of six small volumes, published between 1945 and 1965. A scattering of papers also contains material on heuristic, but almost all of these appear to have been incorporated into the books. There may be some publications after 1965, but I do not know of them.

For those unacquainted with him, Polya is an eminent mathematician, whose work is primarily in classical analysis and combinatorial analysis. Born in Budapest well before the turn of the century (1887), his productive career has an immense span. Throughout this time he has been a professor, first at the Swiss Federal Institute of Technology in Zurich, from 1914 to 1940, and since then at Stanford University (where he has been emeritus since 1953).

By the time he turned his attention strongly to the issue of heuristic in 1945 (with the publication of *How to Solve It*), he was already in the later part of his career; he was 78 when the last book was published. Thus, his explicit work on heuristic was entered into as a reflection on a long and active career, both in research and in teaching. His interest in the issues of how mathematics is done -- how discoveries are made and how problems are solved -- had been with him throughout his career. Many of the attitudes in the book are prefigured in the analysis text he produced in 1924 jointly with Gabor Szego (Polya & Szego, 1972), which consisted entirely of problems, answers and interesting remarks.

The first book, *How to Solve It* (Polya, 1945), has the form of a how-to manual. This little volume is a jewel, compared to the standard books of the how-to genre (see DeBono, 1968, for one of the best of the latter). It is graceful, straightforward, and pleasantly open. Furthermore, it deals with real problems, usually mathematical ones. Though often quite elementary, in response both to the requirements of pedagogy and the focus on teaching mathematics at the high school level, these problems range widely in the mathematics involved and include many problems of classical mathematical interest.

Most noteworthy about *How to Solve It* is the formulation of a set of heuristics for problem-solving, cast in the form of brief questions *(What is the unknown?)* and commands *(Draw a figure!)*, within a frame of four problem solving stages: *Understand the problem*, *Obtain a plan*, *Carry out the plan*, and *Look back at the solution*. These heuristics are given flesh by being worked out in many examples, often in hypothetical dialogues of the problem solving and teaching process.

A second feature of the book is its *Dictionary of Heuristic*, a collection of very brief articles about all aspects of heuristic, from expanded notes on the basic heuristic questions, to historical comments on famous early contributors to heuristic, such as Descartes, Liebnitz and Pappus. This style of very brief, isolated essays on points of interest is carried through the other volumes on heuristic, though as part of *Examples and comments*

at the ends of chapters, rather than as separate sections.

The third feature of the book is the emphasis on the *art of solving problems.* The main substance is communicated by problems worked and discussed. Though the text ranges over the full breadth of issues about problem solving, their packaging as micro-essays keeps them in their place, clearly subordinate to the examples. No systematic theory or scholarly assessment is attempted; it is often explicitly disavowed.

Nine years later, in 1954, Polya published *Mathematics and Plausible Reasoning* in two volumes, *Induction and Analogy in Mathematics* and *Patterns of Plausible Inference* (Polya, 1954). This work may be seen simply as an extension of *How to Solve It* -- more problems, more useful commentary, more examples worked though with an analysis of how to solve them.

Alternatively, this work may be seen as a treatise on how to assess the plausibility or credibility of inductions and inferences. To this end, all the examples of the first volume are case studies of induction and analogy. They provide a base for asking in the second volume whether there is any way to develop a *logic of induction* or a *calculus of credibility* for hypotheses. What, in particular, is the role of the theory of probability?

Like the work on plausible reasoning, the third book, *Mathematical Discovery*, was published in two volumes, one in 1962, the other in 1965 (Polya, 1962, Polya, 1965). Three themes commingle here. The first is the one common to all the prior work. *Mathematical Discovery* contains many additional problems, all solved in ways that lay the analysis bare, all with various micro-essays of illuminating commentary.

The second theme is the analog of the investigation into the logic of plausible inference. Polya attempts to lay out, somewhat generally, a theory of problem solving. Again, the style is similar. The first part, *Patterns*, lays out a series of cases to form the base for the second part, denoted, *Towards a General Method.*[4] Consistent with his style, the treatment of general methods is not systematic or theoretical, but discursive and continuously built around problems.

The third theme is giving advice to high school teachers of mathematics. Though always clearly present in all the earlier volumes, the topic is developed quite explicitly in Volume II.

During the same period when *Mathematical Discovery* was being written, Polya was devoting substantial time to working with teachers of mathematics. Several of the chapters in *Mathematical Discovery* were taken

[4]The parts do not match the volumes: Volume 1 holds the four chapters of Part I plus two chapters of Part II; Volume 2 holds the remaining nine chapters of Part II. This undoubtedly explains why the volumes are not separately titled.

from lectures to such groups. He also gave a series of lectures during a summer program that focused on elementary mathematical methods in physics and astronomy. These lectures were transcribed and published as *Mathematical Methods in Science* in 1963 (Polya, 1977). They differ from the other works in not being about heuristic at all, but in providing treatments of classical problems from a viewpoint embued with his philosophy of heuristic. They are Polya put into practice.

In sum, the entire corpus can be taken simply as a large pool of worked problems with commentary on heuristic. Modulating this are some deliberate investigations, namely, whether there can be a calculus of credibility and what is the nature of general method. But these serve, not so much as the central analyses to be established by the instances, but as themes which channel considerations in a local and pleasing way, producing a coherent selection of example problems. Polya presents a consistent point of view, whose exposition and message is evenly distributed throughout the work.

My own summary of Polya's heuristic will divide into six parts: (1) the nature of the human problem solver; (2) problem solving methods; (3) a theme of working with related problems; (4) a theme of the current problem as preparation for future problems; (5) the calculus of credibilities; and (6) other heuristics. This division hardly seems one that Polya would use. It reflects instead a view of problem solving current in AI.

2.1. Model of Human Problem Solving

AI views a problem solver as an information processing system, which acquires a problem from a task environment by encoding it in an internal data structure, and which solves the problem by processing these structures by (a sequence of) internally available methods, making use of bodies of encoded knowledge (Newell & Simon, 1972, Nilsson, 1980, Winston, 1977). The basic processing system (the architecture) is thus separated from the methods and knowledge. The same methods and knowledge can be used by many different types of processing systems. AI declares thereby for a general theory of problem solving, applicable to humans, computers, animals or Martians. Differences exist between different types of problem solvers, of course. But these differences arise from specific features in the architecture -- different tradeoffs in speed, memory and accessing, which induce different styles of problem solving, different choices of methods, etc.

In part, Polya's heuristic rests on the specific nature of the human processing system, in contradistinction to the general methods of solving problems, which are presumably common to all intelligent agents. We will focus on the three characteristics that loom largest in Polya's treatment.

Attention must be focused. Problem solving happens when the human *attends* to various aspects of a problem. A large amount of advice in Polya is devoted to coaching the reader (as someone learning how to problem solve) to attend to this or that aspect. This is seen as problematical. Attention in humans does not

happen automatically, but occurs only because the problem solver picks up this and that part of the problem, successively attending to the parts and then to their relations.

It is rather easy to see how focusing attention is a necessary condition for a problem solver. Whatever else might be involved in attending, processing some part of a problem implies attending to that part. But there is also a flavor of sufficiency about Polya's version of the proposition. If a problem solver attends to all the right things, then the appropriate processing will automatically occur. To paraphrase an old adage (one that Polya doesn't happen to use): "Take care of attention and the problem solving will take care of itself." There is no notion in Polya that attention is actually sufficient, only that one gets something more by attending than just getting the inputs assembled correctly for some planned operations.

Memory must be tickled. Memory is indirectly accessed. Explicit throughout Polya is that contact must be made with what the problem solver already knows. Implicit is that this cannot be done directly, but occurs only because attending to X suggests Y. Thus, many things are justified, not because they lead to the solution (indeed, they may be failures), but because they lead to tickling something in memory that will help.

> "Many of these questions and suggestions [in Polya's main set] aim directly at the *mobilization* of our formerly acquired knowledge: *Have you seen it before? Or have you seen the same problem in a slightly different form? Do you know a related problem? Do you know a theorem that could be useful? Look at the unknown! And try to think of a familiar problem having the same or a similar unknown.*" (Polya, 1945 p146)[5]

Problem solvers must be motivated. Polya has a theory of motivation as part of his model of specifically human problem solving.

> "The intelligent problem-solver tries first of all to understand the problem as fully and as clearly as he can. Yet understanding alone is not enough; he must concentrate upon the problem, he must desire earnestly to obtain its solution. If he cannot summon up real desire for solving the problem he would do better to leave it alone. The open secret of real success is to throw your whole personality into your problem." (Polya, 1945 p180)

> "Teaching to solve problems is education of the will. Solving problems which are not too easy for him, the student learns to persevere through unsuccess, to appreciate small advances, to wait for the essential idea, to concentrate with all his might when it appears." (Polya, 1945 p89)

Motivation for Polya also has a social component. In common with many others concerned with teaching mathematics and science, he views people, especially young people, as naturally curious and interested in both knowing and exercising competence. They are especially interested in what they can discover for themselves. Yet, our schools tend to make learning dull and gradually stifle the natural curiosity of the young. Thus an important role of the teacher of problem solving, and by extension these works on heuristic, is to awaken and

---

[5] Here and throughout, italics in quotations occur in the original.

kindle the interest of the student.

The psychological plausibility of Polya's model. These propositions of Polya's agree with all that is known in modern cognitive psychology. The role of attention has been emphasized throughout psychology. Its surplus action (ie, beyond its being just a way to refer to the operands of cognitive processes) has been emphasized more for learning than for performance, namely, that learning takes place automatically on whatever is attended to.

Modern cognitive psychology has generally accepted the proposition that memory is content addressed -- that what is recalled next is a function of what is being attended to now. This is explicit in production system models of cognition (Anderson, 1976, Newell, 1973) and implicit in generalizations such as the *Encoding Specificity Principle* (Tulving & Thomson, 1973), which states that retrieval is on the basis of cues, and these must match the cues stored at learning time.

Interestingly, one proposition that does not play a strong role in Polya's heuristic is that humans have a tendency to *block*, ie, to get into cognitive ruts, and that they must do special things to open themselves up to alternatives that break out of the restrictive sets. Such a proposition is taken as central in the popular literature of how to think and solve problems, as indicated for instance in the title of Adam's (1974) *Conceptual Blockbusting* (see also DeBono, 1968).

2.2. Problem Solving Methods

Much of Polya's heuristic is taken up with specific problem solving methods. These are proposals for generalized steps to be taken to solve a problem. As far as the problem solver's structure is concerned, they presume only that it attempts problems by setting goals and is capable of following procedures. Some of the methods are the stock in trade of work in AI, some of them are not. Consistent with Polya's goal of educating, the methods are usually given by examples, with their generality indicated by comments, but it is still rather easy to extract many of the methods themselves.

Consider first the main method that occurs in Polya, namely, the one that graces the inside covers of *How to Solve It*. It is outlined in Figure 2-1. It is used to organize all the major questions that constitute the core of Polya's heuristic: *Understand the problem*; then *devise a plan*; then *carry out the plan*; and finally *examine the solution*. Explicitly laid out and used as an organizing principle throughout *How to Solve It*, these

questions or their paraphrases also run through the entire corpus.

Understand the problem

What is the unknown? The data? The condition?
Is the condition satisfiable? Sufficient for unknown? Insufficient? Redundant? Contradictory?
Draw a figure. Introduce suitable notation.
Separate the parts of the condition. Write them down.

Devise a plan

Seen the problem before? In a different form?
Know a related problem? A theorem that could be useful?
Know a familiar problem with the same unknown? With a similar unknown?
Given a related solved problem, use it. Its result? Its method? Could an auxiliary element help?
Restate the problem. Restate it still differently.
Go back to definitions.
Solve first some related problem.
    More accessible? General? Special? Analogous?
    Solve a part of the problem? Keep part of condition?
    Use the data somehow. What other data could determine the unknown?
    Change the unknown? The data? Both? Make them closer to each other.
Use all the data? The whole condition? All essential notions?

Carry out the plan

Check each step.
See it clearly? Prove it?

Examine the solution

Check the result? The argument?
Derive the result differently? See it at a glance?
Use the result elsewhere? The method elsewhere?

Figure 2-1: Polya's main method (adapted from Polya, 1945).

We can use this method to make an important point. No matter how general this statement seems to the reader, it is indeed a *method.* It has all the earmarks of what has come in AI to be called a method (Newell, 1969):

1. *It is a specific way to proceed.* There are lots of ways of working on problems (even successful ways) that do not follow these steps. (Eg, many methods do not involve planning; and modern programming methodology promulgates proving as you go, to avoid the need for verification at the end.)

2. *It is a rational way to proceed.* Following the steps provides chances of actually solving the problem. The chances are much better than just following any old sequence of actions. Polya, of

course, argues that they are better than that, ie, better than many other methods which are also rational.

3. *It involves subgoals and subplans.* Methods of any scope attain their generality by means of subgoals, ie, positing obtaining things that satisfy general specifications, without stating how they are to be obtained. In Polya's method the main intermediate object (the plan) is itself a method, thus making the method itself seem exceedingly unspecific. But a plan is a symbolic object, just like any other -- something that can be discovered, criticized, revised, etc. -- so this is perfectly legitimate. The only question (an empirical one) is whether, given the capabilities of the problem solver to solve the subgoals, the method organizes behavior sufficiently to increase appreciably the chances of success on the main problem.

4. *Its occurrence is observable.* From an experimental point of view, it is easy to determine whether this general method is being carried out rather than some other (or perhaps no method at all, but behavior in the service of other goals). Exploration of the problem must occur prior to formulating a plan. The plan exists prior to its implementation. Both these events can be detected (see Newell & Simon, 1972 Ch10, for strikingly clear empirical examples).

Many methods exist in Polya in addition to the main method. Only rarely does he actually describe them in clear step by step terms, preferring to illustrate them in the context of specific examples. They vary in interesting ways, as a sample of them shows:

1. *Reductio ad absurdum.* A classical method of proof, going back to the Greeks. It has three steps:

    1. Assume the negation of what is to be shown.
    2. Derive a contradiction (a falsehood).
    3. Conclude that the original must be true.

2. *Method of loci.* A method for doing geometric constructions with compass and straightedge, also classical and going back to the Greeks. Polya's statement of it runs as follows (Polya, 1962 p5):

    1. First, reduce the problem to the construction of ONE point.
    2. Then, split the condition into TWO parts so that each part yields a locus for the unknown point; each locus must be either a straight line or a circle.

3. *Go back to definitions.* Though appearing to be little more than a slogan to direct attention, this is in fact a method carried out by a straightforward symbolic procedure of expanding the immediate working representation of the problem to incorporate the definitions of the terms.

4. *Sequence of convertible transformations.* This also is a straightforward symbolic procedure of iteratively applying operations to an expression to yield a succession of new ones. The specification of *convertible* operations serves to limit the use of this method to situations where transformations exist that do not lose information, so that if a solution to a transformed problem is found, it can be converted back to a solution to the original.

5. *Working backwards.* The method is actually the formation of the inverse problem space, in which the states of the space remain the same, but the operators are inverted, the original desired state becomes the new initial state, and the original initial state becomes the new desired state. Thus, the method can be stated as:

1. Form the inverse problem space.
2. Solve the inverse problem in this space.
3. Use the solution path to generate the solution in the original space.

Although it is not always possible to form this inverse space or to pose the inverse problem, it can be a powerful device.

6. *Accumulate knowledge.* This method is implicit in many of Polya's bits of advice about understanding the problem:

    1. Examine a part of the problem to add new knowledge about the problem.
    2. Retry the whole problem from its expanded description.

    Again, however simple-minded this may seem, it is a genuine method. The special knowledge it carries is the advice to reconsider the whole given the new information at hand, even though (for instance) the new constraint or relation did not of itself seem to solve the problem.

7. *Test by dimension.* This method applies to the goal of testing a result. It is an adaptation of the general technique of dimensional analysis in physics, based on the fact that if $A = B$, then they must be in the same units. If A is an unknown and B is a formula for computing it, then the units of A are known, and those of B may be determined from the components of the formula.

8. *Interpret the formula.* This method applies to the goal of deriving a result in a different way from that originally discovered. Given that the result is expressed in a formula, the subexpressions and operations in the formula can be interpreted in the model or diagram of the problem as components and relations. This will suggest a way of deriving the result (often different from the original).

9. *Switch the unknown.* This method applies to the goal of deriving a new problem after a problem has been solved, something that Polya encourages. It is a simple procedure whereby the unknown and one of the givens simply switch roles: Unknown → Given, Given → Unknown.

As can be seen from examining this list, some of the methods (1 and 2) are classical, well known enough to have their own names. Others (3 and 4) are not only familiar, but are straightforward symbolic procedures -- substituting and applying known operators. These methods represent the sorts of things routinely carried out by computer programs that do symbolic mathematics, such as MACSYMA. If they were not so obvious in classical times (and Polya's quotations from the Greek, Pappus, show much care lavished on describing a sequence of convertible transformations), it is because the notion of a *calculus* has become familiar to us beyond notice.

The next two (5 and 6) are typical of methods currently in routine use in AI programs. These are both typical abstract behavior schemes, which make use of subgoals and submethods to actually attain some definite behavior. The Accumulate-knowledge method is actually somewhat more general than its AI correspondents. For instance, the Transform Method in GPS (Newell & Simon, 1972) says: create a subproblem of reducing a difference; if succeed, add a new expression; then retry the original goal, now

working from the new expression. This is narrower in the more specific character of the subproblem, and the more limited form of the resulting new knowledge, which is always an expression.

The last three methods (7, 8 and 9) show that not all methods deal with Polya's top goals of solving a general problem or proving a general theorem. Subgoals exist in the main method, eg, to test if a result is correct; to derive the same result in a different way; and to use the result to obtain a new problem, different from the original. Each of these subgoals have their own specialized methods, though they are still highly general methods in themselves.

As the range of this sample shows, the methods in Polya include both famous methods and obscure ones; and methods of great and of little power. They also include the full range of methods with respect to their use in current AI programs: some that have already been used; some that have not, but probably easily could be; and some (eg, the main method) that pose challenges of undetermined magnitude.

2.3. Working with a Diversity of Related Problems.

A striking feature of Polya's heuristic is its pervasive use of related problems as a path to the solution. Just from the basic questions in Figure 2-1 under **Devise a plan** there is a litany of suggestions whose point is to generate *auxiliary* problems (as Polya calls them). There is a wide diversity of these from many different sources. This diversity of auxiliary problems of a given problem is not to be confused with the wide range of example problems that Polya uses to illustrate his ideas. To indicate the emphasis on auxiliary problems in Polya, 50% of the questions in the basic set are devoted to auxiliary problems in one way or another. In addition, 60% of the worked examples in *How to Solve It* have at least one auxiliary problem, sometimes several.

As the questions above indicate, one reason why auxiliary problems are so prevalent is that they play several distinct roles, their total frequency of occurrence thus being fed from several sources. For instance, they can be used for their results, for their method, or to suggest further useful approaches, methods or problems. Polya gives examples of all types and on a few occasions he is able to use an auxiliary problem in more than one way on the same problem, producing a powerful impression of elegance.

As an example of an auxiliary problem, consider finding the center of gravity (CG) of a homogeneous tetrahedron (Polya, 1945 p38). This is a problem in solid (three dimensional) geometry. An analogous problem can be set up in plane (two dimensional) geometry, namely, to find the center of gravity of a homogeneous triangle. This latter is the auxiliary problem. It is easier to solve and the hope is that it will be useful in solving the larger problem. In fact, Polya is able to use its method as a plan for the three-dimensional problem. As another example, consider the problem (the top of Figure 2-2) of finding the closest distance of

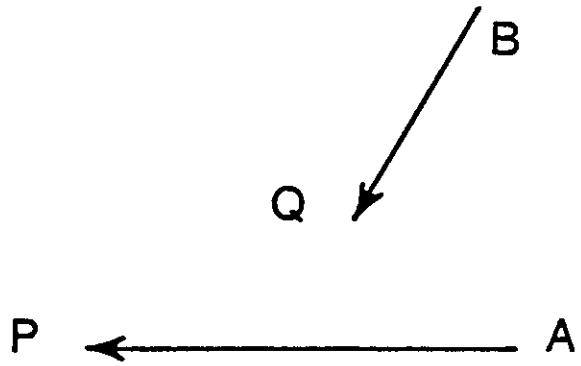
approach of two ships, given their positions and (constant) velocities at a given initial moment (Polya, 1945 p166). Here the auxiliary problem is the special case (shown in the lower part of the figure) in which one of the ships, say *B*, is at rest. Again, the auxiliary problem is much easier to solve and Polya finds a way of using its result in solving the main problem. Even these two examples make apparent the diversity of auxiliary problems.

Problem:

Given: 1. The speed and positions of two ships at a given moment.

2. Each ship steers a linear course with constant speed.

Find: The distance of the two ships when nearest to each other.



Auxiliary Problem:

Find: The distance in the special case that one ship is at rest.

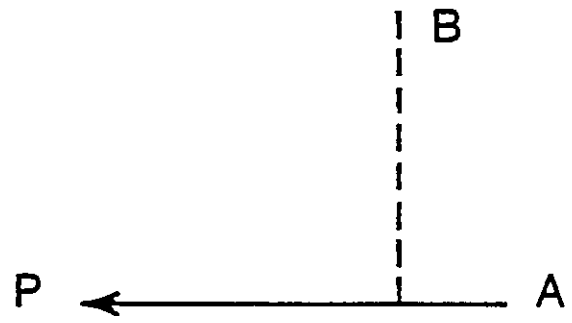


Figure 2-2: Two ships problem (from Polya, 1945).

2.4. Preparing for Future Problem Solving

Another striking feature of Polya's heuristic is its emphasis on preparing for the future -- on learning. The final step of every problem is to *Examine the solution* (Figure 2-1). Three pairs of questions are given in the figure. The first pair, on checking, certainly relates to *verification*, a final step in solving the main problem. But the last two pairs extend beyond that to using the present problem to provide things for future problems -- methods, results, etc. Polya leaves no doubt about this. For instance, he quotes Descartes with approval (Polya, 1965 p2):

> " 'Each problem that I solved became a rule
> which served afterwards to solve other problems.'
> Descartes: *Oeuvres, vol VI,; Discourse de la Methode.*"

Given the other elements in Polya's heuristic, this emphasis is understandable. It goes beyond the usual defense of the importance of subject matter in education. For Polya, problems are often solved by analogy from the solutions of prior specific problems. Thus, a large stock of specific problems, their solutions, and the specific method of their solution is a necessary feature of a good problem solver.

One effect of this emphasis in Polya's work, if taken seriously, would be to make no problem important for itself, but important only for what it contributes to future problems. This emphasis might be taken as arising from the special educational context in which Polya's books are written -- where all problems are toy problems, made up only for pedagogical purposes and of no intrinsic interest. But that is not so. I daresay Polya would argue strongly that the chances of solving real problems depends every bit as much on what has been learned from prior problems, both real and educational. Further, he would aver that the acquisition of useful and helpful problems depends on the problem solver always attending to lessons as he goes along.

This total emphasis on future-orientation could be entirely justified. The sketch in Figure 2-3 shows one way to view this. The figure represents the space of problems. The capability of the problem solver is shown by the inner area. Any problem within the area can be solved essentially by recognition -- it will seem a familiar problem. The true dynamic problem solving ability amounts to only a small rind on the outside of this surface.. This is how far search will reach beyond what is already known. This is indeed governed by various methods, but there are a small variety of these in any event. Once they are mastered, the rind comes to have a typical absolute thickness representing the maximum that can be extended in a given problem solving session. Each problem solving attempt, if successful, grows a small bulge on the boundary, that is, the rind becomes part of the interior. Reworking a solution adds some modest additional amount to the bulge. Integrated over the life of the problem solver, the area plus the perimeter becomes the important determiner of total problem solving power. Thus, the perimeter should be maximized, as well as the area. Whenever two perimeter segments are close to each other, the rinds (ie, the projections from each) overlap, thus adding less

than two rind's worth to the total coverage. Thus, it pays to be diverse.

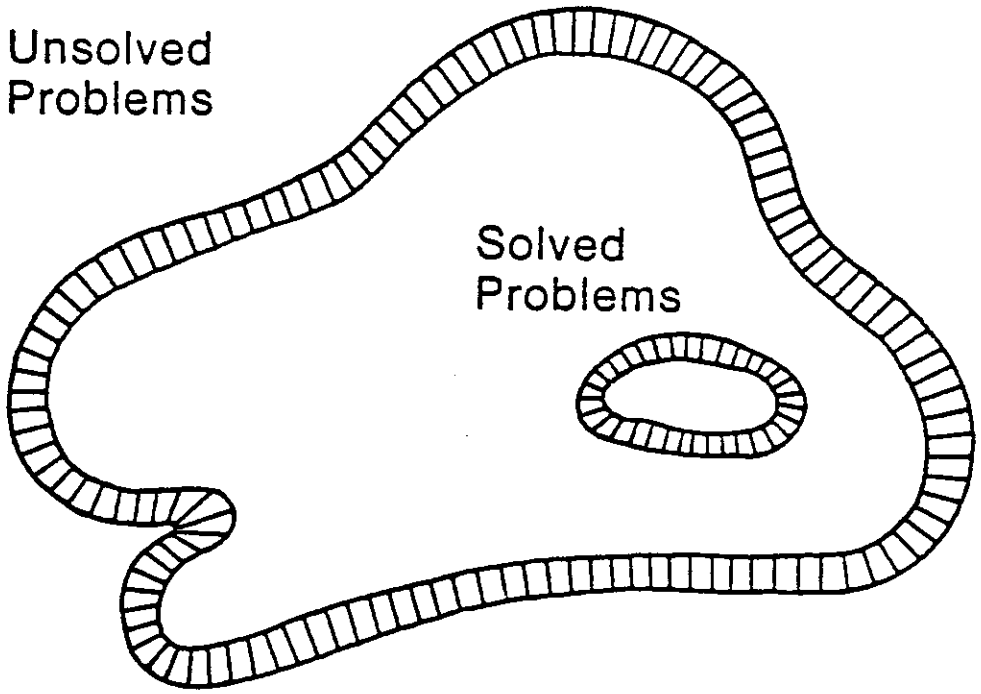


Figure 2-3: A world where only future oriented problem solving makes sense.

2.5. Credibility of Inductions and Analogies

Polya treats in detail the credibility of analogies and inductions. It shows up most explicitly in *Patterns of Plausible Inference*, but is present throughout. The reason for this emphasis is clear. The process of problem solving continuously generates new problems -- transformations of the original problem as well as auxiliary problems. These are problems precisely because the problem solver does not know how to solve them. Therefore the persistent question is whether a just posed theorem or conjecture is true or a just posed problem can be solved.

But the concern runs deeper. For there is the sharp contrast for Polya between the results of mathematics, which delivers results of certainty, and plausible methods, which do not. The gulf between these two is repeatedly emphasized throughout the entire corpus.

"We secure our mathematical knowledge by *demonstrative reasoning*, but we support our

conjectures by *plausible reasoning.*" (Polya, 1954 vol. I p v)

The upshot of his position is easy enough to state in bald form (though Polya himself always shies away from categorical statement). In a nutshell, Polya's answer is that a *qualitative* notion of credibility of a conjecture or an induction is possible and useful, but that it cannot be extended to a *quantitative* calculus. Moreover, the informal logic of induction and credibility is mirrored quite precisely in the qualitative form of the calculus of probability. He develops this correspondence in considerable detail. But he also includes several examples to show that any attempt to use the probability calculus quantitatively leads to foolishness.

Polya's basic attitude toward heuristic and the enterprise of its discovery and elucidation is nowhere better seen than in his treatment of this matter. It is, to put the matter simply (and recursively), heuristic. Polya's own catch phrase is "*investigating induction inductively*" (Polya, 1954 vol. I p*viii*). The mathematics of probability entails the existence of cardinal measures; there is no way in which the axioms of probability can be satisfied uniformly but qualitatively over a domain without also implying that it can be applied quantitatively, though obtaining relevant numbers on specific occasions may be frustrated or only approximately accomplished, due to a variety of causes.[6] But Polya appears to have had no urge to push the analysis into such directions, either to provide some qualitative calculus that can exist without being quantifiable or to ask wherein the use of the probability calculus breaks down in its applicability to heuristic situations, so that the entailment doesn't carry though.

2.6. Heuristics

Polya is usually thought of as a source of heuristics for problem solving. Heuristics, as every AIer knows, are rules of thumb and bits of knowledge, useful (though not guaranteed) for making various selections and evaluations. The classical tongue-in-cheek example is the advice to beginners in chess: *Always check, it may be mate* (Selfridge, 1959). A large, useful set of heuristics is found in Lenat's (1976) AM program for conceptual discovery in arithmetic. An example from his set is, *An analogy is interesting if it associates (for the first time) two concepts which are each unusually filled out.*

In referring to Polya's work, I have used the singular *heuristic*, to indicate thereby a body of doctrine, rather

---

[6]The search for a notion of *subjective probability*, in contradistinction to *objective probability* satisfying a frequency interpretation, explored this terrain thoroughly (Savage, 1954).

than the plural *heuristics*, to indicate a collection of rules.[7] As set out here, this doctrine contains elements of a theory of the architecture, a collection of methods, and some strong principles respecting the role of auxiliary problems and mixing learning with performance. The question is whether Polya's heuristic contains heuristics. The answer is that it does, but that they do not seem to be of great importance. In general, the rules of thumb in his work are parts of his integrated system of heuristic, showing up primarily as methods (eg, his main method). However, here are a few examples of separate heuristics that do occur in Polya's work:

1. Work in letters not numbers. (Polya, 1945 p58)

2. Treat symmetric elements symmetrically. (Polya, 1945 p47)

3. If you can't construct a figure, construct one geometrically similar. (Polya, 1965 p9)

4. Check only "touchy" parts of an argument. (Polya, 1945 p17)

5. If the unknown is a derivative, then introduce derivatives generally. (Polya, 1945 p29)

6. Two proofs are better than one. (Polya, 1945 p60)

7. Good problems, like mushrooms, grow in clusters. (Polya, 1945 p64)

8. Do not believe anything, but doubt only what is worth doubting. (Polya, 1945 p197)

9. Teachers should ask questions that are simple, general, unobtrusive and from a short list. (Polya, 1945 p20)

10. An idea which can be used only once is a trick. If one can use it more than once it becomes a method. (Polya & Szego, 1924 p viii)

All of these appear useful, but they do not comprise a coherent body of advice.[8] The justification for many of them is patent, as in the advice to formulate problems algebraically rather than numerically, which permits many additional operations to occur easily, such as finding special cases, applying the problem to several cases, etc. Others are perhaps primarily psychological. By stating something in paradoxical form -- two proofs versus one, not believing versus doubting -- a simple truth is commanded to the attention. A few actually carry rather subtle insights, such as the next to last heuristic on the questions teachers should use. The first

---

[7] Having used a linguistic variation to convey a matter of substance, a comment on etymology is in order. The root sense of the term *heuristic*, which pertains to *aiding discovery*, is not at issue. Dictionaries define *heuristic* both as an adjective and as a noun. As an adjective, the examples in the prior paragraph are *heuristic principles* or *heuristic advice*. As a noun, *heuristic* (also *heuretic*, a variation due to Sir William Hamilton) means the science or art of discovery, ie, a body of doctrine. In AI, standard usage extends the adjective to be a noun, so that a *heuristic device* becomes simply a *heuristic*. This does not appear to be recognized in any dictionary yet, though it happened a quarter century ago when we deliberately decided that the adjectival form would be too awkward and therefore nouned the adjective (Newell, Shaw & Simon, 1957).

[8] Attempts in AI to lay out a large body of heuristic, as in AM (Lenat, 1976) or MYCIN (Shortliffe, 1976), often do appear somewhat systematic, though no attempts have been made to study this aspect of such bodies of knowledge.

three characteristics (*simple*, *general* and *unobtrusive*) are perhaps obvious. But why should the questions be *from a short list?* Polya's reason is grounded in some not completely obvious psychology: The goal is to get the student to assimilate such questions to his own use; therefore a little stereotypy will let the student believe he could think of such advice himself.

2.7. Summary

There is of course more to Polya's heuristic than has been laid out. Some things have been left out and substantial expansion is possible within the main headings that have been sketched. In a serious exegesis of Polya's work, an important question would be the extent to which the ideas above (possibly in expanded form) cover the content of Polya's works on heuristic. I do not intend to do that here. The points brought out reveal much of what is there and are sufficient for investigating the relation of Polya's heuristic to AI, the task of this paper.

3. Six Theses.

The body of work just reviewed is not only relevant to problem solving but, on the face of it, relevant to AI. As already mentioned, AI has treated Polya with as much distance as respect, with the implication that little can be taken from his work, except perhaps inspiration. What might account for this state of affairs? In particular, what might justify it? Let us consider the possibilities. They will come rather directly from how the ingredients of Polya's heuristic were described.

3.1. Thesis: Polya's Heuristic is Not Relevant.

The first thesis is simply to accept the situation for what it appears to be on the surface, namely, that Polya's work is irrelevant to AI, except as inspiration. The basis for this lies in the difference in aim between teaching problem solving and constructing intelligent programs. Teaching leads toward qualitative generalities and maxims -- what can be assimilated by human problem solvers. It leads away from precise formulations -- which are useful only if analysis is the aim.

I reject this thesis. Positive grounds for rejection can come only from showing (in succeeding sections) the relevance of Polya's heuristic in particular detail. However, its irrelevance is certainly implausible on the face of it. On the one hand, a number of problem solving methods are enumerated in Polya, some of which are in use, others of which are not. That surely seems like relevance. On the other hand, no programs existing today come close to being able to do the sorts of exemplary problems that Polya sprinkles through his text in great profusion. Certainly programs exist that do problems that are difficult by human standards, though it is difficult to assess whether they are comparable in difficulty to the problems in Polya. However, all such programs are substantially narrower; none handle anything like the range of problems that exists in Polya.

In any event, this must remain the default thesis.

3.2. Thesis: Human Psychology is too Different

A thesis, similar to the first but narrower, is that the heuristic of Polya is too specifically adapted to human psychology to be relevant to AI. Indeed, there is a great deal in Polya that stems from the three features listed earlier: attention focusing, associative retrieval, and motivation. It is difficult to find ways to make this aspect of the work relevant to AI. Even though there are problems of attention focusing and associative retrieval in AI systems, no AI system has the problems that Polya addresses, which is how to modify behavior through exhortation and rhetoric. However true and important, what Polya has to say substantively about these functions is elementary. His main problem is how to attain some modification of the novice problem solver's behavior through repetition with variation in interesting contexts.

The psychology-dependent aspects were separated out in our description of Polya's work to make it clear that much else exists besides this psychology. This thesis seems unreasonable, ultimately, because of the relevance of all the rest.

3.3. Thesis: Polya's Methods are either Known or Inapplicable

As our illustrations attempted to indicate, a substantial component of Polya's heuristic can be stated in terms of specific problem solving methods. Distinct possibilities exist for how these might enter into the failure of AI to make use of Polya's heuristic.

One extreme is that Polya uses methods that AI simply cannot assimilate or use. However, the list we gave makes transparent that this is not the case in any obvious way. Not only is the form in which these methods are cast similar to the form in which AI methods are often cast (Newell, 1969), but some of them are even identical. Polya's methods seem quite appropriate to AI.

The other extreme is that AI already knows all the methods Polya has to offer and has long since assimilated them to its use. Again, the list makes transparent that this is not so. Although some of the methods on the list are clearly in use in current AI programs, others clearly have never occurred in AI.

In sum, it seems unpromising to look for the explanation to lie in some fashion within the types of specific methods.

3.4. Thesis: The Action is Outside Polya's Heuristic

One is struck when reading Polya -- and indeed much other work that attempts to instruct in how to solve problems -- by how general the advice is. There seems to be a large gap between the advice and what is actually required to solve the problem. It is entirely possible that all the advice that Polya formulates, even if assimilated and assiduously applied, goes only a very small way toward helping solve problems. In short, the action lies outside Polya's heuristic. AI, in pursuing the actual difficulties in obtaining an intelligence, has found no need to use the sorts of material reflected in Polya.

There are actually two forms of this criticism. The strong form asserts that -- pure and simple -- Polya's heuristic does not contain the important knowledge for solving problems. It is not of great use either for humans or for AI systems. The weak form asserts only that it does not provide what AI requires. It may be quite useful for humans (or perhaps for novice humans). Irrelevance for AI may arise because AI must be concerned with basic system and representation issues that are not problems for humans, being already built in. Alternatively, human and artificial intelligence could simply be different sorts of things (although our remarks about methods do not make this seem likely). In any event, we are concerned exclusively with the

implications for AI. The advice could make large differences for human problem solvers, given their existing abilities and knowledge, but be all but useless in the present state of AI.

The grounds of this issue are not evident in the summary of the previous section. Consequently, let us consider an example (Polya, 1945 p130):

> *Problem:* Write numbers using each of the ten digits exactly once, so that the sum of the numbers is exactly 100.

Following Polya's injunctions to draw a figure (ie, create a representation of the situation) and state what is the unknown and what are the conditions, yields the series:

$$19 + 28 + 30 + 7 + 6 + 5 + 4 = 99$$

This set of numbers does not sum to 100. However, it can easily be fixed up:

$$19 + 28 + 31 + 7 + 6 + 5 + 4 = 100$$

Unfortunately, the digit 1 is now used twice and the digit 0 not at all.

Some more experimentation leads to the conjecture that there really isn't any way to do this -- that in fact the task is impossible. The problem now shifts to being a problem to prove instead of a problem to find:

> *Prove:* It is impossible to write such numbers as required in the main problem.

The first thing to be noticed is that sometimes it makes no difference what way the numbers are composed, only the digits themselves are relevant. For example, 19 might just as well be written 10 + 9; or the 19 + 28 could just as well be 18 + 29. However, 19 is very different from 90 + 1. So what does make a difference is whether a digit ends up in the tens column or in the units column. Exploiting this, the problem can be represented as starting from a basic equation:

$$0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45$$

Any attempted solution will move some of the digits to the tens column:

$$\begin{array}{lllllll} 0 & + 2 & + 3 & + 5 & + 7 & + 8 & + 9 = 34 \\ 10 & & + 40 & + 60 & & & = 110 \end{array}$$

The total sum will be the sum for the units and the sum for the tens, namely 34 + 110 = 144.

This way of looking at the problem suggests considering the sum of the digits that are moved into the tens column:

> Let T = The sum of the digits moved to the tens column.

Now the condition of the problem can be expressed succinctly:

$$10T + (45 - T) = 100$$

$$T = 55/9$$

But 55/9 is not an integer, while T surely is, being the sum of digits. Therefore we have shown (in a reductio ad absurdum proof) that the original problem is impossible.

For us, the important part of this problem is that, right after posing the proof problem, Polya remarks:

> "Some of the figures denote units and others tens. It takes a little sagacity to hit upon the idea that the sum of the figures denoting tens may be of some importance."

A little sagacity, indeed. This may be seen to be the key step in the problem -- the hidden element.[9] What help has Polya provided for seeing this key step? All of his suggestions, his main methods and his remarks, all deal only with the peripheral issues. To be a good problem solver is to be able to have such good ideas without undue flailing around. That is where the action is in problem solving!

At least so goes the thesis. I think it must be taken seriously. There does not even exist, so far as I know, good evidence on whether Polya's advice, taken to heart, actually makes for good problem solvers.

Interestingly, this issue is echoed within AI itself. On the one hand, AI tends to focus on the formulation of general reasoning methods and heuristics, such as means-ends analysis. On the other hand, problem solving success seems to result from large amounts of knowledge producing intricate orchestrated patterns of behavior. It seems to many that the general methods carry almost no information about how to behave -- that they are analogous to Polya's advice. They may even be essential -- that is not often in dispute -- but they are not where the action is in intelligence, artificial or otherwise.

3.5. Thesis: Future Orientation is still Beyond AI

A thesis that seems quite different on the surface from any considered so far is that the key element in Polya is learning, ie, that the primary function of each problem solving session is to prepare to solve future problems, not to solve the particular problem at hand. The reason AI has not been able to make use of Polya is that it is not prepared to engage in such learning.

In fact, this thesis reflects the same general view as the previous thesis about Polya's advice not being where the action is (at least the major version of that thesis). Both say that the main aspect of successful problem solving is detailed knowledge of a massive number of specific problems, solutions and methods. Focusing on the learning is just focusing on how that knowledge gets built up, rather than on having it. In both cases, the general situation is summarized by Figure 2-3, in which the general advice of Polya relates only to the rind, but the action is all in having (and getting) the inner area. Thus, if this thesis were to hold, it would explain

---

[9] Perhaps it could be taken as two steps: first abstracting the digits from the numbers (seeing 40 + 3 = 43); and then representing the sum of all tens digits (T). But it comes to the same thing, as far as the point of the example for us.

away the prior thesis. The importance of the example is not in the process that discovers using "the sum of the figures denoting tens", which depends primarily on earlier history, but in putting away one more problem instance (or some aspects of it) that permit a slight extension of problem solving on some future occasion.

Learning has always been an important part of AI. However, it has never yet been the main line of development, AI always giving priority to performance over learning.[10] AI's progress in constructing learning programs can be tracked roughly by the problem space of performance programs within which the learning program was able to search. Most learning programs have been simple hill climbers in a space of parameters to the performance program. Examples are the weights on Perceptrons (Rosenblatt, 1961) and the coefficients of an evaluation function (Samuel, 1963). The work on concept learning and induction programs worked in spaces of program-like expressions, richer structurally than n-dimensional parameter spaces, but still quite limited because the concepts themselves were limited. Examples are sequential concepts, such as ABC BBD ABE BBF --- (Simon & Kotovsky, 1963), and block-like structures (Winston, 1975). Much richer learning spaces still are used in automatic programming (Green & Barstow, 1978), but even here the end structure is still simply a program.

The issue of learning from present problem solving to future problem solving, which is what Polya urges, seems more like learning an entire set of dispositions and augmenting a present total problem solving capability. A recent thesis by Don Cohen at CMU (1980) on a learning theorem prover is interesting in this regard. His system builds up its performance program (for proving theorems in a typed logic) entirely from the theorems it proves, assimilating each one in the several different ways it can contribute to future problem solving. But, though this is a step in the right direction, it shows that there is a long way to go. It would seem that this thesis, too, is a genuine possibility.

3.6. Thesis: The Diversity of Auxiliary Problems is still Beyond AI

The final thesis is responsive to the other main feature of Polya's heuristic that we noted in the previous section, namely its intensive use of auxiliary problems. These appear to be of immensely diverse character, both in terms of their content relative to the original problems, and in the way they are used to aid the original problem. Though Polya offers some description about the use of auxiliary problems, as we noted, it is mostly the actual mathematical content that appears to be important.

Thus, it seems plausible that equivalent behavior is simply beyond AI in its current state of development. Certainly, no existing AI system can handle auxiliary problems in anything like the manner in Polya's work.

---

[10] The fundamental reasons for this are entwined with many other factors. There were paradigms that did give maximum prominence to learning, eg, the approach through *self-organizing systems*; and they went down different and less productive paths.

The source of this deficiency is not evident, eg, whether it is beyond AI because of some major deficiency in its current repertoire of mechanisms or whether it indicates simply that AI has failed to attempt problem solving systems of this type. In any event, this thesis seems to be a major candidate.

3.7. Focusing on a single thesis

We have enumerated and briefly assayed six theses for why Polya's work on heuristic has not made much impact on AI. They start with the default thesis that matters are just what they appear to be -- Polya has obtained exactly the attention his work deserves. They then range over the major features of his work, raising whether this or that feature is the important one. They are not monolithic in their implications. Some point in the direction of current limitations in AI; others point toward limitations in Polya's work. The theses do not seem to be mutually exclusive, though some work against each other to some extent.

Notably, all six are substantive theses. The generally positive attitude toward Polya's work and its widespread acquaintance throughout the AI community precludes simple explanations of a sociological sort. The example-driven and discursive character of Polya's work might be taken by some as a kind of sociological barrier, taking it as a matter of style. But this aspect seems much better interpreted as being an essential part of Polya's view of problem solving. The examples are necessary, because beyond a certain point, solving a problem implies dealing with its content, not any generalized descriptions of it. From AI's standpoint, though qualitative statements of heuristics might be difficult to assimilate in operational form, the example-driven format is tuned *exactly* to its needs. Most AI programs are generated by attending to a small handful of spanning examples.

No doubt, these six theses are not all that could be generated. Since there is no space to deal with more than one in any detail, proliferation of additional theses seems pointless. Some of the theses seem implausible on surface examination and have been so labeled even as they were introduced. But several remain, notably, the failure of Polya's work to contain the important heuristic knowledge, the inability of AI to handle complex learning, and the inability of AI to handle diverse auxiliary problems.

We will select the last thesis for further discussion. There are some grounds for this choice. The assessment of whether the action is really elsewhere is a more ambitious and technical undertaking. In fact, it is not clear whether the conceptual tools exist currently to do the analysis. With respect to the learning thesis, I have always had a predilection that learning issues can be understood only against an understanding of performance issues. That operates here. Major aspects of the learning issue require an understanding of what it means to learn about auxiliary problems. Without understanding the nature and role of auxiliary problems such an analysis must remain superficial. In any event, the remainder of the paper is devoted to the sixth

thesis, on auxiliary problems.

4. Thesis: The Diversity of Auxiliary Problems is still Beyond AI

The starting point for this thesis is the diversity in Polya of the problems created in the course of solving a given main problem. Not that a large number of auxiliary problems are considered in any single attempt, in general only one or two are. But when the auxiliary problems for problem X are compared with those for problem Y, they seem to have almost nothing in common. The examples given earlier of finding the center of gravity in two dimensions to provide the method for the analogous problem in three dimensions; and of solving a minimum distance problem in a special case (one ship at rest) to help solve the problem in full generality make the point.

In a superficial way, Polya's use of auxiliary problems can be captured in a simple general method:

Problem: To solve problem, *P:*
Auxiliary-problem method:
1. Find an auxiliary problem, *AP.*
2. *Subproblem:* Solve problem *AP.*
3. Use the solution.

Earlier, Polya's main scheme was claimed to be a method by AI standards. The scheme for using auxiliary problems is certainly a part of that, hence should be a legitimate method as well. But now it is necessary to examine whether it really can be fitted into the sorts of methods that AI can currently handle or poses new types of demands.

First of all, Polya uses the concept of auxiliary problems very broadly. On the one hand, it includes examples such as we gave above (the center of gravity and the ships problems). On the other hand, it includes examples such as the following (Polya, 1945 p132):

Main Problem: Find X satisfying the equation:
$8(4^X + 4^{-X}) - 54(2^X + 2^{-X}) + 101 = 0$

Let $Y = 2^X$

Auxiliary problem #1: Find Y satisfying the equation:
$8(Y^2 + Y^{-2}) - 54(Y + Y^{-1}) + 101 = 0$

Let $Z = Y + Y^{-1}$

Auxiliary problem #2: Find Z satisfying the equation:
$8Z^2 - 54Z + 85 = 0$

$Z = 5/2$ or $Z = 17/4$

For Polya, each of these substitutions produces an auxiliary problem. However, this is an instance of the *Method of convertible transformations* and these steps are garden-variety subproblems, such as occur in AI symbolic manipulation programs. Finding the auxiliary problem is one of selecting among a set of possible operators, one of which is substitution; using the auxiliary problem is simply setting the goal of going the rest of the way. We wish to attend primarily to subproblems that do not fit so obviously within the existing AI notion of method. When we refer to *auxiliary problems* we will always understand the more exotic of the versions that occur in Pólya.

4.1. The Space of Auxiliary Problems

The first impulse is to think of auxiliary problems as elements in a *space of problems* that come to exist around a main problem. Most of the problems in this space will never be generated in a given attempt, but they all exist *potentially*. This impulse arises because the basic formulation of problem solving in AI is as search in a space of problems. It would be the natural way for AI to assimilate auxiliary problems. With this view the thesis becomes:

> **Thesis (reformulation):** Auxiliary problems require problem spaces that are still beyond AI's capability.

4.1.1. Problem spaces

To investigate this thesis requires sketching the AI paradigm of *heuristic search*, though the story is familiar. A central hypothesis in AI is that intelligent action occurs by means of search in a *problem space*. A problem space consists of a set of *states* and a set of *operators* that when applied to states yield new states. Thus, the operators provide *paths* through the space. An agent, being at some state, can apply a sequence of operators to trod some path, eventually arriving at other states. A *problem* is given by giving an *initial* state and specifying in some fashion a class of *goal* states plus possibly some *path constraints* -- the problem being to get from the initial state to any of the goal states along a path that does not violate any of the constraints.

Problem solving in a problem space is necessarily search. However, the search need not be blind. The problem solver has *knowledge* about the space, ie, about the operators, about paths, etc., knowledge that can be both general and specific. If there is enough knowledge (and if the problem solver can actually bring it to bear) there may be very little search indeed. The problem solver may know exactly which operators to apply at which points. If there is little knowledge (or if it cannot be brought to bear), then the search becomes increasingly combinatorial in character. In fact, uncertainty in the application of an operator always translates into the chance of search. Thus, the combinatorial character of search follows inevitably from the cascading of uncertain applications. The term *heuristic search* has come to mean this combinatorial search, modulated as much as possible by whatever knowledge (the *heuristics*) can be brought to bear.

A problem solver's basic capabilities are expressed by the problem spaces it can employ. It is able to work on any problem in a space. It may not solve the problem, but it can attempt it and engage in appropriate problem-solving behavior. Problems outside this space lie outside the problem solver's competence or even comprehension. A problem solver may be able to work in many problem spaces, its total competence being expressed in their union.

All the earliest AI programs for playing games, proving theorems, doing symbolic mathematics, etc, were transparently heuristic search programs (Feigenbaum & Feldman, 1963). It gradually became clear that heuristic search was not simply one particular technique among many for solving problems, but was a central component of all intelligent action (Newell & Simon, 1976). Even where the knowledge is sufficient, there is always a contextual threat of combinatorial search. Indeed the hypothesis has gradually evolved that problem spaces not only exist for genuinely problematical situations, but for routine ones as well (Newell, 1980).

4.1.2. The character of actual problem spaces

Abstractly considered, there seems no reason why problem spaces should not exist that include the auxiliary problems that arise in Polya. To pin this down requires examining the character of the spaces that AI has actually constructed. Very briefly, here are a few examples:

- GEOMETRY THEORIST. The task is proving a theorem in elementary synthetic geometry (Gelernter, 1963). The space consists of all abstract plane geometric figures made up of points and lines, hence including angles, triangles, etc. The operators are theorems. The initial state is the conclusion of the theorem to be proved, the goal state is any of the givens of the theorem (which are already known to be true). The heuristic knowledge pertains to which theorems are similar to which figures (hence, presumably relevant targets for a proof path), which are trivial variants of others (eg, syntactic transforms), and a diagram as a *model* of abstract geometry. An additional mechanism is used to apply this latter knowledge: constructing a diagram (ie, a model) for each generated figure (ie, element of the problem space) to test its acceptability.

- SAINT. The task is symbolic integration as in a first year calculus course (Slagle, 1963). The space consists of all algebraic expressions containing signs for integration and differentiation. The operators are known integrations (as in a table of integrals), plus an array of algebraic simplifications and manipulations. The initial state is the original expression, the final state is any expression that no longer contains integral signs (hence, the program works backward). The heuristic knowledge pertains to when expressions seem integrable, which operators are useful to apply, what are simple expressions, etc. It is brought to bear through mechanisms for selecting or rejecting operators and states -- which are the basic techniques for making search intelligent.

- DESIGN PROBLEM SOLVER. The task is laying out spatial structures, such as arranging the furniture in an office (Pfefferkorn, 1968). The space consists of all possible partial layouts. The operators place a given piece of furniture in a specified place satisfying various relations -- a given location, next to another object, avoiding a restricted space, viewable from a given point, etc. The initial state is an empty room. The goal state is a room with the full complement of furniture and also satisfying a given set of constraints. The heuristic knowledge pertains to preserving as much uncluttered space as possible, the greater difficulty of fitting in larger objects, and the

irreversibility of the failure of constraints. Additional mechanisms to bring this knowledge to bear are a *causality analysis*, which assigns a cause for the failure of a given constraint, and then backtracks far enough to avoid that cause on future tries; and the construction of *macro-objects*, which weld together sets of objects with tightly interrelated constraints.

- NOAH. The task is spatial movement of objects, such as stacking blocks on a table (Sacerdoti, 1977). The basic space consists of all possible ways of stacking subsets of blocks on the table. The operators consist of basic actions (picking up a block and setting it down on a space) plus conditions for its feasibility (eg, no block is on top of the block to be moved). The heuristic knowledge pertains to satisfying all the conditions plus the interrelations between satisfying one type of condition and another. An additional mechanism for bringing this knowledge to bear is the creation of hierarchies of abstract *plans*, which specify incomplete sequences of actions and unsatisfied constraints. Thus, in fact, NOAH really works in the space of plans, where the operators expand plans to additional levels of detail, and rearrange plans to remove redundancies, avoid conflicts and coordinate actions.

- AM. The task is discovering new concepts in arithmetic (Lenat, 1976). The space consists of a domain of data structures that represent arithmetic concepts (eg, prime numbers), and are composed of attributes such as, generalization-of, specialization-of, examples, extreme-instance, etc. The operators are ways of forming new concepts from old, eg, by specialization or by generalization. The system has no goal in terms of goal states; instead it seeks to make *interesting* concepts, using a number of rules for determining what is interesting (rather than a measure). The operators themselves contain most of the heuristics.

- EL. The task is analyzing an electronic circuit (Sussman, 1977). The problem space consists of all partially analyzed electronic circuits, ie, circuits augmented by constraints that describe relations between the voltages and currents that occur in the circuit. A problem starts with a circuit with no analysis and the desired final state is one which contains sufficient constraints to give the information desired and to verify that the analysis is consistent. An important feature of the search is *dependency directed backtracking*, in which when a contradiction is generated (which shows up as a failure of a particular type of constraint), only those constraints are removed that have caused the difficulty. This is possible because constraints can be known to be independent of the difficulty.

- MOLGEN. The task is planning a genetics experiment (Stefik, 1980). The basic space consists of a domain of data structures that represent genetics experiments. An experiment is a sequence of various organisms and treatments to produce new organisms with desired properties. The problem space of MOLGEN is a space of partially specified experiments, ie, abstract *plans* for experiments. Three kinds of abstraction are possible: classes of organisms, rather than a specific organism; classes of experimental treatments, rather than a specific treatment; and *constraints* on the specific organisms and treatments that can occur at a given point in the experiment. The operators produce expansions of a plan (ie, refinements), and analysis of a plan which generate constraints on other parts of the plan.

This collection, spanning the entire history of AI, is only a sample out of many times as many AI programs that are transparently heuristic search.[11] They exhibit a wide range of tasks, a wide range of heuristic

[11] Many other AI programs do not appear on the surface to be heuristic search programs: this list is not meant to characterize all AI programs.

knowledge, and a wide range of heuristic mechanisms for putting that knowledge into effect -- selection and rejection of operators and of states, evaluation functions, models, causality analysis, planning, etc.

4.1.3. Object-centered problem spaces

A great communality exists in these problem spaces: They are all *object-centered.* This property can be introduced by an idealized example of elementary theorem proving:

> *Given:* Known theorems, $T_1, T_2, ..., T_n$, each of form $H_i \supset C_i$.
> *To prove:* New theorem, $H \supset C$.

In the current art, this problem would be cast as a problem space as follows:

> *States:* Well-formed formula, W_i.
> *Operators:* One for each known theorem, $Q_T(H_T)\ C_T$.
> *Initial state:* The hypothesis of the theorem to be proved, H.
> *Desired state:* The conclusion of the theorem to be proved, C.

The states of the space are taken to be the basic well-formed formulae of the logical calculus (ie, expressions such as H and C). The operators are defined by the given theorems, which take as input any state that matches H_T under substitution of variables and which output the corresponding instantiation of C_T as a new state. The problem in the space is formed by splitting the theorem to prove into two parts: the initial state becomes H and the goal state becomes C. Many variations on this occur, from working backwards (initial state C, final state H, with inverted operators), to using AND/OR trees because the conditions are conjunctions of the expressions (C_1 & C_2 & ... $C_k \supset H$), to permitting the use of additional transformations (eg, definitions) in determining the inputs to operators. None of these complications is of concern here.

What is of concern is that the given problem is to find a *proof.* Yet, the problem space is a space of *expressions,* not a space of *proofs.* The proof itself is there, of course. It is implicit in the exploration of the problem space, being the path that is found from H to C. But a genuine reduction of the problem has occurred in casting it in this form. The basic steps are what next theorem to use. The heuristics of selection, rejection, and so forth, pertain to this decision.

This is a creative reduction, responsive to the underlying structure of the theorem proving task. However, it is a very limited way of solving this problem. Of course, no proofs are actually lost, since ultimately all proofs are expressed as a sequence of theorem applications. What is lost is the ability to discover a proof by taking an imperfect proof and improving it. Or, to come to the pervasive occurrence of auxiliary problems in Polya, the ability is lost to start with a wrong proof (because it is the proof of a different theorem) and to transform it into a good proof. A problem solving program that works in such an object-centered (here, expression-centered) space is unable to see the forest of proofs for the trees of expressions. It knows only about the space of expressions, not about the space of problems.

The difference between the space of problems and the space of expressions in theorem proving is only a special case. More generally:

> *Object-centered problem space:* A problem space whose states are generated from the objects in the problem by structural composition operations.

This is to be contrasted with:

> *Problem-centered problem space:* A problem space whose states are generated from problem statements.

In the first case the objects are decomposed into their parts in the obvious structural way (eg, as logic expressions are composed recursively from alphabets of connectives and terms). In the second case, the entire problem statement is taken as the kernel for generation, the constraints and conditions as well as the objects. All the examples in our list show this same property -- they are instances of object-centered problem spaces, rather than problem-centered ones.

The GEOMETRY THEORIST is a theorem proving program, hence has a structure directly analogous to our paradigm example. The existence of the geometric diagram does not extend the types of problems at all. SAINT, the symbolic integration program, works in the space of algebraic expressions; it can only take an expression and make another try at integrating it. It cannot take an attempt at integration which leads to a similar, but not quite right, result and attempt to transform that attempt into one that is satisfactory. To do so it would need to work in a space whose elements are entire sequences with associated transformations, manipulating it in ways other than adding one more expression to the end of an existing sequence (which is what the object-centered space effectively permits). For the DESIGN PROBLEM SOLVER the important generalization on problems comes from the constraints. It cannot work in a space of problems in which it manipulates the constraints, creating subproblems that involve other sets of constraints than those originally given and using such problems as approximations to the main problem.

Planning, as it occurs in NOAH, does not evade the difficulty, though it takes a step in the right direction. Planning does work with sequences, because the goals in its plans represent undetermined sequences of actions in the basic space. They certainly expand the total set of problems that can be represented. However, this is not enough, because the planning arises from a too simple abstraction of the fully detailed problem space, namely, only ignoring conditions. The class of approximate problems that can be generated this way is still extremely limited, encompassing only a few of the types of auxiliary problems in Polya, eg, those that are generated under the heuristic rule of dropping a condition.

AM, the program for discovering arithmetic concepts, provides no exception to this generalization, because AM has no problems at all. Thus, there is no way to discuss what it would mean to have a problem-oriented problem space.

By now the pattern of analysis should be clear and its application to the last two programs of our list, EL and MOLGEN, can be left as an exercise for the reader.

4.1.4. Alternative paradigms

Not all of AI looks like search in a problem space. Indeed much recent progress in AI has taken place in areas such as memory structure, knowledge representation and natural language, which do not transparently exhibit this search structure. Programs, such as Winograd's SHRDLU (1972) or the several efforts by Schank and his students (see Schank & Ableson, 1977), are appropriate examples. Perhaps we are simply looking at the wrong collection of AI programs; perhaps programs within some alternative paradigm would permit handling auxiliary problems of the type Polya uses.

The answer is negative. The tasks performed by these other programs are not intellectually demanding (for the programs). That is, it is not possible to give these programs tasks that are *problems* by human standards, ie, that require figuring out.[12] That their performance is often interesting is due to the body of knowledge they contain (and their scientific interest derives from the discovery and representation of that knowledge). They bring this knowledge to bear essentially by an act of recognition or sometimes (as in SHRDLU's block manipulation) by a thorough analysis and preprogramming of the task.

4.1.5. Conclusion

This section has revealed that if auxiliary problems are to be taken as elements of problem spaces, then AI, with its current limitation to object-centered problem spaces, has not yet developed the capability for coping with auxiliary problems. However, though the concept of problem spaces is clearly general enough to encompass auxiliary problems, we have not established that this is the appropriate way to view them.

4.2. Finding an Auxiliary Problem

Let us now examine the steps of the auxiliary-problem method in more detail (see page 26). The middle step is unexceptional, since it is the general recursive step common to all methods of solving a subproblem by the total intelligence of the problem solver. But the first and the last steps need scrutiny. In this section we look at the first step. As already noted, Polya asks a wide array of questions to help the problem solver find an auxiliary problem. It is worthwhile to have a list of these suggestions:

1. Have you seen the problem before?

2. Have you seen the problem in a different form?

---

[12] An exception is POLITICS (Carbonell, 1978), a program that produces ideological interpretations (linguistically described) political situations; but this program contains several heuristic search mechanisms.

3. Do you know a familiar problem with the same unknown?

4. Do you know a familiar problem with a similar unknown?

5. Here is a related problem and solved before. Could you use it? Its result? Its methods?

6. Solve first some related problem.

7. Do you know a more accessible related general problem?

8. Do you know a more special problem?

9. Do you know an analogous problem?

10. Use the data somehow! Ie, create a problem using the same data.

11. Solve the problem keeping only part of the condition.

12. Restate the problem. Restate it still differently.

Polya's view (see Section 2.1) is clearly that auxiliary problems are to be found in the problem solver's prior experience, ie, in his long term memory. The difficulty for the problem solver is making *contact* with them. Memory is content addressed, and the retrieval cues are dependent on where attention is focused. Thus, it is useful to provide all sorts of cues to help make contact and this leads to the multiplicity of Polya's suggestions.

Besides the sheer number of suggestions, this list has a striking feature. For many of the suggestions, generating an auxiliary problem is not a constructive task, but a task of retrieval from long-term memory (1, 2, 3, 4) or a gift from an oracle (5). This contrasts sharply with the situation in AI. For instance, here is a typical AI example:

> *Problem:* To get from an initial state *A* to a desired state *B*:
> *Means-ends analysis method:*
> 1. Compare *A* to *B* to detect some difference *D* between them.
> 2. *Subproblem:* Eliminate *D*.

This subproblem arises from the original problem according to a definite procedure. It is reasonable to think of it (the method of generating the subproblem) as an operator in a space of subproblems. Given a collection of such operators, it is reasonable to think of a space of subproblems being searched generatively, in a fashion similar to all heuristic search programs, with all the various techniques of selection, rejection, evaluation, planning, etc. to help out. However, if subproblems are generated by recall or oracles, then no such structure exists out of which to compose a space of problems. Each subproblem is more like an isolated point, available only because of historical accident.

Even where guidance is given that might help generation, it is mostly highly general (eg, 6 to 10). Only the last two suggestions (11 and 12) seem specific enough to be used in current AI programs, eg, dropping a condition, as in NOAH. Such general constructions have not been used in AI methods. A step in this direction is taken by AM, with its operators for generalizing and specializing. As noted already, AM does not actually formulate problems, but only concepts related to the original. Consequently, it avoids whether these related notions could form tractable auxiliary problems or whether they would be useful in solving main problems, even if solved. However, its success in generating interesting concepts, without reference to any problem to be solved, provides another clue that the generation of auxiliary problems is a much looser operation than the generation of subgoals in the typical AI methods.

In conclusion, it appears that the auxiliary problems may not form a problem space, but that they are simply isolated problems. They can bear any of a wide number of weak relations to the main problem, without any commitment to its problem-solving relation, eg, whether it can even be worked on.

4.3. Using an Auxiliary Problem

We now examine the third step of the auxiliary problem method, in which the auxiliary problem has been solved and now must be put to use.

4.3.1. The independence of using from finding

The striking feature of the method, which makes it not quite so trivial as it first appears, is the virtually complete independence between *all* its stages. This is unexceptional for the middle step, which is the standard general recursive step of applying all the problem solver's intelligence to a new subproblem and in general is always independent of the rest of the method. However, in this method, what is done with an auxiliary problem once it is solved need bear no relation at all to how it was found in the first place. Thus the method expresses a genuine factorization.

For using an auxiliary problem, Polya provides only a very simple taxonomy: (1) use the result; (2) use the method; and (3) use the attempt. Any auxiliary problem can be used in any of these ways; which way is not determined in advance. In particular, it is not determined by the way the auxiliary problem was either found or solved. Indeed, the third type -- using the attempt -- invariably arises only after the failure to use the auxiliary problem in the other ways. It is a sort of salvage operation, asking whether the attempt itself suggests new auxiliary problems that might be considered.

Once we get beyond these three possibilities, explicit general advice is almost completely absent. In its stead is placed the working through of examples. Superficially, it seems clear why this is so. Everything seems to depend on the particulars of the auxiliary problem, ie, on its *content*, which bears no consistent

relation to the original scheme which generated it. Actually, in using the auxiliary problem as a method, we do know that the mechanics of plan interpretation must be invoked. However, the method itself is still completely open. In using the result of the auxiliary problem, even this much is not clear.

This independence stems in part from the way auxiliary problems are generated. As we have just seen, in general this is not a constructive operation, but a much looser one. Thus there will be few constraints that the auxiliary problem must satisfy by virtue of its mode of generation, hence little that can be carried over to be relied upon when it comes to use it.

Again, this is in sharp contrast to the situation in AI programs. There, the methods that create subproblems *always* specify exactly what to do with the result. One frequent example is apparently trivial, but in fact very important: The subproblem simply produces another state in the problem space. From this follows: (1) problem solving can just continue from this point without any further consideration of the original problem; and (2) the problem solver necessarily has the capability to work on this problem, since this is what it means for a problem solver to have a problem space. An example is the means-ends analysis method of reducing the difference, described above. The result is a new situation in which the difference is reduced; the problem of using this result is simply the problem of finding a path from this new state to the original desired state.

Another major example occurs in using AND/OR search trees. Each such use implies a method, though the method itself is often implicit (and not always the same in each case). In SAINT, for instance, the method is the following:

1. If the expression contains integrals, say X_1, X_2, ..., X_n, extract them from the expression.
2. Solve independently the subproblems; Integrate X_1, Integrate X_2, ..., Integrate X_n.
3. Replace each integral, X_i, in the original expression with its result.

The third step expresses that the method knows exactly what to do with the results of the subproblems. It is precisely coordinated with the first step that creates the subproblems.

The situation with auxiliary problems can be restated in a variety of ways. In terms familiar from design, the use of auxiliary problems is *bottom-up* problem solving, not *top-down* problem solving. The strictures against bottom-up design apply: often things don't fit together. In terms familiar from AI, the use of auxiliary problems is itself *problematical.* Having in hand a solved auxiliary problem is still not to know exactly what to do. Auxiliary problems are not well behaved; they must be brought under control to get any use out of them. On the contrary, subproblems typical of AI methods are under good control and behave in predictable ways -- they are docile. To coin some slightly colorful terms to capture this distinction:

> *Wild subproblem:* A subproblem whose use in the main problem is still a problem, even after it is solved.

Tame subproblem: A subproblem that is not wild; how to use it is already known when it is generated.

Along with this distinction goes a reformulation of the thesis:

Thesis (second reformulation): AI cannot handle auxiliary problems, because AI does not yet know how to handle wild subproblems.

This version of the thesis does not imply why AI cannot do this. However, as far as I know, there are no AI programs that use wild subproblems, as defined above. To make the notion of wild subproblems concrete, let us examine several examples.

4.3.2. Using a result: The Two-ships problem.

Consider again the problem, described in Figure 2-2, of finding the minimum distance of approach between two moving ships. The auxiliary problem was to consider a special case, namely, when one ship was at rest. As the figure showed, the solution to this auxiliary problem was quite easy to find, namely, just drop the perpendicular from the resting ship (A) to the path of the moving ship (B).

How is this auxiliary problem to help? The key idea is that the general case, in which B also moves, can be reduced to the special case in which B is at rest. This is easy enough to see -- consider yourself solving the problem while perched on the deck of ship B; then no matter how fast or in what direction B moves, it will be at rest as far as you are concerned. Ship A will take on a different motion, namely its motion relative to B. But if the ships both travel at constant speed in a straight line, then A will still appear to go in a straight line. The actual solution to this relative motion is given by the standard vector diagram, as shown in Figure 4-1. A velocity vector equal and opposite to QB is impressed on both ships, yielding B at rest and A traveling along AR.

Our problem is not to solve the Two-ships problem, but to understand how auxiliary problems work. Figure 4-2 shows the trace of the problem solving. The method of auxiliary problems is evoked. This says first to find the auxiliary problem. In this case one of Polya's suggested approaches was used, namely, specialize the problem. This succeeded, producing the special case of one ship at rest. The next step was to solve this problem, which was accomplished by a simple geometric construction. The third step was to use this auxiliary problem. In this case, the result was used, by reducing the main problem to it. This was accomplished conceptually by the relativity of uniform motion and computationally by the geometric construction of vector addition.

This trace makes evident that an act of problem solving is needed after the auxiliary problem has been solved -- reducing the main problem to the auxiliary problem simply arrives from nowhere. In general, a specialization of a problem is not equivalent to the main problem. At best one can expect to anchor the main

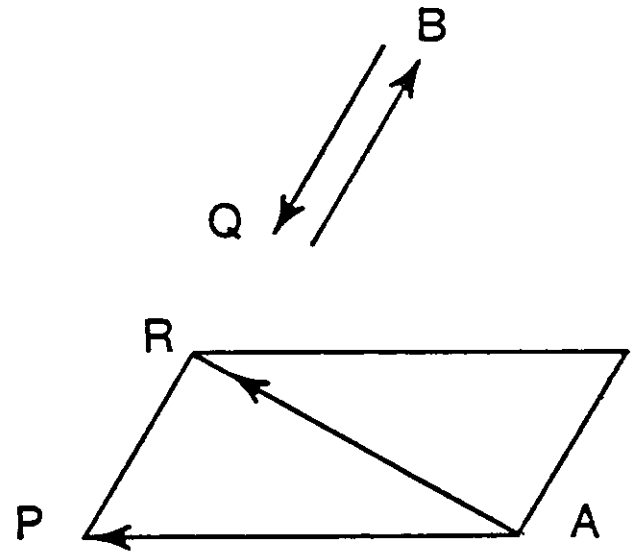


Figure 4-1: Using the result of the auxiliary problem to solve the Two-ships problem

```
Main problem: Find minimum distance between two moving ships
    Auxiliary-problem Method
        Find auxiliary problem
            Specialize
                Solved: Auxiliary problem: Find minimum distance when
                        only one ship is moving
        Solve auxiliary problem
            Solved: Drop perpendicular
        Use auxiliary problem
            Use result
                Reduce main problem to auxiliary problem
                    Solved: Subtract same velocity from both
```

Figure 4-2: Trace of solving the Two-ships problem.

solution (as in the boundary conditions of differential equations), or to be one of a set of cases in a partitioning of the problem. A rabbit has been pulled out of the hat.

Though it is not known in advance how to use a problem, it is certainly possible to pose that very issue as a problem itself. We can give this a name:

The connection problem: How to connect the solved auxiliary problem with the main problem.

This just appears to introduce another phrase instead of "Use auxiliary problem"; however, an important shift has occurred. In the original version of the Auxiliary-problem method, the first and last steps were taken as

procedures and only the middle step was taken to be a recursive subgoal step. Now the last step, to use the auxiliary problem, is taken as containing a general subgoal step as well.

A connection problem is not of the same form as the original problem. It takes two problems as given and asks how one might be related to the other in terms of its solution. Thus, the connection subproblem does not take place in the same problem space as either the main problem or the auxiliary problems. The latter two often take place in the same space, as in this example, where the space comprises the rectilinear movements of two ships.

4.3.3. Using a result: The Trapezoid-construction problem.

Consider another example from Polya (1945 p184):

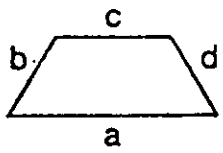
Problem: Construct a trapezoid, given the sides *a*, *b*, *c* and *d*.

We know how to construct triangles given three sides (say α, β and γ), using the method of loci. The loci of each of two sides (say β and γ), attached to the vertices of the common third side (α), can be drawn; they intersect in a point, which is the third vertex. But a trapezoid seems to have too much freedom. In fact, we may also know that in general a quadrilateral is not determined by its four sides. Thought of as a linkage, it will just flop around. But a trapezoid is a quadrilateral with one more condition added, namely, that two of its sides are parallel (say *a* and *c*). This is enough to fix the shape of the trapezoid. The problem, of course, is to actually construct it.

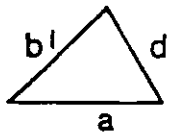
Though it would be useful -- and fun -- to work though this problem as an exercise in heuristic, let us go directly to a trace of Polya's solution attempt (Figure 4-3), to focus on the role of auxiliary problems The connection subproblem is now shown explicitly as the first subproblem under using the solved auxiliary problem.

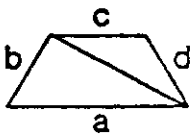
To obtain an auxiliary problem Polya considers *varying the data*, looking for special values that might be interesting. What he finds is a specialization, namely if one of the sides (eg, *c*) is set to zero, then the trapezoid becomes a triangle. Constructing a triangle given its three sides can be assumed to be known. The first step in using this auxiliary problem is to set up the connection subproblem, that is, to find some connection between the auxiliary problem and the main problem. This is called *find an approach*, because all one can expect to find is some way in which the auxiliary problem *might* be useful, ie, some way that will permit focused problem solving to determine whether the approach will pay off or not. The idea behind this attempt at connection is to introduce triangles into the trapezoid. This is easy to do per se (eg, just draw a diagonal), but not in a way that uses the auxiliary problem solution, which requires three known sides.

Abandoning this attempt, Polya considers additional variations of the data. This leads to yet another

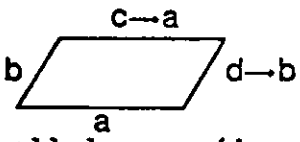


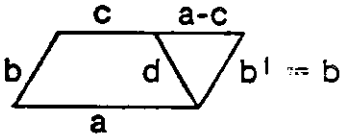

```
Main problem: Construct trapezoid
    Auxiliary-problem method
        Find auxiliary problem
            Specialize
                Succeed: c=0
        Solve auxiliary problem: Construct triangle (Succeed: known)
        Use auxiliary problem
            Connection subproblem: Find an approach
                Introduce solution figure into the main problem
                    Fail: Can't construct triangle with 3 known sides
```






```
    Auxiliary-problem method (try again)
        Find auxiliary problem
            Specialize
                Succeed: c=a
        Solve auxiliary problem: Construct parallelogram (Assume)
        Use auxiliary problem
            Connection subproblem: Find an approach
                Introduce solution figure into main problem
                    Succeed: Extend b' parallel to b
            Solve augmented problem
                Succeed: Simply lay out c along top a
```






```
        Return to auxiliary problem: Construct parallelogram
            Auxiliary-problem method
                Find auxiliary problem
                    Succeed: Use triangle already in parallelogram
                Solve auxiliary problem
                    Succeed: Already solved
                Use auxiliary problem
                    Connection subproblem
                        Succeed: Already introduced in figure
                    Solve augmented problem
                        Succeed: Construct triangle a-c, b, d on top a
```

Figure 4-3: Trace of solving the Trapezoid-construction problem

specialization, namely, setting the two parallel sides to be the same length ($c = a$). This turns the trapezoid construction problem into a parallelogram construction problem. Here Polya doesn't stop to ask if the problem is solvable, but simply suspends the attempt. Though Polya has his didactic reasons for this, in fact, such a move is common in human problem solving, being the essence of planning. Thus, Polya has generated a second auxiliary problem.

Again, the initial subproblem of using the new auxiliary problem is to find a connection, and the same approach is taken, to introduce the result figure (the parallelogram with $c = a$) into the main problem. This is easy to do, just by extending c. Furthermore, this is a connection that works; it is immediately obvious that

the trapezoid can be gotten just by taking the triangle away from the parallelogram, so to speak. Moreover, the triangle is one that is known, namely, all three of its sides are given (d, b, $a-c$), just as in the auxiliary problem.

This leads to returning to the suspended auxiliary problem of constructing the parallelogram. However, the appropriate auxiliary problem for *this* problem is simply waiting in the figure, namely, the original auxiliary problem of constructing a triangle from three sides. This is hardly a wild problem at all, because it is already located within the main problem (constructing the parallelogram), so that no additional connection is required.

Like so many of Polya's illustrative problems, this one has an ingenious twist, namely, the failed auxiliary problem being the key to unlock two doors: both to solve the second auxiliary problem and to use in the main problem.[13] In fact, so intimately are these two parts bound together that it is unclear how problem solving might actually proceed in a real case -- which problem would be posed or solved first. After all, the problem solver is primed for both problems simultaneously and the solutions to both stare him in the face from the same diagram.

Are the first two auxiliary problems wild? (As noted, the third is not.) How they were used was certainly not given. A connection problem was posed and attempted in both cases, even failing in one. Furthermore, the use of these auxiliary problems bears no relation to how they were created, namely, as special cases.[14] Each was used simply as a subproblem, which in no way had to be a special case. If being a special case had any effect, it was simply to increase the similarity of the auxiliary figures to the main problem by common elements, to make its introduction easy. Observe also that the generation of the auxiliary problem in the Two-ships problem was likewise a special case; but that the solution to the connection problem was quite different.

4.3.4. Using a method: The Homogeneous-tetrahedron problem.

The problem of finding the center of gravity (CG) of a homogeneous tetrahedron (Polya, 1945 p37) was already mentioned. This is a problem in solid (three dimensional) geometry. An analogous problem can be set up in plane (two dimensional) geometry, namely, to find the center of gravity of a homogeneous triangle. This is the auxiliary problem and it is easier to solve. What is taken over from it to the original problem is the specific method that was developed for its solution.

---

[13] Actually, the triangle need not be used to solve the main problem given the parallelogram; it is enough to mark off c along the upper a line, to locate the upper right vertex and then d is simply drawn in. However, this is not the way discovery would tend to occur.

[14] It is one more piece of poetry in Polya's problem solving that both auxiliary problems come out of the same approach.

Figure 4-4 shows the relationship between the two methods (it is not a trace of the solution attempt). The *CG* for a triangle is solved (on the left) by decomposing it into strips, for which finding the *CG* is trivial; and then using (implicitly) the method of loci to locate the *CG* as the intersection of the medians. As can be seen (on the right), an identical proof schema holds for the three dimensional case. Some steps are actually identical in both proofs; others are the direct analog. Both proofs rest on corresponding theorems about the coincidence of multiple intersecting elements, the three median lines in the two dimensional case and the six median planes in the three dimensional case.

Auxiliary Problem	Main Problem
Find CG of triangle	Find CG of tetrahedron
View triangle as fibers parallel to AB	View tetrahedron as fibers parallel to edge AB
CG of a fiber = Midpoint of fiber	CG of a fiber = Midpoint of fiber
Therefore CG on locus of midpoints	Therefore CG on locus of midpoints
Locus is a line	Locus is a plane
By symmetry: CG on all 3 lines	By symmetry: CG on all 6 planes
CG is intersection	CG is intersection
Check: Median lines intersect in a point	Check: Median planes intersect in a point

Figure 4-4: Using the method of an auxiliary problem:
Finding the center of gravity (CG) of a homogeneous tetrahedron.

Using an auxiliary problem as a method, unlike using its result, does determine a connection, namely, the auxiliary problem is to be a plan for the main problem. Plan interpreting machinery occurs in systems such as NOAH and MOLGEN. Thus, this variant of auxiliary problems is not as wild. However, domestication does not last for long. The nature of the interpretation loop is to take a step in the plan and then set up a subgoal of finding the analog step in the solution sequence of the main problem to the step in the plan. In programs such as NOAH and MOLGEN, matters are carefully arranged so that there is nothing wild about such subproblems. However, in the case of using the method, such subproblems involve typical connection subproblems. The only thing known is that the step is an *analog*, which may be related to the analogs used for other steps in the proof (if they have been found already). Only in rare cases, such as the one Polya presents, are matters so neat that the relation is either identity or the exact analog under the original mapping (actually, under its inverse).

Unlike the connection problem for results, a few efforts in AI are relevant to analogies of method, growing

out of an independent interest in analogies.[15] Of especial interest is a program called ZORBA (Kling, 1971), which found proofs in a resolution theorem proving system by drawing analogies between the predicates involved. The analogy operated as an attention focusing device, without any attempt to map over the structure of the proof or to pose any of the connection subproblems. The interest is not so much in the program's success (though it had some), but in its being the final outcome of an effort to use Polya's heuristic generally -- exactly the effort we claimed had not occurred successfully. In fact, Kling's earlier efforts to use analogies in the full sense sketched out here (the auxiliary problem as proof plan with the connection problems set up for each step), failed. Only when the role of analogy was used in this weaker form was success attained.

A recent program, called ANA (McDermott, 1979), does use one method as an analogy for another (eg, painting objects for washing objects). This is certainly a step in the right direction. But the situations are quite simple, so that the connection subproblems are limited to the finding of objects in the environment to substitute for other objects. The object-centered character of ANA limits its interest from the present viewpoint.

4.3.5. Can AI deal with wild subproblems?

The failure of AI to exploit Polya's heuristic can now be linked to its inability to deal with wild subproblems. In turn, the key issue for the latter are connection subproblems. To consider the example from the Two-ships problem to explore the nature of the connection subproblem. It can be stated as follows:

Given:	1. The Two-ships problem.
	2. The One-ship auxiliary problem, with solution.
To Find:	An approach to solving the Two-ships problem using the solution to the One-ship problem.

The initial situation is simply the two problems given in isolation (one with solution attached). The final desired situation is less well defined. The ultimate result of course is that the main problem be solved. But the result of the connection subproblem is at some intermediate point -- where it is clear what sort of thing is to be tried, but before it is clear whether the details will work out. This has been rendered as finding *an approach*, but this is just another term for this (still unspecified) intermediate stage. One possibility for it is a new problem whose solution is the solution of the main problem, and which was built using knowledge from the solution of the auxiliary problem. This provides a well defined intermediate state that leaves all of the detail to later, and yet has manufactured a connection of the auxiliary problem to the main problem.

As already noted, except for the simple trichotomy of using the result, method or approach, Polya provides

---

[15] Most analogy programs deal with sufficiently remote domains not to be easily related to the problem here (Evans, 1968, Moore & Newell, 1974, Winston, 1979).

all his advice about using auxiliary problems by means of working concrete problems. This contrasts with finding auxiliary problems, where he provides a rich collection of approaches. However, Polya's actual solutions to using auxiliary problems make use of only a modest number of basic approaches. These provide the same sort of guidance as the suggestions for specializing, generalizing, etc. Here are several of them:

1. Find result of the auxiliary problem already in the existing main problem.

2. Introduce result of auxiliary problem to be a component of main problem.

3. Reduce main problem to auxiliary problem.

4. Use solution method of auxiliary problem as solution plan for main problem.

We have repeated the use-as-method suggestion, since it appears to be exactly at the same level of specificity as all the other suggestions. Thus, just like with finding an auxiliary problem, the initial high-level step in solving a connection subproblem can be taken to be one of these high level functional suggestions. There are few enough of these basic ideas available so that one after another can be tried.

In the case of the Two-ships connection problem, the high-level functional idea is that of reduction. A plausible trace is given in Figure 4-5.

```
Connection subproblem: Use One-ship problem to solve Two-ships problem
    Approach: Reduce Two-ships problem to One-ship problem
        Means-ends analysis method
            Find the difference between Two-ship & One-ship problem
                Succeed: Difference in velocity of ship B
            Find what reduces difference in velocity
                Succeed: Another velocity
            Apply reducing velocity to ship B
                Fail: Cannot arbitrarily apply a velocity to a ship
                Find what permits applying a velocity
                    Succeed: Can apply arbitrary velocity if applied to
                             everything
                Apply reducing velocity, but to everything
                    Fail: Exact velocity is not specified
                    Find exact reducing velocity
                        Succeed: The amount to make B have zero velocity
                    Apply opposite velocity of B to everything (A and B)
                        Fail: Must construct velocity in diagram
            Succeed (connection subproblem): New problem that solves
            main problem:
                Construct the velocity of A when a velocity opposite to
                B is applied to it
```

Figure 4-5: Solution of Connection subproblem of Two-Ships problem.

This trace follows an apparently typical GPS-like means-ends analysis path (Newell & Simon, 1972). Though the details of representation and prior knowledge have not been specified well enough to characterize the gaps in each of the steps, they do not seem very large. The key step occurs right at the beginning, in taking the difference and using the result (velocity) to construct a problem space (with velocity operators) within which the rest of the problem solving can take place.

The reader may not see this solution as the natural course of events, having simply recognized the relevance of the concept of *relativity of velocity*, with the consequent imposition of this entire framework within which to solve the problem. The trace in Figure 4-5 shows only that, when the problem is approached from a less insightful (ie, recognitional) standpoint, cues arise which point toward the solution.

The successful outcome of the connection subproblem is the formulation of a well-defined problem whose solution will yield the solution of the main problem and which makes use of the auxiliary problem. This is the problem of constructing in the diagram the equivalent reduced problem, ie, applying the parallelogram of velocities. This may be an easy or hard problem for the problem solver. Conceivably (though not in this case), it might be impossible, for there could be a flaw in the connection problem. What the connection problem has done is to convert the general notion of *reduce the main problem to the auxiliary problem* to a specific well-defined problem.

A similar plausible development can be construed for the other case, constructing the trapezoid. The a priori approach here is to *introduce the result as a component.* Again, the main method is a GPS-like means-ends analysis. The problem space involved is already available, namely constructions on the originally given trapezoid. We leave the details as an exercise for the reader. The result of the connection problem (in the case where it is successful) is a new problem in the same problem space as the main problem, with the same desired state, but starting from the augmented figure. How to proceed to solve this augmented problem will still be open, for it is not part of the connection subproblem. This approach to connecting is less specific than reduction, but it is correspondingly easier to set up and has more chances of being successful.

These two examples raise the possibility that taming wild subproblems may not be out of sight for AI. The methods involved do not seem too different from typical AI methods. To be sure, there is the problem of constructing the problem spaces out of the given information of the connection subproblem (the two problems) plus the functional orientation (reduce main problem, introduce result, ...). But a lot of structure is available in the givens, so this may not be a major difficulty.

Where then is the major difficulty? Perhaps it does not exist. Rather, AI simply has not yet tried to do this task. When it does try, the task will be found tractable. This conjecture should not be discarded lightly. Some

(by no means all) tasks tackled by AI are revealed to be tractable the first time they are tried. Examples include the chemical structure identification task done by DENDRAL (Lindsay, Buchanan, Feigenbaum & Lederberg, 1980), the medical diagnostic task done by MYCIN (Shortliffe, 1976) and the protocol analysis task done by PAS-I (Waterman & Newell, 1971). These tasks all embodied extensions to AI programs into new domains and some of the systems represent important advances in AI. The techniques and understanding available at the time simply proved sufficient for the new area.

5. Conclusion

It has always been clear why reverence has been due Polya in AI. Just at the time digital computers came into existence, to provide the technical and conceptual underpinning to explore the information processing underlying intelligent behavior, he focussed attention on heuristic. His books -- delightful, provocative and technically astute -- helped to provide the Zeitgeist[16] for looking at approximate reasoning and inference, and how they could be accomplished by computers.

The question is why Polya has been ignored by AI. Why has his work been limited to being part of the groundwork, rather than becoming an integral part of the intricate figurework it has been building? The aim of the paper has been to shed light on this question.

5.1. Review

Let us retrace the path we have followed in this paper. We started by examining the major outlines of Polya's work on heuristic: the implicit model of the human as a problem solver; the role of AI-like methods; the role of auxiliary problems; the focus on the future by learning from the present problem; the analysis of the credibility of inductions and analogies; and, last, the scattered heuristics that are not embedded systematically in the main picture of how problem solving should proceed. This picture reflected already an AI viewpoint. If we had presented Polya's work in its own terms, the same elements would have been present, but the properties of the human problem solver and the AI-like methods would have been implicit and much more attention would have been given to analogy and induction.

The review of Polya's heuristic led to six theses to explain why AI did not take him more seriously:

1. Polya's heuristic is not relevant to the real tasks of AI.

2. Human psychology, as woven into Polya's heuristic, is too different from the character of current computers.

3. Polya's methods are either already known and used, or they are inapplicable to machine use.

4. The action is outside Polya's heuristic, ie, the important knowledge for solving problems lies in the content of the problems themselves, not in the general rules or guidelines that can be transferred from Polya's work to AI programs.

5. The future orientation of Polya's heuristic implies substantial learning abilities, which is beyond

---

[16] So much happened in the post World War II years -- the computer, operations research, game theory, decision theory, control theory, information theory -- that it is hard not to think of them all as related. Yet, the work of Polya on heuristic bears no apparent mark of these events. It is grounded in a deep concern for classical mathematics and a reverence for the great mathematicians of the past, eg, Descartes and Liebnitz. Only Polya himself can assess the role of the larger intellectual currents in bringing forth his own work at a particular historical moment.

AI's current capability.

6. The diversity of the auxiliary problems, which play such a key role in Polya, are beyond AI's current capabilities.

The first three theses we dismissed after brief examination, although the first one remained the default thesis. The last three all had to be taken seriously. One of them (the fourth) is a difficulty with Polya; the other two (the fifth and sixth) are difficulties with AI. Though they all deserve serious investigation, only one could be followed in this paper. We elected to examine the issues of auxiliary problems, giving some reasons why this was the appropriate choice.

Out of the study of this sixth thesis has come a notion of auxiliary problem that distinguishes it from the sort of subproblems that are generated by methods typical in AI. The main feature of the latter is that the relation of the subproblem to the main problem is defined in advance by construction. Contrariwise, with auxiliary problems this relation is problematical. The former might be thought of as *tame* or *domesticated* subproblems, the latter as *wild* subproblems.

Given this distinction, what we found can be summarized directly:

1. Current AI programs have the ability to deal with tame subproblems, but not wild ones.

2. This shows at least one important way in which AI's own limitations have prevented it from making extensive and detailed use of Polya's heuristic.

3. The space of auxiliary problems does not form a well-defined problem space. The generators of auxiliary problems bear at best a weak relationship to the structure of the main problem.

4. Taking the place of the space of auxiliary problems are the problem spaces arising in making auxiliary problems relevant -- the connection problems. A problem solver must have the capability to deal with these connection subproblems to be able to use auxiliary problems.

5. The character of connection subproblems is essentially unexplored. However, there is no indication yet that they are necessarily especially difficult.

5.2. What Finally Follows?

The present paper is hardly a thorough investigation, even into its chosen questions. Yet, though it leaves much unfinished, it does not leave matters where they stood at the beginning. There are by now four independent questions. Each rates a separate concluding statement.

Why has AI ignored Polya? We have at least one plausible answer: Because AI doesn't yet handle wild subproblems. However, the other two theses -- the intrinsic power of Polya's heuristic knowledge and the role

of learning -- are at least as important to pursue. The present investigation helps set the stage for both, by explicating how problems are solved in Polya. A major component of that substance is the nature of auxiliary problems.

What is the nature of the connection subproblem? This seems to be the really interesting conceptual question posed by the present analysis. We were led to the question, but we hardly began its analysis. Connection subproblems clearly have some common structure and their own collection of methods. It would be extremely rewarding to carry out a thorough investigation of this. Polya provides a wealth of examples still to be understood.

Can AI use auxiliary problems? The present paper provides more than adequate initial structure to pose this question. My tentative answer is a simple yes -- that the reason why it hasn't happened, is basically that AI hasn't gotten around to it. However, this question can only be answered in the classical way, by producing AI programs that can solve significant problems by using auxiliary problems that pose significant connection subproblems. Moreover, my answer cannot be entirely correct, because accomplishing this will clearly add significantly to the power and flexibility of AI systems.

What of Polya's student? The outermost story, as you will recall from the opening section of this paper, concerns my own wonderment that I could have been so thoroughly immersed in Polya in my early college years and yet have his heuristic play such a small *explicit* role in my own work on heuristic. This study, of course, was not designed to shed light directly on why I did what I did. Instead, it does something substantially more important to me. It shows that there is in Polya's heuristic a wealth of knowledge that is pertinent to AI, and which AI would do well to take seriously. It goes some small way, I hope, to acknowledge Polya as a genuine contributor to AI.

6. References

Adams, J. L. *Conceptual Blockbusting.* San Francisco: Freeman 1974.

Anderson, J. R. *Language, Memory and Thought.* Hillsdale, NJ: Lawerence Erlbaum 1976.

Bentley, J. L. *Divide and Conquer Algorithms for Closest Point Problems in Multidimensional Spaces.* PhD thesis, University of North Carolina at Chapel Hill, 1976.

Carbonell, J. *POLITICS.* PhD thesis, Computer Science Department, Yale University, 1978.

Cohen, D. *Knowledge Based Theorem Proving and Learning.* PhD thesis, Computer Science Department, Carnegie-Mellon University, 1980.

De Bono, E. *New Think.* New York: Basic Books 1968.

Evans, T. G. A program for the solution of a class of geometric-analogy intelligence-test questions. In Minsky, M. (Ed.), *Semantic Information Processing,* Cambridge, MA: MIT Press, 1968.

Feigenbaum, E. A. & Feldman, J. (Eds.). *Computers and Thought.* New York: McGraw-Hill 1963.

Gelernter, H. Realization of a geometry-theorem proving machine. In Feigenbaum, E. A. & Feldman, J. (Ed.), *Computer and Thought,* New York: McGraw-Hill, 1963.

Green, C. & Barstow, D. On program synthesis knowledge. *Artificial Intelligence,* 1978, *10,* 241-279.

Hunt, E. *Artificial Intelligence.* New York: Academic Press 1975.

Kling, R. E. A paradigm for reasoning by analogy. *Artificial Intelligence,* 1971, *2,* 147-178.

Lenat, D. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search.* PhD thesis, Computer Science Department, Stanford University, 1976.

Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A. & Lederberg, J. *Applications of Artificial Intelligence to Chemistry: The DENDRAL project.* New York: McGraw-Hill 1980.

McDermott, J. Learning to use analogies. *Proceedings of the International Joint Conference on Artificial Intelligence,* 1979, *6,* 568-576.

Minsky, M. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers,* 1961, *49,* 8-30. (Reprinted in Feigenbaum, E. & Feldman, J. (Eds.) *Computers and Thought,* New York: McGraw-Hill, 1963).

Moore, J. & Newell, A. How can Merlin understand?. In Gregg, L. (Ed.), *Knowledge and Cognition,* Potomac, MD: Lawrence Erlbaum Associates, 1974.

Newell, A. & Simon, H. A. *Human Problem Solving.* Englewood Cliffs: Prentice-Hall 1972.

Newell, A. & Simon, H. A. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM,* 1976, *19(3),* 113-126.

Newell, A., Shaw, J. C. & Simon, H. A. Empirical explorations of the Logic Theory Machine: A case study in heuristics. In *Proceedings of the 1957 Western Joint Computer Conference,* : Western Joint Computer

Conference. 1957. (Reprinted in Feigenbaum, E. & Feldman, J. (Eds.) *Computers and Thought*, New York: McGraw-Hill, 1963).

Newell, A., Shaw, J. C., & Simon, H. A. The processes of creative thinking. In Gruber, H. E., Terrell, G., & Wertheimer, J. (Ed.), *Contemporary Approaches to Creative Thinking*, : Atherton, 1962.

Newell, A. Heuristic programming: Ill-structured problems. In Aronofsky, J. (Ed.), *Progress in Operations Research, III*, New York: Wiley, 1969.

Newell, A. Production systems: Models of control structures. In Chase, W. C. (Ed.), *Visual Information Processing*, New York: Academic Press, 1973.

Newell, A. Reasoning, problem solving and decision processes: The problem space as a fundamental category. In R. Nickerson (Ed.), *Attention and Performance VIII*, Hillsdale, NJ: Erlbaum, 1980.

Nilsson, N. *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga 1980.

Pfefferkorn, C. *The Design Problem Solver.* PhD thesis, Computer Science Department, Carnegie-Mellon University, 1968.

Polya, G. & Szego, G. *Problems and Theorems in Analysis, 2 vols.* New York: Springer-Verlag 1972. (Translation and expansion of *Aufgaben und Lehrsatze aus der Analysis*, 1924).

Polya, G. *How to Solve It.* Princeton, NJ: Princeton University Press 1945.

Polya, G. *Mathematics and Plausible Reasoning, 2 vols.* Princeton, NJ: Princeton University Press 1954.

Polya, G. *Mathematical Discovery, vol 1.* New York: Wiley 1962.

Polya, G. *Mathematical Discovery, vol 2.* New York: Wiley 1965.

Polya, G. *Mathematical Methods in Science.* Washington, DC: Mathematical Association of America 1977. (First edition, 1963).

Rosenblatt, F. *Principles of Neurodynamics.* Technical Report, Cornell Aeronautical Laboratory, 1961.

Sacerdoti, E. D. *A Structure for Plans and Behavior.* New York: Elsevier 1977.

Samuel, A. Some studies in machine learning using the game of checkers. In Feigenbaum, E. A. & Feldman, J. (Eds.), *Computers and Thought*, New York: McGraw-Hill, 1963.

Savage, L. J. *The Foundations of Statistics.* New York: Wiley 1954.

Schank, R. & Ableson, R. *Scripts, Plans, Goals and Understanding.* Hillsdale, NJ: Lawrence Erlbaum 1977.

Selfridge, O. Recognition ... In Cherry, C. (Ed.), *Information Theory*, London: Chapman, 1959.

Shortliffe, E. H. *Computer-based Medical Consultations: MYCIN.* New York: American Elsevier 1976.

Simon, H. A. & Kotovsky, K. Human acquisition of concepts for sequential patterns. *Psychological Review*, 1963, *70*, 534-546.

Slagle, J. A heuristic program that solves symbolic integration problem in freshman calculus. In Feigenbaum,

E. A. & Feldman, J. (Eds.), *Computers and Thought*, New York: McGraw-Hill, 1963.

Stefik, M. *Planning with Constraints.* PhD thesis, Computer Science Department, Stanford University, 1980.

Sussman, G. J. Electrical design: A problem for artificial intelligence research. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1977, *5*, 894-900.

Tulving, E. & Thomson, D. M. Encoding specificity and retrieval processes in episodic memory. *Psychological Review*, 1973, *80*, 352-373.

Waterman, D. & Newell, A. Protocol analysis as a task for artficial intelligence. *Artificial Intelligence*, 1971, *2*, 285-318.

Winograd. *Understanding Natural Language.* New York: Academic Press 1972.

Winston, P. H. Learning structural descriptions from examples. In Winston, P. H. (Ed.), *The Psychology of Computer Vision*, New York: McGraw-Hill, 1975.

Winston, P. *Artificial Intelligence.* Reading, MA: Addison-Wesley 1977.

Winston, P. *Learning by Understanding Analogies.* Technical Report Memo 520, MIT Artificial Intelligence Laboratory, 1979.