CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

Canada

National Library of Canada
Canadian Theses Division
Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Division des thèses canadiennes

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

MARC HECTOR JOSEPH ROMANYCIA

Date of Birth — Date de naissance

October 19, 1956

Country of Birth — Lieu de naissance

CANADA

Permanent Address — Résidence fixe

2841 - 24th St. N.W
CALGARY, ALBERTA
T2M 3Z4

Title of Thesis — Titre de la thèse

~~AN ANALYSIS OF THE~~ THE CONCEPT OF HEURISTIC AS USED ~~IN~~ IN THE ARTIFICIAL INTELLIGENCE COMMUNITY

University — Université

THE UNIVERSITY OF ALBERTA

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

MASTER OF ARTS

Year this degree conferred — Année d'obtention de ce grade

1983

Name of Supervisor — Nom du directeur de thèse

DR. F.J. PELLETIER

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date

September 16, 1983

Signature

Marc Romanycia

THE UNIVERSITY OF ALBERTA

THE CONCEPT OF HEURISTIC AS USED IN THE ARTIFICIAL INTELLIGENCE COMMUNITY

by

MARC HECTOR JOSEPH ROMANYCIA

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF ARTS

DEPARTMENT OF PHILOSOPHY

EDMONTON, ALBERTA

FALL, 1983

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR Marc Hector Joseph Romanycia

TITLE OF THESIS THE CONCEPT OF HEURISTIC AS USED IN THE ARTIFICIAL INTELLIGENCE COMMUNITY

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF ARTS

YEAR THIS DEGREE GRANTED Fall 1983

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Marc Romanycia

PERMANENT ADDRESS:
2841 - 24 Street N.W.
CALGARY, ALBERTA
T2M 3Z4

DATED October 17, 1983

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled

THE CONCEPT OF HEURISTIC AS USED IN THE ARTIFICIAL INTELLIGENCE COMMUNITY

submitted by Marc Hector Joseph Romanycia in partial fulfillment of the requirements for the degree of Master of Arts.

.....
Supervisor

.....

.....

.....

Date October 17, 1983

ABSTRACT

The concept of heuristic has always played an important role in Artificial Intelligence. However, since the birth of this discipline in the mid 1950's to the present, various differing definitions have been given and various conflicting statements have been made concerning the concept of heuristic. During this period the concept also appears to have evolved in meaning somewhat.

In Chapter One I define the purpose of this thesis. I remark that to date no detailed analysis of the concept of heuristic has been given, and that topics of traditional interest to philosophers like the theory of creativity, logics of discovery, and problem-solving methodology stand to gain from such an analysis, not to mention AI research itself. Within the bounds of this thesis, however, the goals of the analysis are solely the provision of a well substantiated definition and, secondarily, the resolution of certain key questions regarding heuristics, such as whether or not heuristics are opposed to algorithms and whether heuristics need to save effort.

In Chapter Two I prepare for the Analyses that I give in Chapter Three. I do this in two ways. First, I run through the history of the concept of heuristic in AI, quoting and commenting on many of the definitions that various authors have given. I highlight all the properties that are ascribed to heuristics, whether or not these are compatible. Second, I develop in greater detail the concepts of heuristic that have been used in one area of AI, Symbolic Logic Theorem Proving.

In Chapter Three I give my detailed analysis of the concept of heuristic. I organize the chapter around four sections, each of which deals with one category of properties that were highlighted in Chapter Two. Section A covers the fallibility properties: lack of solution guarantee, incompleteness, imprecision, and opposition to algorithms. Section B covers the plausibility property which arises from partial knowledge about the problem domain. Section C covers the performance properties: search reduction, effort reduction, efficiency increase, practicality, and empirical orientation. Section D, the last section, covers the properties associated with the search-directing function of heuristic devices: their characterization as rules, procedures, methods, strategies, filters, etc.

In Chapter Four, the concluding chapter, I summarize my results and bring up three topics for future enquiry: the possibility of describing all human decision making in terms of heuristics; the specifications for and design of the so-called meta-heuristics, those heuristics that operate on other heuristics; and the forms of interaction that are possible between the heuristics in a complex problem-solving system, such as cooperation, competition, and dependence.

ACKNOWLEDGEMENTS

I wish to thank my principal thesis supervisor, Prof. F. Jeffry Pelletier of the Philosophy Department at the University of Alberta, for his generous investment of time and effort both in direct supervisory work on the thesis, and in the background work of organizing the defense, scheduling supervisory meetings, etc. Prof. Pelletier's extensive knowledge of the Theorem Proving literature was invaluable to my work in Chapter II.

I am also very grateful to the remaining members of my supervisory committee, Prof. Bruce Hunter and Prof. Bernard Linsky, both of the Philosophy Department at the University of Alberta, who with their insightful comments and constructive criticism helped keep this thesis within the bounds of Philosophy and comprehensible to a philosopher.

Prof. Jeffrey Sampson of the Department of Computing Science at the University of Alberta acted as my external examiner and I wish to thank him for having diligently read, corrected, and commented upon my thesis. Prof. Sampson is very much responsible for having stimulated my interest in AI during my undergraduate years.

My fellow employees at GULF CANADA LIMITED have all given me encouragement, not to mention the opportunity to explain my ideas to the uninitiated.

I am also grateful to GULF CANADA LIMITED – Information Systems for allowing me to use their computing facilities for the preparation of this thesis, and for having granted me a leave of

absence to complete this thesis.

Finally, I wish to thank the members of my family for their numerous support activities.

TABLE OF CONTENTS

THE CONCEPT OF HEURISTIC AS USED IN THE ARTIFICIAL INTELLIGENCE COMMUNITY

I. Introduction

Within the past few decades the notion of heuristic or rule of thumb has been called upon to play a central role in theories of human problem-solving and discovery. Attempts to systematize human problem-solving behavior have resulted in recognition of the existence of repeated patterns of rule applications which do not guarantee success but which are often observed to be useful in arriving at a solution. In particular, wherever the task domain is sufficiently difficult to exclude the possibility of applying efficient algorithms, but not so difficult as to allow only a random search for the solution, heuristics can often be found to help improve the efficiency of search.

Likewise, some theorists of the creative process argue that although there may be no "logic of discovery" which guarantees the discovery of all knowledge, discovery need not proceed from mere intuition and guesswork. Rather, there appear to be numerous valuable heuristics which aid in discovery.

Indeed, in so far as all human behavior can be thought of as problem-solving behavior, the problem being one of living wisely, achieving assorted long and short term goals, etc., the potential domain of heuristics can be likewise extended. The concept of heuristic rule is presently being submitted to such a broader

application in a number of computer models of human behavior. These range from models of mathematical theorem generation, speech understanding, and inductive learning of relations, to the detection of conversational patterns in dialogues.[1]

In very few of the research efforts which employ heuristics is the concept of heuristic given any detailed analysis in its own right.[2] Individual heuristics are discovered, tested, and modified in conjunction with a particular task or subtask, but the concept of heuristic itself is rarely reflected upon. Definition by example is as a rule the primary method of introducing the concept to the newcomer.[3] If definitions are given they are short and unsatisfying; one feels aspects of the concept have been overlooked. Is it that Artificial Intelligence (AI) researchers think the concept is simple, or is it that they feel a detailed study would contribute little compared to actually discovering and applying individual heuristics?

It is one of the traditional jobs of philosophers to apply their analytic tools to discovering the deeper meanings of important concepts in other disciplines. Such extradisciplinary work often results in the shedding of new light on traditional

---

[1] A sampling of research in each of these areas can be found in Waterman and Hayes-Roth (1978).

[2] The only noteworthy exception to this claim is the recent work by Lenat on the nature of heuristics: Lenat (1982, 1983a, 1983b). Lenat's analysis, however is not philosophic in character, it is not a careful exposition of key concepts; rather it appears to be a variegated mixture of hypothetical key ideas and speculations presented as an account of his and his colleagues latest reflections on the subject. I do not mean this as a criticism. Such work is very valuable in forefront scientific research.

[3] Barr and Feigenbaum similarly remark on the general lack of definitions. See Barr and Feigenbaum (1981), p.28.

philosophic problems. It may also have practical influence in those outside disciplines themselves. Just as philosophic analyses of the physicists' concepts of Space, Time, and Causality have both enriched the traditional philosophic ideas about these topics, as well as contributed to the theories of relativity and quantum mechanics, I believe a philosophic analysis of the concept of 'heuristic' could lead, on the philosophy side, to new or enhanced theories of problem-solving, discovery, scientific methodology, and perhaps even philosophic methodology, and on the AI side, to a more complete account of how heuristics interact, of where they can be found, of how they can be built, etc. I will not here, however, try to establish any new philosophic theories, or do anything more than give guidelines for how AI can improve its use of heuristics. In this thesis I will be content to lay the groundwork for such new theories and practical consequences by analyzing 'heuristic' in more detail than I believe has ever been done before. I believe both philosophers and AI researchers will find the discussions interesting. The final product of these discussions will be a definition. But in addition to providing a substantial definition it will be my purpose to answer some of the intriguing questions that have arisen concerning heuristics: are heuristics and algorithms opposed? Must a heuristic save effort? Is there just one concept underlying 'heuristic' or is 'heuristic' just a name given to a mix of several distinct ideas?

To achieve all this I have chosen the following procedure. First we will look at the history of AI and select instances of how 'heuristic' has been defined and how it has been used. This

chapter will serve as an introduction to the concept and to all the properties which have at some time been associated with it. Then I will carry out my philosophical analysis by discussing in detail each of these main properties. Finally I will summarize my results and indicate possible future lines of enquiry.

II. History and Usage

A. Definitions

> heuriskein (ancient Greek) and heuristicus (Latin): "to find out, discover".[4]
>
> Heuretic: The branch of Logic which treats of the art of discovery or invention. 1838 Sir W. Hamilton Logic App. (1866) II. 230 That which treats of these conditions of knowledge which lie in the nature, not of the thought itself, but of that which we think about .. has been called Heuretic, in so far as it expounds the rules of Invention or Discovery. Heuristic: Serving to find out or discover. 1860 Whewell in Todhunter's Acc. W.'s Wks. (1876) II. 418 If you will not let me treat the Art of Discovery as a kind of Logic, I must take a new name for it, Heuristic, for example. 1877 E. Caird Philos. Kant II xix. 662 The ideas of reason are heuristic not ostensive: they enable us to ask a question, not to give the answer.[5]

The earliest reference that I could find in the AI literature for the term 'heuristic' is to George Polya (1945).[6] Allen Newell, the noted heuristics researcher and a student of Polya's, says "Polya ... is recognized in AI as the person who put heuristic back on the map of intellectual concerns."[7] Polya himself claims credit for reintroducing this somewhat forgotten term:

> Heuristic, or heuretic, or "ars inveniendi" was the name of a certain branch of study, not very clearly circumscribed, belonging to logic, or to philosophy, or to psychology, often outlined, seldom presented in detail, and as good as forgotten

---

[4] Random House Dictionary, 1972 edition.
[5] The Oxford Dictionary of the English Language, 1933 edition.
[6] See Marvin Minsky's extensive 1961 subject bibliography for AI in Minsky(1961b).
[7] Newell(1980), p.1.

> today. The aim of heuristic is to study the methods and rules of discovery and invention. A few traces of such study may be found in the commentators of Euclid; a passage of PAPPUS is particularly interesting in this respect. The most famous attempts to build up a system of heuristic are due to DESCARTES and to LEIBNITZ, both great mathematicians and philosophers. Bernard BOLZANO presented a notable detailed account of heuristic. The present booklet is an attempt to revive heuristic in a modern and modest form. See MODERN HEURISTIC.
>
> Heuristic, as an adjective, means "serving to discover."[8]

It would take me too far astray to adequately discuss the ideas Pappus, Descartes, Leibniz, or Bolzano had on heuristic. Descartes, however, can by way of contrast help us clarify Polya's conception of heuristic. When Polya speaks of Descartes' heuristic he has in mind the Rules for the Direction of the Mind and the Discourse on Method.[9] Although Descartes certainly does provide rules for discovering truths, it is questionable whether he intended his rules to be merely plausible and helpful. There does appear to be an element of certainty and infallibility in his intentions.

> ... seeking the true Method of arriving at a knowledge of all the things of which my mind was capable.[10]

> [by applying his method] there can be nothing so remote that we cannot reach it, nor so recondite that we cannot discover it.[11]

---

[8]Polya(1945), p.112-3. This passage is an entry in "The Short Dictionary of Heuristic", Part III of How To Solve It. Polya capitalizes all the words or phrases that are separate entries in his dictionary.

[9]Polya(1945), p.92-3.

[10]Descartes, The Discourse on Method, p.91.

[11]Ibid., p.92.

> For, in conclusion, the Method which teaches us to follow the true order and enumerate exactly every term in the matter under investigation contains everything which gives certainty to the rules of Arithmetic.
>
> But what pleased me most in this Method was that I was certain by its means of exercising my reason in all things, if not perfectly, at least as well as was in my my power.[12]

Polya, however, is strictly opposed to such a position:

> Heuristic reasoning is reasoning not regarded as final and strict but as provisional and plausible only, whose purpose is to discover the solution of the present problem. We are often obliged to use heuristic reasoning. We shall attain complete certainty when we shall have obtained the complete solution, but before obtaining certainty we must often be satisfied with a more or less plausible guess. We may need the provisional before we attain the final. We need heuristic reasoning when we construct a strict proof as we need scaffolding when we erect a building. ... Heuristic reasoning is often based on induction, or on analogy;[13]

In particular we note that heuristic reasoning is contrasted with the deductive type of reasoning in providing proofs.

Elsewhere he writes:

> Provisional, merely plausible HEURISTIC REASONING is important in discovering the solution, but you should not take it for a proof; you must guess, but also EXAMINE YOUR GUESS.[14]

One last contrast with Descartes:

> It is also emphasized that infallible RULES OF DISCOVERY are beyond the scope of serious research.[15]

So Polya sees himself as reviving "heuristic", the study of methods and rules of discovery. He wishes to do this in a "modest

---

[12] Ibid., p.94.
[13] Polya(1945), p.112-3.
[14] Ibid., p.132.
[15] Ibid.

and modern form". I guess 'modest' is used to contrast his science with Descartes' and others. To explain his modern version he writes:

> Modern heuristic endeavors to understand the process of solving problems, especially the mental operations typically useful in this process.
>
> ... a list of mental operations typically useful in solving problems [includes] particular questions and suggestions [like:] ... WHAT IS THE UNKNOWN? IS IT POSSIBLE TO SATISFY THE CONDITION? DRAW A FIGURE ... CAN YOU USE THE RESULT? ... "Go back to definitions" ... COULD YOU RESTATE THE PROBLEM?[16]

> Heuristic discusses human behavior in the face of problems; this has been in fashion, presumably, since the beginning of human society, and the quintessence of such ancient discussions seems to be preserved in the WISDOM OF PROVERBS.[17]

> Heuristic aims at generality, at the study of procedures which are independent of the subject-matter and apply to all sorts of problems. The present exposition, however, quotes almost exclusively elementary mathematical problems as examples.[18]

Hence to paraphrase Polya, Heuristic is a science of problem-solving behavior that focuses on plausible, provisional, useful, but fallible, mental operations for discovering solutions. Examples of such behavior are the suggestions or questions a problem-solver applies to himself while attempting to discover a solution. Regarding proverbs, lest the impression be given that all proverbs represent condensed heuristic discussions, under the dictionary entry "Wisdom of proverbs" Polya explicitly states that only proverbs about problem-solving are the products of heuristic

---

[16] Ibid., pp.129-31.
[17] Ibid., p.132.
[18] Ibid., p.133-4.

science.[19]

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

I will now give examples of AI definitions from this early period. I have chosen all my examples from Computers and Thought, which is generally considered the most representative anthology of important AI work of this period.

Allen Newell, Cliff Shaw, and Herbert Simon, while writing about their groundbreaking symbolic logic theorem proving program, the Logic Theorist, were the first to use 'heuristic' as a noun meaning heuristic process.[20] They claim to be using 'heuristic' here according to the standard dictionary definition, "serving to discover or find out", but they also strongly oppose its meaning to

---

[19]Examples of such proverbs are:
Who understands ill, answers ill.
Think on the end before you begin.
An oak is not felled at one stroke.
As the wind blows you must set your sails.
Look before you leap.
Step after step the ladder is ascended..
Polya (1945) p.222-5.

[20]Newell, Shaw, and Simon (1957), p. 114, and Newell (1982), p.17.

that of 'algorithm':

> The research reported here is aimed at understanding the complex processes (heuristics) that are effective in problem-solving. Hence, we are not interested in methods that guarantee solutions, but which require vast amounts of computation. Rather, we wish to understand how a mathematician, for example, is able to prove a theorem even though he does not know when he starts how, or if, he is going to succeed.[21]

> One very special and valuable property that a generator of solutions sometimes has is a guarantee that if the problem has a solution, the generator will, sooner or later, produce it. We call a process that has this property for some problem an algorithm for that problem.
>
> A process that may solve a given problem, but offers no guarantees of doing so, is called a heuristic* for that problem.
>
> * As a noun, "heuristic" is rare and generally means the art of discovery. The adjective "heuristic" is defined by Webster as: serving to discover or find out. It is in this sense that it is used in the phrase "heuristic process" or "heuristic method." For conciseness, we will use "heuristic" as a noun synonymous with "heuristic process." No other English word appears to have this meaning.[22]

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

Herbert Gelernter, who devised an early geometry theorem proving program that grew upon the Logic Theorist work, gives a

---

[21] Newell, Shaw, and Simon (1957) p.109.
[22] Ibid., p.114.

definition reminiscent of Polya:

> A heuristic method is a provisional and plausible procedure whose purpose is to discover the solution of a particular problem at hand.[23]

Indeed, after giving this definition he advises his reader to read Polya for a "definitive treatment of heuristics and mathematical discovery". Like Newell, Shaw, and Simon, Gelernter emphasizes avoidance of algorithmic exhaustive search as the rationale for introducing heuristics into a problem situation.

Gelernter is also one of the first to point out that heuristics work in effect by eliminating options from an impractically large set of possibilities:

> A heuristic is, in a very real sense, a filter that is interposed between the solution generator and the solution evaluator for a given class of problems.[24]

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

For Fred Tonge, who worked on a heuristic program for minimizing the number of workers needed on an assembly line, the

---

[23] Gelernter (1959), p.135.
[24] Ibid., p.137.

non-guaranteed element is less important and the filtering element is not present. He emphasizes efficiency and effort reduction in achieving a satisfactory solution. His definition also shows the tendency to abstract the meaning of 'heuristic' away from "process" and towards any arbitrary "device". Often the "device" is a portion of his program with an identifiable function. He also speaks of heuristics as providing "shortcuts", and as employing "simplifications", in contrast with several of the algorithmic methods that theoretically guarantee solutions. His official definition is:

> ... by heuristics we mean ... principles or devices that contribute, on the average, to reduction of search in problem-solving activity. The admonitions "draw a diagram" in geometry, "reduce everything to sines and cosines" in proving trigonometric identities, or "always take a check — it may be a mate" in chess are all familiar heuristics.
>
> Heuristic problem-solving procedures are procedures organized around such effort-saving devices. A heuristic program is the mechanization on a digital computer of some heuristic procedure.[25]

Marvin Minsky was one of the first to use 'heuristic' in the context of "search" through a large "problem space". Speaking of chess, which has been estimated to have 10^{120} paths through its game tree, he says "we need to find techniques through which the results of incomplete analysis can be used to make the search more efficient."[26] His official definition, like Tonge's, emphasizes efficiency rather than an opposition to algorithms:

---

[25] Tonge (1960), p.172.
[26] Minsky (1961a), p.408.

> The adjective "heuristic," as used here and widely in the literature, means related to improving problem-solving performance; as a noun it is also used in regard to any method or trick used to improve the efficiency of a problem-solving system. A "heuristic program" to be considered successful, must work well on a variety of problems, and may often be excused if it fails on some. We often find it worthwhile to introduce a heuristic method which happens to cause occasional failures, if there is an over-all improvement in performance. But imperfect methods are not necessarily heuristic nor vice versa. Hence "heuristic" should not be regarded as opposite to "foolproof"; this has caused some confusion in the literature.[27]

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

James Slagle, who developed a Logic Theorist-like program to solve integration problems in mathematics, uses 'heuristic' primarily to stand for any of a class of rules that transform a problem into one or more subproblems. Examples of such rules would be "try integration by parts" and "try a trigonometric substitution He distinguishes algorithmic and heuristic transformations, the latter being defined as follows:

> A transformation of a goal is called heuristic when, even though it is applicable and plausible, there is a significant risk that it is not the appropriate next step.[28]

---

[27]Ibid., p.408.
[28]Slagle (1963), p.197.

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

> Although many authors have given many definitions, in this discussion a heuristic method (or simply a heuristic) is a method which helps in discovering a problem's solution by making plausible but fallible guesses as to what is the best thing to do next.[29]

We will return to discuss this kind of confusion in the next chapter.

Finally we come to the definition of Feigenbaum and Feldman, the editors of Computers and Thought:

> A heuristic (heuristic rule, heuristic method) is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits search for solutions in large problem spaces. Heuristics do not guarantee optimal solutions; in fact, they do do not guarantee any solution at all; all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time.[30]

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

---

[29] Ibid., p.192.
[30] Feigenbaum and Feldman(1963), p.6.

domains.

It is interesting to stop at this point in our historical survey and observe how the definition of heuristic has been transformed since its original introduction to the AI community via Polya. Polya used 'heuristic' primarily in the context of logic or psychology of discovery. His heuristic methods were to apply helpful reasoning processes like asking certain questions, drawing diagrams, guessing, looking at the problem from a different perspective, etc. Somehow these methods direct the mind towards seeing a solution. 'Discovery' is used here very much in the sense of invention; it presumes a kind of groping exploration prior to the discovery. By the end of this early period in AI, however, 'heuristic' has been reshaped to the AI landscape. Rather than a vague psychological groping for a solution, we were presented with the notion of an exploration guided along paths in a formal problem-solving structure or space. For this reason 'discovery' is used less in the sense of exploring a previously untrodden solution path than in the sense of finding a successful path amongst those already explicitly or implicitly prespecified in the predefined state-space structure.

Another reemphasis is that, rather than having heuristic methods derive from general problem-solving psychology and be made applicable to specific domains like mathematics, in AI we have specific problem domains giving rise to their own brand of heuristic methods. Indeed, in AI the whole driving force for introducing heuristics and discovering new ones is to improve the performance of a program in a particular problem domain. In

contrast, for Polya the reason to introduce heuristics was to have math students learn how to think, i.e., to acquire the type of psychology necessary to do good mathematics.

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

We will now continue the history of the term 'heuristic'.

Following this era of definitions a new usage for 'heuristic' was introduced, as part of the phrase "heuristic search". This usage has become very popular, to the point where some authors do not use 'heuristic' in any other form, eg., Winston (1977), while others prefer to use it over the simple noncomposite form, eg., Barr and Feigenbaum (1981).

According to Newell and Simon, in 1965 Ernst and Newell

introduced the concept of "heuristic search, which itself was simply an attempt to formulate what seemed common to many of the early artificial intelligence programs."[31]

In 1969 Newell and Ernst wrote:

> HEURISTIC SEARCH. This research approaches the construction of a general problem-solver by way of a general paradigm of problem solving: heuristic search (Newell and Ernst, 1965). In simplified form the heuristic-search paradigm posits objects and operators, where an operator can be applied to an object to produce either a new object or a signal that indicates inapplicability.
>
> The operators are rules for generating objects, and thus define a tree of objects. ... A method for solving a heuristic-search problem is searching the tree, defined by the initial situation and the operators, for a path to the desired situation.[32]

As Barr and Feigenbaum remark, 'heuristic' appears to play an odd role here.[33] If heuristic search is just search through a tree then even blind search is a form of heuristic search. Nowadays it is more common to call Ernst and Newell's heuristic search "state-space search" and to reserve 'heuristic search' for search through a state-space that is based on heuristic decision processes. In other words 'heuristic search', as used nowadays, does not involve a totally new usage of 'heuristic'.[34]

From after the time of the early AI literature on heuristic problem-solving to the present, one will find few definitions of 'heuristic' except when authors are writing for a primarily lay

---

[31]Newell and Simon(1972), p.888

[32]Newell and Ernst(1969), pp.247-8.

[33]Barr and Feigenbaum (1981), p.30.

[34]For examples of modern usage of 'heuristic search' see Barr and Feigenbaum (1981), p.28-30, Winston (1977), p.122ff., and Nilsson (1980), p.72.

audience. In these cases the term is typically defined very superficially so as to include all the standard definitions.[35] Samples of these are:

> ... heuristic methods, i.e., features that improve the systems' problem-solving efficiency or range of capability. These range from ad hoc tricks for particular kinds of problems to very general principles of efficient administration and resource allocation.[36]

> A heuristic is a rule of thumb, strategy, method, or trick used to to improve the efficiency of a system which tries to discover the solutions of complex problems.[37]

> ..."heuristic programming" refers to computer programs that employ procedures not necessarily [but possibly] proved to be correct, but which seem to be plausible. Most problems that have been considered by AI researchers are of the sort where no one knows any practical, completely correct procedures to solve them; therefore, a certain amount of proficiency in using hunches and partially verified search procedures is necessary to design programs that can solve them. So, by a heuristic is meant some rule of thumb that usually reduces the work required to obtain a solution to a problem.[38]

> [Guzman's scene analysis program uses] a set of informal reasoning rules (sometimes called heuristics) which were derived by an empirical, experimental method. ... Although the resulting programs might not be explainable in terms of some deep underlying theory, they perform adequately in most situations and therefore in a very practical sense they solve the problem.[39]

> A heuristic is any strategem for improving the performance of an artificial intelligence program. The heuristic programming approach to artificial intelligence is perhaps the

---

[35] Barr and Feigenbaum (1981, p.29) cite the Feigenbaum and Feldman (1963, p.6) and Minsky (1961a, p.408) definitions as standards. These were quoted above on p.14 and p.13, respectively.
[36] Minsky (1968), p.8.
[37] Slagle (1971), p.3.
[38] Jackson (1974), p.95.
[39] Raphael (1976), pp.237-8.

> most popular and productive one today. It contrasts with another major approach, ... [the] simulation of human thought. In this approach the aim is more to understand and use the features of human intelligence than to apply any technique which works.[40]

> A heuristic is a method that directs thinking along the paths most likely to lead to the goal, less promising avenues being left unexplored.[41]

> An important distinction underlying much of t work in AI is that between two types of methods used to solve problems. One method is called algorithmic, the other, heuristic. Algorithms are commonly defined as procedures that guarantee a solution to a given kind of problem; heuristics are sets of empirical rules or strategies that operate, in effect, like a rule of thumb.[42]

> Heuristics, as every AIer knows, are rules of thumb and bits of knowledge, useful (though not guaranteed) for making various selections and evaluations.[43]

But we also find novel interpretations emerging. These appear to be second generation ideas on what heuristics really are. But, unfortunately, these too are never clearly defined and explained. For example, Douglas Hofstadter has a view of heuristic as "compressed experience":

> Of course, rules for the formulation of chess plans will necessarily involve heuristics which are, in some sense, "flattened" versions of looking ahead. That is, the equivalent of many games' experience of looking ahead is "squeezed" into another form which ostensibly doesn't involve looking ahead. In some sense this is a game of words. But if the "flattened" knowledge gives answers more efficiently that the actual look-ahead — even if it occasionally misleads — then something has been gained. [44]

---

[40] Sampson(1976), p.128.
[41] Boden(1977), p.347.
[42] Solso(1979), p.436.
[43] Newell(1980), p.16.
[44] Hofstadter(1979), p.604.

Another example of a cursorily presented novel interpretation comes from James Albus, a noted roboticist:

> Procedures for deciding which search strategies and which evaluation functions to apply in which situations are called heuristics. Heuristics are essentially a set of rules that reside one hierarchical level above the move selection and evaluation functions of the search procedure. A heuristic is a strategy for selecting rules, i.e., a higher level rule for selecting lower level rules.[45]

Our final example is from Douglas Lenat who appears to have a view of heuristics similar to Hofstaedter's:

> Heuristics are compiled hindsight: they are nuggets of wisdom which, if only we'd had them sooner, would have led us to our present state much faster. This means that some of the blind alleys we pursued would have been avoided, and some of the powerful discoveries would have been made sooner.[46]

So far we have just reviewed some of the assorted definitions of heuristic that have appeared over the past forty years. We have seen that different researchers have emphasized different properties and we have been introduced to most of these properties. However, we have not yet seen many actual heuristics, nor have we seen any actual AI problem domains where heuristics are employed. Without this experience we will not be able to fully appreciate the nature of search through a problem space, nor appreciate how heuristics are formulated, expressed, implemented, and evaluated.

---

[45]Albus (1981), p.284. See also pp.222-3.
[46]Lenat (1982), p.223.

Background knowledge of this sort will be necessary for my third chapter where I will perform the detailed analysis that reveals the essential properties of the concept.

Accordingly, I have chosen an AI application area which should meet out needs.[47] Theorem proving spans the history of AI, and heuristics have been discussed in its context from the very earliest. In particular, symbolic logic theorem proving, because it has drawn heavily from the work of philosophical logicians, should be easier to introduce, if not of substantial interest, to my reader.

---

[47]Other areas which have been heavy users of heuristics include Game Playing, Language Comprehension, and Expert Systems.

B. An Area of AI to Observe the Application and Usage of the term 'Heuristic': Symbolic Logic Theorem Proving

i. The Logic Theorist

The very first occurrence of the term 'heuristic' as a noun meaning "heuristic process" occurs in the 1957 paper by Newell, Shaw, and Simon, "Empirical Explorations with the Logic Theory Machine: A Case Study in Heuristics". The Logic Theorist (LT) is a program that attempts to prove theorems in the propositional logic system of Principia Mathematica. Using the five axioms and three rules of inference in Principia[48] LT was able to generate proofs of 38 of the first 52 theorems from chapter two of Principia. Of the others, some it could never prove while others it was not given sufficient space and time to prove. Since it can be shown incapable of ever proving some theorems its procedure is termed "incomplete". A proof procedure is complete if it could eventually prove any theorem given to it. We shall see later that

---

[48] Axioms:
1.2 (p V p) -> p
1.3 p -> (q V p)
1.4 (p V q) -> (q V p)
1.5 (p V (q V r)) -> (q V (p V r))
1.6 (p -> q) -> ((r V p)) -> (r V q))
Rules of Inference:
substitution: any expression may be substituted for all occurrences of any one variable in an expression.
replacement: any one connective in any expression may be replaced by its equivalent formulation; eg., 1.2 can have '->' replaced so as to become ¬(p V p) V p.
detachment: if A and (A -> B) are theorems then so is B.

incompleteness has often been a criterion of whether to use the the adjective 'heuristic' or not. Newell, Shaw, and Simon contrast LT with their version of a "British Museum" algorithm. This algorithm is complete — it does guarantee proving every theorem — but its method requires an inordinate amount of time for all but the simplest theorems.[4]

Heuristics occur in two forms in LT. In the first form they are called "methods". A method is a more or less self-contained procedure responsible for a particular operation at the current stage of the proof. LT uses four methods: the substitution method which searches through the current list of theorems and axioms trying to prove the desired theorem by substituting expressions for variables and replacing sub-expressions with equivalent expressions; the detachment method which, if B is the expression (original theorem or intermediate subproblem) to prove, looks for (A -> B) in the list of theorems and axioms, and if found then sets up A as a new subproblem; and the chaining methods which are like detachment except that if the expression to prove is of the form (A -> C) it searches for either (A -> B) or (B -> C) (forward and backward chaining, respectively) in the current list of axioms and theorems and if either is found it then sets up the corresponding

---

[4]In 1960, Hao Wang in "Towards Mechanical Mathematics" gave a very efficient and complete theorem prover for propositional logic and on its basis discredited LT. However, Minsky (1961a, pp.437-8) defends LT for its contribution to heuristic research and remarks that even Wang's algorithm can be considered heuristic since it is "directed toward the reduction of search effort" and "since practically no one still uses 'heuristic' in a sense opposed to 'algorithmic'". Note how this remark agrees with Minsky's previous definition of heuristic quoted above, p.13.

(B -> C) or (A -> B), respectively, as an additional subproblem.

These methods do not form an effective problem-solving system without being coordinated in operation by an "executive" routine. This is quite a common feature in heuristic programs and presents the problem of distinguishing where the true heuristic power resides. Since we will later want to distinguish executive and non-executive heuristics, it will profit us to study a sample executive routine in depth. The executive routine in LT basically operates as follows. A static list of axioms and theorems is provided. The executive starts with the main problem to prove and if unsuccessful with it goes on to the next subproblem on the list. With each problem/subproblem it tries out all four methods in the order: substitution, detachment, forward chaining, and then backward chaining, unless of course one of the methods succeeds. For each method the entire theorem/axiom list is scanned. The process continues until either a proof is found, there are no more subproblems, the problem list exceeds memory space available, or the time limit for this proof expires. The search is basically breadth-first, that is, all subproblems discovered at the same "level" in the proof are attempted in sequential order before attempting subproblems discovered while working on any prior problem.

Newell, Shaw, and Simon consider their executive routine an algorithm since it guarantees that every subproblem generated will be attempted, provided of course a solution is not found beforehand. They suggest that replacing this algorithm with a more selective heuristic procedure would alleviate the exponential

growth of subproblems that occurs.[50]

The other form of heuristic in LT occurs in a procedure called the "similarity test", a part of the "matching process" which is used by each of the above four heuristic methods. Matching is described as neither algorithmic nor heuristic in the 1957 paper, and I suspect this is because the authors were not sure whether the procedure would find all possible matches. Matching is used when looking for a theorem/axiom on the theorem list to see if one corresponds in form to the current problem or to the desired detachment, backward chain, or what have you. The similarity test is a process of screening out some of the theorems which have a low likelihood of matching, either because they have a different number of distinct variables, of variable places, or of parenthesized levels from the main connective. Again the criterion for its being a heuristic is the lack of guarantee of selecting only those theorems which match, since it filters imperfectly. Surprisingly, performance is not a criterion for the similarity test being a heuristic. Newell, Shaw, and Simon speak of a "good" heuristic as one that reduces effort but they imply that it is possible to find a heuristic where the processing costs of implementing it outweigh any benefit in the search reduction it affords. No doubt, however, the idea behind trying search reduction tricks is to save effort.

It is interesting to compare the two forms of heuristic in LT. The first form posits processes that are coordinated by an

---

[50]They actually did this later and report that a combination of the two heuristics, "eliminate complex problems" and "eliminate apparent unprovables", improved performance by a factor of 2.3. (Newell and Simon (1972), pp.127-8.)

executive, and these processes acquire their effectiveness by being properly coordinated. The executive, however, is just an algorithm, operating in a fashion that guarantees application of every method to every problem and guarantees the use of every theorem. The second type of heuristic operates each time a match is requested; there is no need to coordinate it with any other processes for it to be effective.

This is not the end of the LT story. In 1972, in Human Problem Solving, Newell and Simon discussed LT afresh and, remarkably, their use of 'heuristic' changed substantially. They no longer emphasized being non-algorithmic as the criterion for being a heuristic. Selectivity and "reduction of options to consider" were now important. Efficiency was also downplayed. They said that in so far as their British Museum algorithm selects just theorems and not just formulas it is restricting search and is hence heuristic![51] For the same reason they then said the Matching process is definitely heuristic.[52]

ii. Resolution Theorem Proving

We will now proceed to consider another theorem proving application, resolution based theorem proving, but rather than cover a particular program in detail I will present the general

---

[51]Newell and Simon (1972), p.120.
[52]Ibid., p.139. Siklossy et al. (1973), p.4, report that in a personal communication with them Newell has remarked that the Matching process is algorithmic.

method of resolution and then discuss how 'heuristic' has been used in the context of this method.

Resolution theorem proving is based on an inference rule for the predicate calculus that was shown by J.A. Robinson in 1965 to be both sound and complete, i.e., it generates every valid formula and only valid formulas of the predicate calculus.[53] To prove a theorem by resolution we first negate it, and then convert it to "clause form" which is basically a conjunction of disjunctions,[54] and then apply the resolution principle wherever possible.[55] The resolution principle resolves formulas of the form (A V B1 V B2 V

---

[53]The completeness of the algorithm means that if any theorem is supplied then the algorithm will eventually supply a proof. If an invalid formula is supplied, however, the algorithm will in general not be able to show this. If it could then first order predicate calculus would be decidable, which it is not.

[54]The handling of quantifiers is fairly complicated. A good introduction to resolution, clause form conversion, etc. can be found in Chang & Lee (1973). A "clause" is defined as a disjunct of literals, and a "literal" as an atomic expression or its negation. Basically, the process of converting an arbitrary formula to clause form involves first converting the formula to "prenex normal form", i.e., a form where all quantifiers have been moved to the front of the expression. Secondly, it requires that we eliminate all existential quantifiers and their variables as follows: existentially quantified variables that are not within the scope of a universal quantifier are replaced with constants, whereas those which are, are replaced with "skolem functions", i.e., functions of those universally quantified variables. Thirdly, we are required to drop all quantifiers. All remaining variables are therefore implicitly universally quantified. Fourthly, we use DeMorgan's rules to convert the expression into a conjunction of disjunctions. And, finally, we split the conjunctions up into separate clauses. In a later footnote I give an example of the original and final expressions of such a conversion process.

[55]Of course resolution can be applied to show that a formula A follows from a set of formulas (S1, S2, ..., Sn) by proving that (S1 & S2 & ... & Sn) -> A is a theorem. Most resolution authors prefer using this "deduction from hypotheses" mode rather than the "proof of a theorem" mode. We then need only negate the conclusion before converting all formulas to clause form.

... V Bn) and (¬A V C1 V C2 V ... V Cm) into (B1 V B2 V ... V Bn V C1 V C2 V ... V Cm), where m and n can be any non-negative integers.[56] This latter formula is called the "resolvent" of the first two formulas. If m and n are zero, i.e., we are just resolving A and ¬A, then we get what is called the "null" or empty clause. This means we have derived a contradiction from the negation of the conclusion, and so our original formula must have been valid. Therefore, deriving the null clause completes the proof.

Resolution has proved easy to program on computers. Completeness, however, is only guaranteed if the algorithm resolves clauses in a systematic fashion that guarantees that no clause that is ever vital to any proof is ever ignored forever. Unfortunately, to date, all algorithms that abide by this restriction have to spend exponentially increasing amounts of effort for just linearly increasing numbers of initial clauses to resolve. There is however a great deal of variation possible amongst strategies to select the order in which to resolve clauses and still preserve completeness. The most basic strategy is a "breadth-first" strategy. It resolves all the possible pairs of initial clauses adding the resolvents to the initial set. It then resolves all possible pairs from this new set and again adds in the resolvents, and so on. The breadth-first strategy is the most obvious strategy. No interpretation of clauses is required; in fact, all clauses are treated identically without discrimination. Because of this, and because it performs

---

[56] A and ¬A needn't be the first formula of their respective clauses, as is shown here.

very poorly, the breadth-first strategy has acquired the status of the "British Museum" algorithm of resolution theorem proving algorithms. This is where the usage of 'heuristic' becomes interesting. If we improve on the British Museum algorithm by adding some discriminating clause selection, then is the procedure heuristic? The answer is yes or no depending on whom you read and on how the additions affect completeness.

For example, consider the "set of support" strategy. The "British Museum" algorithm is basically a "forward" reasoning algorithm since we start with the initial clauses and resolve these until we find a contradiction. The "set of support" is a kind of backward reasoning strategy since it requires that we give preference to resolutions involving the negated conclusion clause and its descendants.[57] If no such resolution is possible then we

---

[57] A is a descendant of B if B and C resolve to A, or if D and E resolve to A and D is a descendant of B.

For purposes of explication, the following resolution sequence could have been generated by following a set-of-support strategy:

Suppose we wish to show (Ax) P(x) from the following premises:

(Ax) (R(x) & (Ey)(P(y) v S(y)))
(Ay) (S(y) -> (Ex)Q(x))
(Ay) ¬(Q(y) & R(y) & (Aw)¬P(w))

So if we negate the conclusion and convert to clause form we get the following clauses. Those suffixed with an 'S' are in the set-of-support, therefore, according to the set-of-support strategy, at least one member of this set must be used in each resolution.

1S: ¬P(x)
2 : R(x)
3 : P(f(x)) v S(f(x))
4 : ¬S(y) v Q(g(y))
5 : ¬Q(y) v ¬R(y) v P(h(y))

Now 1S and 3 resolve to 6S: S(f(x))
" 4 " 6S " " 7S: Q(g(f(x)))
" 1S " 5 " " 8S: ¬Q(y) v ¬R(y)
" 7S " 8S " " 9S: ¬R(g(f(x)))
" 2 " 9S " " the null clause;

hence the conclusion does follow from the premises.

can resolve as normal. The set-of-support strategy is complete and is usually more efficient than the British Museum algorithm since the set of clauses only grows in a direction that guarantees using the negated conclusion. Many irrelavant forward derivations are not developed. But note this; I could find no one who is a contributor to the work of resolution theorem proving that calls the set-of-support a heuristic strategy. This is not true for outsiders. Sampson, whose definition states that a heuristic is any strategem for improving performance, is consistent in calling it a heuristic strategy.[58] Cohen and Feigenbaum, who claim to be using Nilsson's definition of heuristic search, are likewise consistent in calling it a heuristic strategy.[59] Nilsson himself, who is a recent contributor to non-resolution theorem proving work, does not expressly call it heuristic, although neither does he imply it is not.[60] Other resolution theorem proving workers, however, definitely adopt the convention of using 'heuristic' only when talking about strategies that forgo completeness. Robinson and Hunt do this, as do Chang and Lee, although Chang and Lee do also on occasion speak of a few complete strategies as being heuristic.[61]

---

[58]Sampson (1976), p.171.

[59]Cohen and Feigenbaum (1982), p.78.

[60]Nilsson treats complete and incomplete strategies identically (1980, pp.165-74) and says that completeness is of little importance compared to efficiency. (1980, p.165) (Also see 1979, pp.101-3.) In Nilsson (1971), p.228, he states that a theorem proving strategy may be both complete and "heuristically effective". Therefore, if he were pinned down I expect he would admit the set-of-support is heuristic. Nonetheless, he discusses heuristics in this and the 1980 book only in the context of formal graph search and game playing, never while describing the set-of-support or any other specific theorem proving strategies.

What we can gather from this is that within a subdiscipline of AI it is possible for a term like 'heuristic' to acquire a restricted, somewhat artificial role, as compared with its more global usage. There is a wide spectrum of possible clause selection strategies to employ in resolution theorem proving and some researchers think a significant division exists between those that are and those that are not complete. 'Heuristic', because of its association with lack of guarantee and with less formal reasoning, is seized upon as a convenient adjective to describe incomplete strategies. Actually, however, there is often little difference between complete and incomplete strategies. For instance, consider the case of the "linear input form" strategy. This strategy requires that of each pair of clauses to resolve, at least one is from the initial or "base" set of clauses.[62] It is incomplete but can be made complete by weakening the condition on

---

[61]Robinson (1971), p.5, Hunt (1975), pp.331-6, and Chang and Lee (1973) pp.152-8. Loveland (1978) rarely uses 'heuristic' at all so I cannot categorize him with the others.

[62]For purposes of explication, the following resolution sequence could have been generated by following a linear-input-form strategy in resolving the clauses from the preceeding example (used in illustrating the set-of-support strategy). Clauses 1 through 5 are in the base set so we suffix them with 'B'. According to the linear-input-form strategy at least one member of this set must be used in each resolution.

1B: ¬P(x)
2B: R(x)
3B: P(f(x)) v S(f(x))
4B: ¬S(y) v Q(g(y))
5B: ¬Q(y) v ¬R(y) v P(h(y))

Now 1B and 3B resolve to 6 : S(f(x))
" 4B " 6 " " 7 : Q(g(f(x)))
" 5B " 7 " " 8 : ¬R(g(f(x))) v P(h(g(f(x))))
" 1B " 8 " " 9 : ¬R(g(f(x)))
" 2B " 9 " " the null clause;
hence the conclusion does follow from the premises.

the parents slightly, so that now at least one parent must either be in the base set or be an ancestor of the other parent.[63] Since this small change has not significantly affected the character of this strategy's clause selection, it seems to me incorrect to make such a change the basis for retracting the attribute 'heuristic'. True, there may be more work required to determine when a clause is an ancestor of another, but it is precisely practical considerations like this that are ignored by those who use incompleteness as the criterion for calling a srategy heuristic.

Another argument for the slight relevance of co eteness in distinguishing strategies comes from Bernard Meltzer. He has pointed out that some incomplete strategies are nonetheless complete for all theorems we should ever likely want to prove.[64]

In addition to this usage meaning incomplete, one can find other interesting usages of 'heuristic' in the resolution theorem proving literature. For example, Chang and Lee say that using heuristics is often the only way to prove a difficult theorem.[65] The impression given is that one has a general-purpose complete strategy that fails due to time limitations for some theorem. One tries out an assortment of heuristics and one of them manages to restrict search enough to bring the theorem within reach during that time. Heuristics are thus likened to tools used in a trial and error fashion.

Another usage is associated with a class of resolution

---

[63]Nilsson (1980), p.169-71.
[64]Meltzer (1971), pp.27-9.
[65]Chang and Lee (1973), p.152.

directing strategies which employ heuristic evaluation functions. Evaluation function strategies are similar to those found in game playing programs (such as computer chess programs) for guiding tree search. They employ numerical functions to assign a degree of "goodness" to a clause, i.e., its desirability for use at some stage of resolution. For example, if it has a minimal number of parentheses, disjuncts, symbols, or some weighted combination of such features, then it is given preference. Also, sometimes clauses that are evaluated poorly will be temporarily or permanently removed from the set of clauses to consider.

Another usage can be inferred from the idea of an arbitrary evaluation function for clauses. Chang and Lee describe a method of estimating the weighting to give to the features used in the function.[6] Any such function, however, is called a heuristic evaluation function because of its heuristic character, regardless of the quality of its evaluation. So Chang and Lee are here using 'heuristic' divorced from the usual qualitative element of the search. All that remains is the directive element. Ordinary language is flexible like this; any word can have its meaning abstracted or stretched. What an abstraction like this tells us is that some feature of the abstracted concept is strong enough to warrant applying the term when solely it is present. In this case the feature is the search directing element. The search directing feature of heuristics is the subject of section D in the following chapter.

---

[6] Ibid., p.153-8.

The above usages of 'heuristic', which are employed when we wish to mean incomplete strategies, trial and error tools, evaluation functions, and search directing functions, cover all my encounters with 'heuristic' in a resolution theorem proving context where it is used to describe a type of strategy used to select which clauses to resolve next. There have been other uses of 'heuristic' in relation to resolution, however. For example, Bledsoe uses 'heuristic' to stand for assorted rewrite rules which transform set theory formulas as well as general first-order predicate logic formulas into more convenient pieces, and then calls resolution as a subroutine to try to establish a portion (subgoal) of the desired proof.[67] to resolve next, they are just transforming the set of clauses in order to make resolution more fruitful. The rules have the form: if a formula with form X is present then transform it to form Y. An executive routine organizes the use of the heuristic transformation rules and of the resolution subroutine. Heuristics here are very much like Newell, Shaw, and Simon's "methods".

Recently non-resolution techniques which emphasize natural deduction have come into vogue again and have challenged resolution's supremacy.[68] 'Heuristic' has tended to be used

---

[67]Bledsoe (1971). An example of such a rule, called a "split" rule because it applies to simplifying logical expressions, is "split 'A<->B' into 'A->B' and 'B->A'". Another such rule, called a "reduction" rule because it simplifies expressions by employing definitions on the semantics of the domain, is "reduce 'x is an element of A intersect B' into 'x is an element of A' and 'x is an element of B'".

[68]See Bledsoe (1977) and Cohen and Feigenbaum (1982) for surveys of recent non-resolution theorem proving work.

The reason resolution techniques are contrasted with natural

differently to suit this different mode of deduction. Although natural-deduction programs can also have heuristics that, like reduction heuristics, are technical strategies for deciding what formulas to work with next, and these strategies can be very dependent on the peculiar logic of their individual programs, natural-deduction heuristics can also be common-sensical problem-solving rules based on principles that we all use, often automatically, and which we might not think to call heuristic.

Take for example the powerful natural-deduction program, THINKER, of Pelletier and Wilson,[69] which is based on the natural-deduction logic system of Kalish and Montague.[70] Some of THINKER's strategies remind one of resolution strategies; they are technical and non-obvious, and they often appear specific to the particular programming organization. For example, to establish a goal which is universally quantified THINKER will first try instantiating with one of a finite number (40) of variables. But the interesting heuristics are those that good problem solvers often apply automatically, and which research like that on THINKER

---

[68](cont'd) deduction techniques is because resolution is very unnatural: the converted clause form expressions are often not as intuitive as the equivalent expressions employing implications, equivalences, and existential quantifiers; the Modus Tollendo Ponens ($A \vee B, \neg A \mid B$; $A \vee B, \neg B \mid A$) rule of inference upon which resolution is based is only one of the reasoning patterns that humans use and even then it is not as popular as Modus Ponens ($A \rightarrow B, A \mid B$) and Modus Tollens ($A \rightarrow B, \neg B \mid \neg A$); and resolution tends to confuse the premises with the conclusion and thereby often lacks apparent goal-directedness which even strategies like the set-of-support do not adequately capture.

[69]Pelletier and Wilson (1983). Also described very thoroughly in Pelletier(1983).

[70]D. Kalish and R. Montague, Logic, (New York: World, Harcourt, and Brace), 1964.

has uncovered. For example, one of THINKER's heuristics is: "if we have just established a step in a proof then we ought to first test if we are only one step away from establishing our goal/subgoal, rather than blindly see what other steps we can generate by applying the inference rules on this last step". Another is: "if a goal has a main connective of '&' or '<->' then first try splitting the goal before trying to establish the goal by contradiction." Such rules are akin to resolution theorem proving strategic rules in that they allow us to better organize our activities. They differ, however, in the manner in which we work with them, the way they are regarded, and the new ways they can be discovered and tested. Resolution heuristics are a bit of an embarrassing necessity since bare bones resolution performs so poorly; the impression one receives from the resolution literature is that the fewer heuristics needed, the better. Resolution heuristics are only discovered by analysis of properties of clauses and of paths in a decision tree, and we can only test resolution heuristics in computers, or at best in small hand simulations. Contrast this with natural-deduction heuristics, where one can also introspect to find the heuristics, one can test the heuristics on one's own style of problem-solving, and where the heuristics tend to be viewed positively as another facet of human capability, the more of which we can include, the better.

Therefore, what I am saying is that although resolution and natural-deduction heuristics are alike in essence, by being fallible but plausible strategic rules, they are nonetheless regarded and employed quite differently in their respective user communities.

This concludes Chapter II. Let us review our progress. In section A we encountered a variety of definitions given to 'heuristic' from the inception of AI to the present. There I made a point of highlighting all the properties of heuristic that have been emphasized by someone or other. These properties will reappear as the subject of the subsequent chapter, where an in-depth analysis of the concept of heuristic will be undertaken. In section B we succeeded in getting acquainted with actual computer implemented heuristics in the dynamic field of theorem proving. We saw firsthand how 'heuristic' has evolved in meaning for a particular subgroup of AI. Basically, section B has stocked us with detailed examples, has given us the opportunity to see firsthand many of the properties described in the definitions of section A, and has in general allowed us to clarify our intuitions regarding heuristics. In sum, this chapter has provided us with the raw materials and background knowledge with which to build and appreciate the analyses given in the next chapter.

III. The Essential Properties of Heuristic Devices

We have set out to define 'heuristic'. The most basic question is, what category of thing is a heuristic? If there is anything that is clear from the mix of definitions cited in Chapter II it is that a heuristic can be almost anything: a rule, a strategy, a tool, a piece of knowledge, a trick, a function, a program, a data structure, and so on. It appears that what is important is not so much the thing as the nature of its use by, and interaction with, a problem-solving system. Therefore, I will follow Feigenbaum and Feldman and choose the noncommital term 'device'. We can start by saying: a heuristic can be any device used by a problem-solving system. What other properties such a device must possess, I will now describe.

A. The Element of Uncertainty in Applying a Heuristic Device

We have seen how, in many of the definitions, 'heuristic' has been opposed to terms like 'algorithmic', 'guaranteed', and 'complete'. I will argue in this section that the central idea underpinning these definitions is that heuristics exist in a context of subjective uncertainty as to the success of their application. I will explain in just what respects those, like Minsky, who think that heuristics are perfectly compatible with algorithms are correct, even though there is a genuine conflict between these two notions. I will also clear up some of the confusions that exist over this issue. In particular I will show in precisely what sense Nilsson's A* algorithm is heuristic, and in only what sense we can say a British Museum algorithm is heuristic.

Central to this key property of uncertainty, and as we saw, present at the very earliest adoptions of the concept of heuristic by AI, is the notion of algorithm. 'Algorithm' has many meanings, although I doubt if ambiguity has caused any of the disagreements over definition. If we define algorithm as merely "a set of [formally defined and uniquely interpreted] rules which tell us, moment to moment, precisely how to behave",[71] then any procedure for making decisions is algorithmic, and hence all heuristics implemented on computer, or otherwise strictly formulated, are algorithmic. I will use "procedure-algorithm" to mean this type of

---

[71]Minsky (1967), p.106.

algorithm. However, when 'heuristic' has been considered opposed to 'algorithm', 'algorithm' has always had a much stronger sense which includes an element of guarantee about finding a solution:

> Definition. Given both the problem and the device, an algorithm is the precise characterization of a method of solving the problem, presented in a language comprehensible to the device. In particular an algorithm is characterized by these properties.
>
> 1. Application of the algorithm to a particular input set or problem description results in a finite sequence of actions.
> 2. The sequence of actions has a unique initial action.
> 3. Each action in the sequence has a unique successor.
> 4. The sequence terminates with either a solution to the problem, or a statement that the problem is insc[illegible].[72]

If the last restriction is too strong we may define 'semi-algorithm' as follows: "a method that will halt in a finite number of steps if the problem posed has a solution, but will not necessarily halt if there is no solution."[73] For some problems there is always a solution, eg., adding two integers, and so this distinction does not apply. I call such algorithms "simple-algorithms".

We have seen how in the case of theorem proving a strategy was called "complete" if it was a semi-algorithm, generating proofs when these existed but not necessarily when a proof did not exist.[74] We have also seen that 'heuristic' has indeed been opposed to this

---

[72]Korfhage (1976), p.48.
[73]Ibid.
[74]There are some partial termination criteria. For example, in resolution theorem proving, if a complete strategy, after application to the entire set of clauses, generates no new clauses, then resolution will never terminate and so the formula is not a theorem.

type of semi-algorithm, although not universally by all authors. Newell, Shaw, and Simon opposed it to semi-algorithms,[75] while Tonge and Slagle opposed it to simple-algorithms.[76] Feigenbaum and Feldman implied a contrast with simple-algorithms and also seemed to say that there is no issue here.[77] We have seen that Minsky denied any opposition with algorithms. Both Nilsson and Jackson denied that heuristics need sacrifice a guarantee of finding a solution, although neither said anything about starting out with a guarantee, or what happens if one should find that the so-called heuristic does guarantee finding a solution.[78] Raphael and Sampson both implied that there is no opposition.[79] Boden argued on the one hand that there is no opposition to simple-algorithm or to semi-algorithm, but on the other hand there is a contrast insofar as heuristic programs postpone decision making whereas algorithms require all decisions be precisely specified beforehand.[80]

Given this mix of conflicting claims one could simply do as Barr and Feigenbaum do, namely, state that 'heuristic' is an ambiguous term and that to keep things clear one will be using such and such a definition.[81] This response is inadequate, however,

---

[75]See quote on p.10-11 above.
[76]Tonge (1960), p.172; Slagle (1963), p.194.
[77]Feigenbaum and Feldman (1963), p.6.
[78]Nilsson (1980), p.72, Jackson (1974) p.95.
[79]See quotes on p.20 above.
[80]Boden (1977), p.347-8. I disagree with this latter point of hers. Numerous heuristic programs, lik DENDRAL, AM, LT, Samuel's checker player, all have all of their rules prespecified. True, I do not know what decisions will be made a ter their commencing, but neither can I for the classic British Museum type algorithms. It is whether or not these rules can guarantee a solution that has always been the traditional issue.
[81]Barr and Feigenbaum (1981), pp.28-9.

because it ignores several reasons for believing in one correct definition. These are: the single origin of the term in the AI literature, i.e., the work of Polya; the fact that AI authors have placed so much theoretical weight on this specific term; and that they have not given their definitions with the air of "for convenience I use the term ..." but with the impression that they have captured what is really important and common to this branch of research, namely, the use of rules that save effort, or provide satisfactory solutions, or lack a guarantee, or what have you. Therefore, I believe a proper analysis of heuristic must result in one definition, and this one definition must show adequate appreciation for all the ideas which have been linked to it, and it must be able to explain any incompatibility among such ideas.

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

An algorithm presumes a problem and a precise step by step procedure that solves the problem or shows it unsolvable. Therefore, if we have a problem and we have an algorithm for that problem, then, so to speak, we should have no problem. So why do such problems so often remain? Because the real problem has not been solved, which is to provide a solution within certain resource limits. Such resources include time, space, and processor type, and, on the user's part, the effort to use and to remember how to use this algorithm. So the real problem is much more complicated and often the guarantee of providing a solution must be withdrawn.

Now, if we realize that heuristics never occur outside a context of such practical problem-solving then we can see how to resolve the algorithm versus heuristic conflict: heuristics and algorithms usually are not opposed because they usually apply to different classes of problems. Heuristics apply to the complete or practical problem whereas algorithms apply to the narrow "any solution will do" problem.

Heuristics were never meant to be distinguished from algorithms except in those cases where: a) the algorithm provides poor solutions to the p ctical problems, and b) the algorithm claims to guarantee solving the practical problems. A strategy like the set-of-support is therefore algorithmic for proving theorems, but, because it is better than the British Museum algorithm for solving the real problem of proving theorems in reasonable time, and because it does not claim infallibility at such, it can also qualify as a heuristic. Therefore Minsky and others were right all along in saying that practical algorithms can be heuristic.[82]

In sections B and C of this chapter I will study how heuristics are opposed to poor solutions, be they algorithmic or blind and random. In what remains of this section I wish to vindicate the other side of the argument. There is a reason why authors naturally choose algorithms as contrast with heuristics. I

---

[82]It was actually in this sense that Newell, Shaw, and Simon originally opposed 'heuristic' to 'algorithm'. See their first definition on p.10 above. Their second definition (p.11) ignores practicality and opposes heuristic to semi-algorithm. 'Heuristic', however, did not abide by this artificial restriction, and, as we saw, by 1972 Newell and Simon had conformed to standard usage.

now hope to show that the claim to guarantee a solution is based on the element of confident and assured decision making which is antithetical to the notion of heuristic.

James Slagle hit the nail on the head when he emphasized in his definitions of 'heuristic' the property of not knowing if an action is the best thing to do next.[83] I have never found heuristics used in a context where it is known that the device at hand is the best thing, i.e., the optimal thing, to apply to help solve the problem. For example, we know the set-of-support strategy is complete, but are its decisions optimal? Not only are there no known instances of this, a fact that could be accidental, but I also believe there could be no such legitimate usage. This is because any situation employing such a usage would be (s)trained and unnatural, hence illegitimate. I will establish this point by build an intuitive picture of proper usage surrounding situations involving optimal procedures.

First of all, if we have an optimal strategy then solving its associated problem would either be trivial or else it would involve an unavoidable and tedious execution of the required automatic procedure. We would be doing the absolute best possible, so we could not rationally expect any room for improvement. However, heuristics are either rules to aid discovery, a la Polya, or rules that offer tradeoffs, simplifications, etc. which overall introduce a small cost in hopes of a bigger payoff. Hence they are used in an atmosphere of expected performance improvement.[84] The problems

---

[83] See his definitions on p.14 above.

heuristics are posed to help solve are always formidable and challenging, and heuristics offer a toehold for solving them. Such problems are neither trivial nor solved in an automatic fashion in an atmosphere of tedium. Therefore it is safe to say that we would never apply 'heuristic' to a process which was already believed optimal. Such a situation simply wouldn't arise.

So now we only have to see about applying 'heuristic' to processes which start out tentative but are later thought to be optimal. For any problem worthy of the name, i.e., a tough one for which all trivial solution techniques have been tried and have failed, it would take some very clear, intuitive, faultless analysis and reasoning to convince me that any process, hitherto believed non-optimal, was now the best imaginable. This, I dare say, would be a dramatic and surprising event. In fact so dramatic as to change my entire perception of the problem and of this solution technique. The problem would recede from my concerns; it would now be either trivial or as unexciting and unavoidable as putting out the garbage; and I would not worry about how to handle such a problem. The solution technique would be likewise transformed. It would now be a clear-cut obvious method. All competition would fade away. There would be no question of tradeoffs with, or simplifications over, such methods. What for, since all competitors would be just misguided? In sum, I think I have painted a picture in which no one could claim continuity with tradition by applying 'heuristic' to such a transformed process.

---

[84] I show in section C of this chapter that expected performance improvement is implied in the most common AI usage of 'heuristic'.

The only excuse for continuing to call it heuristic would be one based on habit, since, after all, its entire character would have changed.

My argument has layed out a clear boundary between optimal and non-optimal strategies as regards the use of the term 'heuristic'. This I believe is sufficient to establish a definitional property. Therefore I claim that heuristics are incompatible with knowledge of optimal decision making and that this is an essential property of heuristics.

Note that this property amounts to saying that heuristics are never thought to guarantee a solution to the practical problem. However, uncertainty as to optimality is a better criterion to use than lack of solution guarantee, because the meaning of optimality includes the element of practicality. Optimality forces us to assume the practical context whereas solution guarantee risks the confusion of problem solution and practical solution.[85] So heuristics really are opposed to algorithms, but only to the algorithms for the complete practical problems.

The following example will help demonstrate the superiority of "uncertainty as to optimality" over just "uncertainty as to solution". Suppose there is a dispensing machine holding five red gum balls and five black gum balls. The machine can be operated indefinitely and for free, but you can pull out only one gum ball at a time and you cannot pull out a second ball unless you have replaced the first ball and shaken the machine a little. You would

---

[85] In other words, the essence of my argument is that this confusion is responsible for the whole heuristic versus algorithm issue.

like to have a red ball, so you use the following strategic rule: choose a ball and if it is red keep it and stop, but if it is black replace it and repeat the process. Using this stategy it should not take you too long to get a red gum ball, but there is no certainty of success. It could take forever. The rule however is optimal, so if you know this you will not say you are following a heuristic; you will merely say you are doing the best that can be done under the circumstances. Now, had you used the solution guarantee criterion, however, you would not have been so readily able to disqualify this strategy from being heuristic, since it does not guarantee a solution.

Having made my main point for this section, viz., that 'heuristic' can not correctly refer to an optimal decision process, I would now like to develop some refinements, ramifications, and consequences of this criterion and thereby help consolidate our intuitions concerning it.

My first point is that subjective uncertainty is meant in our criterion. A particular device may in fact always arrive or seem to arrive at a solution, but as soon as the user recognizes or even strongly suspects this, then he is likely to withdraw the adjective 'heuristic'.

According to Barr and Feigenbaum this actually happened in the case o[illegible]lpha-beta pruning tree search rule.[86] Many game

---

[86]Barr a[illegible]genbaum (1981), p.29. For a good introduction to, and thorough treatment of, mini-maxing game-tree search and the alpha-beta technique see Nilsson (1980), pp.112-28, or Nilsson (1971), pp.137-52.

Alpha-beta pruning is a technique applicable in game-like situations where each player alternates moves. It can apply only

playing programs use alpha-beta pruning as a matter of course because it guarantees retaining all mini-maxed optimal solutions, costs little to implement, and otherwise does not affect search. It is known to always help search and never to hinder it, and there is no doubt of this whatsoever. But against Barr and Feigenbaum I say that it is not because alpha-beta pruning guarantees a solution that 'heuristic' has been dropped, but rather because the alpha-beta operation is thought to be so absolutely fault-free that it is no longer considered a strategy; it is considered merely an automatic aspect of mini-max type game-tree search.

Barr and Feigenbaum go on to mention that, unlike the case of

---

**(cont'd) if the player employing it (call him A and his opponent B) uses a mini-maxing strategy, i.e., where A evaluates his and B's moves at least two moves into the future, and where A will choose the best move left after B chooses his best move (or rather the move that A thinks is in B's best interest) after A has chosen his best move, etc. Mini-maxing allows valuations to be backed up the decision tree as follows. All states n moves away are evaluated and the min (B's turn) or max (A's turn) of states reachable from the immediate precursor states n-1 moves away is made the value of each such immediate precursor state. The min or max of these is then backed up one more level, and so on.

All alpha-beta pruning allows one to do is to avoid further move evaluation when this would be pointless. Suppose A determines that from one of the possible game states that can be reached from the current state player B could then move (m1) so as to give A several options, the best of which is to move to a state of low valuation (say value=5). Then if A subsequently determines that there is a state accessible to him should B make some other move (m2) which yields a higher value (say value=8), then A can stop trying to find any high value moves to play after B's m2, because according to the mini-maxing assumption, B will now definitely not want to choose m2, since this would give A a move to at least a value 8 state when m1 would have given A at best a value 5 state. This is an example of a "beta-cutoff". An alpha-cutoff is like a beta-cutoff only where the roles of A and B are reversed; i.e., A needn't consider any further moves open to B from state S2 after discovering B has an option of going to a low value state from S2, because A has previously determined that he can move to a state S1 of higher valuation, since in the end A must prefer going to S1 rather than S2.

alpha-beta pruning, in the case of the A* algorithm 'heuristic' was not withdrawn when it was discovered to be an algorithm.[87] This does not contradict my claim, however. Rather it supports it, because the A* algorithm is not known to be practically optimal for finding optimal paths in a state-space, even though it will in time find an optimum path.[88] The optimality resides in A*'s solution,

---

[87]An excellent description of A* occurs in Nilsson (1980), pp.61-88. See also Barr and Feigenbaum (1980), pp.64-6. The account given in Nilsson (1971), pp.43-71, has been extensively revised and corrected in Nilsson (1980).

Actually A* is the name given to a class of algorithms. These are used for searching graphs for minimal cost paths from a start to a goal node. In brief, an A* algorithm is like a breadth-first or depth-first algorithm, only, rather than expanding the nodes in the order in which they are found (breadth-first) or always expanding the most recent one found (depth-first), an evaluation function is applied at each step to estimate the shortest or least expensive (if costs are associated with each arc) path, and this path is developed next. The algorithm represents a class because the function $f(n)$ is just a schema of the form $f(n) = h(n) + g(n)$ where $g(n)$ is the estimated cost of proceeding from the start node to node n, and $h(n)$ is the estimated cost of proceeding from n to a goal node. The heuristic element is introduced in h which we want to be a good estimator. (It turns out that no heuristic information is ever given to g since it is desirable to use for g the cost of the actual current path (while running A*) to n. We can in time find a shorter path to n, but so long as when this happens we make sure to go back and recalculate all g for n and its successors then this will not affect A*'s guarantee nor the character of its search, i.e., it will continue following the estimated minimal path to date.)

What we choose for h depends entirely on the problem and the researcher's intuitions; A* says nothing about what h should be. For instance, if $h(n) = 0$ (and $g(n)$ as usual is the actual cost to date of arriving to n) then A* is nothing more than the standard Minimal Cost Algorithm given in Operations Research texts. And if the graph has equal costs for all its arcs then this algorithm is further reduced into the breadth-first search algorithm.

[88]Actually, it is guaranteed to find a minimal cost path only if a condition is placed on the choice of h such that, for each node n, h's estimate of the cost from start node to n is less than or equal to the actual minimal cost from start node to n.

N.B., there is a result sometimes called the "optimality of A* theorem" which however only concerns the relative optimality of two A* algorithms. It states that if one A* algorithm is "more informed" than the other (i.e., for all nongoal nodes n, its $h(n)$

not its method of arriving at this optimal solution. Nilsson recognized this himself and was quick to distinguish minimum solution path cost and minimum search cost. The latter includes memory and time considerations. A* makes no claims regarding search cost. Therefore, being an algorithm is irrelevant to A*'s being a heuristic. It is heuristic because the choice of h[88] can dramatically affect the number of nodes expanded, which might then dramatically reduce search cost. We base our choice of h on some aspect of the problem and thus its search improvement prospects have plausibility, and yet we are not certain that this h will minimize the total expense required to come up with a minimal cost path.

Another point about the non-optimality criterion is that it explains our willingness to accept, and our only expecting, a satisfactory solution when we employ heuristic devices. Tonge could easily include this aspect in his definition because his assembly-line balancing problem allows for a range of satisfactory solutions. In theorem proving we can have only one acceptable solution to the problem of providing a proof,[90] but we may have many satisfactory solutions to the practical problem of supplying a

---

[88](cont'd) is greater than or equal to the others h(n)), then this more informed A* will expand fewer nodes than the other. However, nothing is said either about optimality over other non-A* algorithms, or about any practical effects of the one h over the other. For example, the efforts to compute each h may vary considerably.

[89]See the preceeding footnotes for a definition of h.

[90]Although, because a proof is defined as a sequence of steps, not all of which need to to be useful, there are always a number of proofs possible for any given theorem. But even if we required all steps to be useful, ther are still typically many possible ways to prove any theorem.

proof within certain resource constraints. For example, any such proof generated by using less than 30,000 bytes of storage and supplied within an hour could be satisfactory.

A third point about the non-optimality criterion is that optimality is perceived by comparison with other decision devices. If we are not aware of any other devices then we are not likely to be in the type of competition oriented frame of mind required to call something heuristic, even though upon reflection we might call it such. For example, take the standard rules for adding two integers which we all are taught in school. I would not call these heuristic, nor optimal, until I had reflected upon the problem domain and had realized that there are other possibilities of whose performance I am uncertain. Because of this potential competition I would lose confidence in the standard rules and might then begin treating them as heuristic. In AI, heuristic methods always exist in a context of competition with other methods, be they algorithms or other heuristics.

James Slagle hinted at another refinement of the non-optimality criterion when he spoke of "significant risk" of not doing the best thing next. For him, an algorithmic transformation, as opposed to a heuristic transformation, is one that "is always or almost always appropriate".[1] I gather that with all words a point is reached where you can choose to now apply or stop applying them; that is, there is a spectrum of confidence in applying a term. Some devices are more obviously heuristic than others. Some are

---

[1] Slagle (1963), p.194.

less than optimal but sufficiently close so as not to be called heuristic. For example, a procedure that solved all its problems in less than half a second could be improved, but who would care? For all intended purposes it would be optimal.

A final point about this criterion is that risk and non-optimality are, in humans at least, often associated with other psychological phenomena such as interest, thrill, challenge, mystery, worry, and contention. Hence these associated phenomena are heuristically useful in detecting the presence of heuristics. For example chess has all of these phenomena present, hence one would expect the presence of heuristics. Contrast this with tic-tac-toe once we've realized it has an optimal strategy.'[2]

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

'[2]In case you were not aware, there is an optimal strategy to tic-tac-toe; it guarantees a draw as the worst case outcome. The point is, if you were not aware of this then you might now feel disappointed, because the associated challenge, thrill, etc. of playing tic-tac-toe would have disappeared.

Uncertainty can also partially explain other ideas often opposed to heuristics such as "exhaustive search" and "complete analysis". If we are thorough and complete then we are certain of an answer. The epitomy of thoroughness is the British Museum style algorithm which systematically but indiscriminately searches everywhere for a solution. This is one reason the British Museum type algorithms are so often contrasted with uncertain, unthorough, incomplete heuristic methods. We will encounter another reason in the next section. Nonetheless, British Museum algorithms can be heuristics for practical problems. The lack of intelligence in these algorithms means they can often search through more possibilities in a given period of time than a heuristic method, since applying discrimination requires effort. It is conceivable that in problem domains where solutions are not sparsely distributed, the British Museum algorithm could perform quite well.[33] Hence the British Museum algorithm can be a plausibly successful strategy which we nonetheless do not believe to be optimal. Therefore it too can qualify as a heuristic.

---

[33]This has actually been observed for two brute-force breadth-first search algorithms. For one of these Siklossy et al. (1973) showed that, by applying minimal computing effort at each node in a theorem proving decision tree similar to the one LT used, for many of the theorems in chapter II of Principia it is possible to outperform LT by several orders of magnitude.

B. The Element of Partial Insight which Underpins and Gives Plausibility to a Heuristic Device

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

Partial insight is what makes heuristics of such interest to theorists of Intelligence. If one has some information about a problem domain's structure but not enough to provide an efficient algorithm for solving all such problems then this information can still be put to use in the form of heuristics to improve problem-solving performance. Since so many real world problems are of this form it is no wonder heuristics have become so popular and are so worth studying.

Lenat has remarked similarly on the domain of heuristic applicability:

> At an earlier stage [of knowing a domain], there may have been too little known to express very many heuristics; much later, the environment may be well enough understood to be algorithmized; in between, heuristic search is a useful

> paradigm. Predicting eclipses has passed into this final stage of algorithmization; medical diagnosis is in the middle stage where heuristics are useful; building programs to search for new representations of knowledge is still pre-heuristic.'[4]

Thus we have a spectrum of confidence levels in decision making. At one extreme are efficient algorithms and other decision processes which we believe are optimal whether or not they guarantee a solution, and at the other extreme we have the most inefficient algorithms and other unprofitable processes in which we place little confidence. Heuristics fall in between; they are plausible'[5] without being certain. The placement of a particular process along this spectrum is, however, relative to our perception of the extremes. For example, Newell and Simon originally spoke of their British Museum algorithm as producing such "simple and cheap" expressions that it could not be heuristic, whereas they later call it heuristic because its generator is only "apparently blind-trial-and-error" since it is so much more selective than one

---

'[4]Lenat (1982), p.222.

'[5]I use the word 'plausible' with a positive connotation to emphasize the element of being "deserving or worthy of some confidence", not with a negative connotation of "apparentness or mere appearance of worth". In section A and B of this Chapter I wish to treat separately the uncertainty of heuristics and the partial confidence placed in heuristics. 'Plausible' could be made to handle both roles, but it might then have kept us from so readily treating these two components separately. Furthermore, 'uncertain' is the superior choice because it avoids the negative connotations that might make the uncertainty seem founded on suspicion of deception or dishonesty.

On the other hand I could find no word superior to 'plausible' for describing the idea of "worthy of partial confidence". Words like 'reasonable', 'admissible', 'probable', and 'tenable', all have their own peculiar connotations that I did not wish to include, and furthermore there is a tradition in AI of using 'plausible' in definitions of 'heuristic'. For examples, see the definitions quoted above by Polya (p.7 above), Gelernter (p.11), Slagle (p.13 and p.14), and Jackson (p.18).

that generates all wffs."[96]

As a defining ingredient in heuristics, partial insight offers more than just confidence. Insight is the core of a heuristic's intelligence, its reason for being. A particular heuristic is represented as a particular insight. Without a genuine grasp of some aspect of the problem a device must perforce contribute nothing to problem-solving. It could only masquerade as a heuristic until its luck wore out.

I will illustrate how heuristics are based on partial insights with several examples.

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

In geometry, experience shows that one can grasp relations of proportion better when these are presented graphically rather than numerically. The same applies in set-theory or graph-theory, where diagrams often make patent what the symbolism leaves obscure. The heuristic "draw a diagram" is based on such insights. At other

---

[96]Newell, Shaw, and Simon (1957), p.116. Newell and Simon (1972), pp.120-1.

times the human visual system misleads, say when trying to determine which area is larger, which angle is closer to 90°, etc.; so here one will be advised to use the less biased numerical tools.

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

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

This is the main point I wished to make in this section.

I would now like to develop this idea of partial-insight based plausibility by discussing some of the forms this insight can take. In the end we will have sufficient reason to believe why the study of forms of insight could be an active subfield of heuristics research.

When we analyze insight we see that it comes in a variety of forms. There is a simple insight that can be expressed in simple terms. The example from symbolic logic, that $\neg\neg A$ is equivalent to A, is a simple insight since we can describe it simply. Then there are insights that are not easily expressible, but are nonetheless present. For example, Samuel's checker playing program employed a polynomial evaluation function that included features like "center control", "mobility", "number of forceable exchanges", etc.[7] By

the use of learning algorithms, the evaluation coefficients of these features were automatically adjusted during the course of play in a direction which improved play performance. In one test run 16 function coefficients more or less stabilized after 42 games, with, in this case, the King-Center-Control feature being valued 256 times more than the "MOVE" feature which in turn was valued 8 times more than the Center-Control feature."[8] Now this 16 element polynomial represents an insight into checkers, but how would one express it simply? For one thing the insight is highly dependent on Samuel's particular program and his test samples. One might argue that hence it is really only an insight into how to play good checkers with this particular program. I think, however, that the insight is more universal; it tells us, among other things, that as a general rule kings in the center are more powerful than we might have expected.

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

Among insights that are known prior to their being used for practical decision making, there are those that are describable or expressible and those that are mere intuitive hunches. The "A is equivalent to ¬¬A" case is an example of the former. Polya gives

---

[7]Samuel (1959), pp.100-2.

[8]Ibid., pp.92-3,103. The "MOVE" feature is complicated. It involves "vertical files", "piece counts", the "move system", and more; see p.102.

examples of the latter when he discusses the "signs of progress" in solving a math problem. He speaks of moods and feelings, such as elation, depression, or a sense of aesthetic harmony, which can tell us whether or not we are on the right track, although at times they can deceive us."[99] Of course, AI has tended to work with expressible heuristics, even if these are only expressible in some computer programming language.

Some AI researchers have reflected on the abstract nature of heuristic insight. Boden, Minsky, Newell, Shaw, and Simon speak of moving from the start to goal state and avoiding many fruitless paths by sensing whether one is getting warmer or colder.[100] A kind of negative feedback keeps one on the right track. At each point where alternatives are presented a decision is made. Only some of these decisions need to be fruitful to keep one from going too far astray. Evaluation functions fit this description well.

Another set of reflections comes from Boden, Minsky, and Polya, who speak of the power of analogies and models.[101] Analogies may be as complicated as or even more complicated than the original problem. If sufficient parallelism between the two cases exists then they allow us to transfer both the insights and the heuristics based on these, rather than be forced to rediscover these same insights and heuristics. Models are a form of analogy. They allow us to focus on, or make more salient, some of the more

---

[99]Polya (1945), pp.183-4.
[100]Boden (1977), p.351, Minsky (1961a), p.409ff., Newell, Shaw, and Simon (1957), p.122.
[101]Boden (1977), p.341-4, Minsky (1965), p.425ff., Polya (1945), pp.37-46,180.

relevant aspects of a problem. They offer a more compact representation of a problem's essentials, and are thus a form of partial insight. Their simplicity may also make it easier to discover new insights. And those discovered are likely to concern the more essential aspects of the problem.

I could here just touch the surface of the subject dealing with the forms of insight. Other forms of knowing or packaging knowledge that come to mind, other than intuitions, analogies, models, are sense perceptions, inductions, deductions, theories, and abstractions. I speculate that each brand of insight can spawn a class of heuristics with its own brand of implementation forms, relations to other processes, levels of success, etc. For example, sense perceptions can probably only be used in stimulus-response heuristics like "see a red patch --> stop". Heuristics based on intuitions ("bet with Max, he's lucky") are probably less successful than those based on accepted theories ("never play when the odds favor the house"). Low-level sense perception heuristics ("pain --> withdraw") might be "hard coded" in the cerebellum and operate relatively independently of any other brain processes, whereas high-level theory based heuristics ("if the blood is bad, attach leeches to the neck") might only be executed once certain verification mechanisms determined that the theory applied in this case.

I can only speculate on such matters here. A serious enquiry would examine what Epistemology has classified as the ways of knowing, and what Psychology and Computer Science have determined about human and machine knowledge representations. Because

heuristics are always founded on partial knowledge, the results of this enquiry should tell us much about the kinds of heuristics that are possible.

C. The Element of Performance Improvement Expected from a Heuristic Device

In section A of this chapter we were introduced to the idea that heuristics presumed a practical problem-solving context. In section B we were told that insight was the foundation of their being useful. Here we shall examine how heuristics relate to performance. I will first show what may seem obvious but should not be, that heuristics are used to help improve the performance of a problem-solving system. In this regard they are like tools introduced to fix or enhance a system. I shall then examine the notion of performance improvement and argue that it reduces to increased efficiency, that is, receiving more benefit out for effort put in. I will then show how the various permutations of decreasing effort and increasing benefit explain many of the forms in which heuristics occur.

First of all, it should be clear that we would not be using heuristics in problem-solving, in discovering solutions, guiding search, etc., if we did not believe that they were useful — that they contributed something. This is so patent as to be almost not worth mentioning. However it is not the same thing to say that a device is useful and that it improves performance. An automobile's steering wheel is useful but it does not improve performance. It is a standard fixture. Electronic ignition, being an option that is superior to the standard electro-mechanical ignition, can be said to improve performance. There appear to be two distinct

opinions in AI as to whether heuristics improve performance or are merely useful. Minsky and Sampson explicitly include performance improvement in their definitions. In fact, for them, this is the only significant property of heuristics. Along with Minsky and Sampson are all those that express performance improvement in the form of effort reduction or search reduction. These include Barr, Chang & Lee, Feigenbaum & Feldman, Hofstaedter, Hunt, Jackson, Nilsson, Raphael, Slagle, Tonge, and Winston.[102] In the other camp we find heuristics not introduced to improve the system, but rather, there in their own right from the very start. For this group, heuristics can be standard mechanisms, not just newly introduced superior features. This camp includes: Polya and Newell, Shaw, and Simon, for whom heuristics are aspects of human problem-solving; Boden and Solso, for whom heuristics are just types of problem-solving methods; and Albus, for whom heuristics are high level decision rules. But notice this: almost all of the researchers whose principal field is AI are in the first camp. Polya is a mathematician, Boden a philosopher and psychologist, Solso a psychologist, and Albus a roboticist. Only Newell, Shaw, and Simon are noted AIers in the second camp.[103] But they are not really an exception. Their work is very psychology oriented; much of it surrounds building computer models that reproduce human problem-solving behavior, and their heuristics are often derived from observing human protocols. This observation suggests that

---

[102]Chang & Lee (1973) p.152, Hunt (1975) p.331, Nilsson (1980) p.72, Winston (1977), p.123. For all the others see the definitions quoted on pp.12-20 above.

[103]Simon also has a background in psychology.

what underlies the different usages of these two camps is some sort of different emphasis: the one toward a practical, task-oriented kind of problem-solving by computer, the other toward a more global man-machine theoretical kind of problem-solving. This is also suggested by the fact that even someone like Minsky, who makes performance improvement the primary feature of his definition, uses 'heuristic' independent of performance improvement when discussing human heuristics.[104]

In sum, it appears the majority of members of the AI community employ 'heuristic' to refer to some device applied as an addition to some problem-solving system in expectation of performance improvement. Therefore performance improvement is a property included in the most popular usage of 'heuristic'. All this being said, we must nonetheless admit that there does appear to be a legitimate tradition of allowing 'heuristic' to stand for a preexisting internal mechanism of some problem-solving system, prior to any additions being made.

I have uncovered some very interesting secondary properties of heuristics in AI while investigating performance improvement. These properties all demonstrate the element of adding something to a system that was not present before, and they are all commonly, though not always, attributed to heuristics. These properties are reflected in the use of the following adjectives when describing heuristics: 'practical' (as opposed to theoretical), 'domain-specific', 'ad hoc', and 'empirical', and in the use of the

---

[104]Minsky (1968) p.27.

following nouns in place of 'heuristic': 'trick', 'patch', and 'tool'. To illustrate the performance improvement characteristics isolated by these terms I will describe sample AI programs or programming situations where these terms arise. Heuristics in AI tend to have many of these properties at once. I believe this is because they are all common to the experimental research framework in which AI takes place.

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

Thus where Polya and Newell, Shaw, and Simon would have used 'heuristic' to refer to general methods that are initially part of the problem-solver's outfitting, such as means-ends-analysis, try-and-test, analogous reasoning, and inductive reasoning, others

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

Another research framework within AI in which a type of heuristic based performance improvement occurs can be illustrated as follows. A skeletal program schema is written to handle

heuristics for some problem domain. Heuristics are then tossed in whenever the researcher sees fit, as he acquires experience with the problem, his program, and the behavior of its heuristics. Terry Winograd's natural language understanding program SHRDLU employs a skeletal program schema that allows him to encode world and grammatical knowledge in MICRO-PLANNER statements.[105] For example, he has rules (heuristics) that are "triggered" in the presence of pronouns in order to find their referents.[106] They scan the previous dialogue to see when and how a thing has been mentioned and they give preference to some parts of the context over others. The point is that his schema allows him to continuously add heuristic rules and information as his knowledge of language grows, and to test and refine heuristics as well. Therefore, this experimental add-and-test manner of working with heuristics, which is made possible by the skeletal program schema, reinforces the view of heuristics as performance improving. We add grammatical rules because we want the system to handle more instances of common usage; we test them to discover oversights, grammatical exceptions, etc. Therefore heuristics in AI are often called ad hoc and empirical, and this is not viewed negatively, but rather positively, as part of their general property of being performance improvers.

Numerous other AI systems adopt this same sort of skeletal framework for attaching heuristics. Virtually all of the so called "expert systems" are designed to facilitate this experimental

---

[105]Barr and Feigenbaum (1981), pp.295-9.
[106]Winograd (1973), pp.175-6.

additive performance improvement. They are built so that human expertise can be readily transferred to them. Often the expertise is in the form of heuristics.[107] For example, Douglas Lenat's mathematical concept discovery program, AM, at one point had some 250 heuristics coded as production rules.[108] Examples of such rules are:

> "If f is an interesting relation, Then look at its inverse".
> "If concept G is now very interesting, and G was created as a generalization of some earlier concept C, Give extra consideration to generalizing G, and to generalizing C in other ways."[109]

Each rule is a small mini program written in LISP.[110] Lenat designed his system to facilitate the addition of new rules and he hopes to add more in time.[111] Again each new rule is seen as potentially improving the discovery abilities of the program. Lenat also experimented with AM; he appears to have added rules in a try-and-test fashion as various ideas for enhancing AM's performance occurred to him.[112]

We have just seen how performance improvement tends to be a popular activity in AI and how heuristics adopt this property by patching impractical theories or by being incrementally added to a

---

[107]A good overview of many of the most important expert systems can be found in Barr and Feigenbaum (1982), chapters 7 & 8. A general account of the operation of a "typical" expert system can be found in Feigenbaum and McCorduck (1983), Part 3.
[108]Production rules are basically antecedent consequent rules of the the form "if situation S occurs, then perform action A".
[109]Lenat (1978), pp.30,43.
[110]Lenat (1979) p.266.
[111]Lenat has actually succeeded in automating the process of adding new and improving old rules in his program, EURISKO. See Lenat (1982, 1983a, 1983b).
[112]Lenat (1982), pp.205-7.

general problem solving schema. For the rest of this section I would like to analyze the concept of practical performance. From this analysis we should be able to isolate some of the effects that heuristics have on a problem-solving system that are responsible for their performance improvement.

Performance improvement is called by some, "increasing efficiency", and by others, "reducing effort". What these alternate expressions are emphasizing are but different aspects of the heart of performance improvement: a comparison of benefit with cost, with a perception of either the former increasing and/or the latter decreasing. Nils Nilsson has correctly refined this idea. Because a heuristic is typically applied to many problems, and because the types of problems have a particular frequency distribution, Nilsson says that what we really wish to measure when comparing heuristics is the combined benefits and costs "averaged over all problems likely to be encountered".[113] If we note that Nilsson represents benefits as negative costs then we can interpret his expression, "minimizing ... the cost of the path [to a solution] and the cost of the search required to obtain the path",[114] as maximizing the quality of the solution and minimizing the effort to obtain it. However, he notes that because it is rare to be able to assign a probability distribution to a set of problems, and because it is difficult to quantize and compare both the costs of applying

---

[113]Nilsson (1980), p.72. Recall that Tonge also emphasized that effort reduction on average is all that is required from a heuristic. See the definition given above, p.12, as well as the additional comments in Tonge (1960), p.173.

[114]Ibid.

a heuristic as well as the value of the particular solution generated, the problem of comparing one search method with another or deciding whether a single method is worthwhile is "usually left to informed intuition, gained from actual experience."[115] Although currently we may only be able to rely on informed intuition, I believe Nilsson's analysis points the way to an entire scientific subdiscipline of methods comparisons. Such a science could work towards mapping the distribution of problem types in assorted domains, establishing standards by which costs and benefits can be quantized, and cataloguing the types of effort and benefit and the importance to be assigned to each. AIers have already contributed much to this science. Nilsson has derived a number of interesting theorems concerning semi breadth-first methods.[116] Bernard Meltzer and R.A. Kowalski have invented equations for measuring the efficiency of theorem proving procedures.[117] Lenat has begun classifying the graphs generated for different kinds of heuristics by expressing their performance ("power" or "appropriateness") as a function of task difficulty.[118]

As a conclusion to this section I would like to develop this basic benefit-greater-than-cost intuition and show that a number of properties that are ascribed to heuristics can be derived from it. For instance, Gelernter's idea of heuristics as "sufficiently nonporous filters"[119] and the popular notion of "selective pruning

---

[115]Ibid.
[116]Ibid., pp.72-96. The A* results are amongst these theorems.
[117]Meltzer (1971).
[118]Lenat (1982), pp.210-8.
[119]Gelernter (1959), p.137.

of decision/game trees"[120] both focus on our desire to eliminate from consideration more useless items than valuable ones. Then we have Tonge's shortcuts, simplifications, and adequate solutions. These are attempts to keep the costs (of not having to perform detailed analyses) down but the benefits (quality of solutions) sufficiently high.[121] Abstractions or generalizations of decision devices, in so far as they reduce the number of detailed devices that need to be memorized, and also reduce the need to consider each one each time a decision is required, but do not grossly mishandle too many of the exceptional cases, are also candidates for being heuristic. Or, more generally, any area where we can trade off resource utilization for a slight loss of number of solvable problems, or of quality of solutions, is an area open to performance improvement by heuristic methods.

Conversely, if by whatever means we can marginally increase resource utilization (time, memory, tool, etc.) costs, but recoup a dramatic increase in solvable problems, or a significant increase in quality of some solutions, then this too can qualify as heuristic performance improvement. Expert systems are good sources for finding such effort increasing heuristics since we typically add rules to them, which implies occupying more space and spending more time considering extra rules. True, sometimes this extra effort is offset by reduced search effort, but at other times it is only offset by the benefit of the new problem horizons that are thereby opened up. For example, with MYCIN, an infectious disease

---

[120] Sampson (1976), p.150.
[121] Tonge (1960).

consultation system which after a question asking and answering session with a physician will often generate a list of plausible diagnoses, the addition of a new rule can result in more deduction and other processing going on, but this is judged insignificant compared with the possibility of improving the quality of the diagnoses, perhaps thereby saving a life.[122] Similarly, Lenat wanted AM to discover new mathematical concepts and was quite willing to add extra rules or otherwise increase the effort required to make this possible.

A consequence of the existence of this class of heuristics is that all those definitions of heuristic that use the phrases "effort reduction" or "search reduction" are misleading. "Performance improvement" is the more accurate phrase since it covers all the cases of relative cost-benefit improvement. Although one could argue technically that by increasing effort for even larger benefits one is really reducing the effort required to reach the new hitherto inaccessible solutions, since previously infinite effort could have been spent trying to reach these, this argument would still be inadequate. For the search reduction paradigm is suited to state-space type problem formulations where one knows what one is looking for and one wishes to reduce the effort/search involved in getting to it. This paradigm is unnatural for open ended learning or discovery type problems, as found in Lenat's AM program, where one cares less about finding a particular theorem than discovering new and useful things, and

---

[122] An example of such an improvement is given in Barr and Feigenbaum (1982), pp.95-101.

finding what types of rules and reasoning processes make such discoveries possible.[123] Granted one would like to do all this with as little effort as possible, but "effort reduction" misplaces the priority, whereas "performance improvement" is wide enough to include it.

---

[123]Minsky appears to have realized this long ago: "heuristic methods [are] features that improve the systems' problem-solving efficiency or range of capability." (emphasis mine) Minsky (1968), p.8.

D. The Element of Decision Guidance as the Basic Function of a Heuristic Device

Heuristics have been variously presented in the form of proverbs, maxims, hints, suggestions, advice, principles, rules of thumb, production rules, programs, procedures, methods, strategies, simplifications, option "filters", goal transformers, and no doubt there are others.[124] What is common to all these forms? In this last section on properties I hope to show that heuristics always try to help the problem-solver by guiding his decisions during the course of moving from initial to solution state. Since this is not really a contentious point with anyone I will not belabour it. Nonetheless, because it is a key property it deserves a clear statement. In the end we will discover a few new things about decision guidance. I hope in particular to clear up the issue of whether heuristics can be passive options presented to an executive decision maker, or whether they must be the higher order decision rules guiding the search for a solution.

To show that decision guiding is the primary function of heuristics I will show that the element of choice is always present when heuristics are discussed, and that heuristics as a group do not consistently influence any other element of a problem-solver or his situation. For example, they are not devices that consistently influence memory, clarity of vision, creativity, thoroughness, or

---

[124] See Chapter II for a quoted example of each of these.

any other feature of problem-solving. To phrase it differently, I will argue that the use of 'heuristic' always presumes the existence of a decision mechanism and that the heuristic's effect is to lead this mechanism down one path as opposed to another. The influence may be direct, i.e., the heuristic actually decides where to go. For example, evaluation functions are direct. Or the influence may be indirect, i.e., the heuristic simply changes some aspect of the problem situation. For example, "eliminate complex theorems from the subproblem list". There is no sharp line dividing these two types of influence.

My argument will be by illustration. I will simply bring forth and discuss a representative sample of usages and definitions to support this claim about the universality of decision guidance. We can start with Polya, whose usage we recall was rather different from what is prevalent in AI today. For Polya, any behavioral method considered useful while problem-solving could be a heuristic method. This includes asking oneself certain key questions, drawing a diagram, or trying to rephrase a problem. Since Polya did not use the paradigm of search when describing mathematical problem-solving, these behavioral methods need not be affecting any decision making. They could be influencing some unconscious processes which suddenly inspire the solver to see a solution. Nevertheless, Polya only speaks of his methods as being chosen by a solver. The student should try this, think of it, ask himself this question, etc. Polya does speak of the value of subconscious work but there is no mention of any heuristic methods being used by it.[125]

In the early AI period, as represented in the Computers and Thought anthology, the paradigm of heuristic use is one of guiding search through a problem space.[126] This applies to every author I covered in Chapter II. In their Logic Theorist paper Newell, Shaw, and Simon, like Polya, officially leave open the possibility of heuristics being arbitrary useful processes applied during problem-solving. Yet in fact they solely use them to influence the order of development of the solution path along the subproblem tree. The value of the heuristics is explained by their effect on movement through the subproblem tree, and all their heuristics are clearly decision guiding. The four primary methods in LT directly choose some of the paths to be followed, while the "similarity test" acts as a filter, screening some theorems prior to matching and hence indirectly guiding the course of search.

Gelernter employs a somewhat tree search paradigm and uses his heuristics to filter out less promising decision options. Tonge uses heuristics to simplify wherever possible the entire pattern of activity used to balance assembly lines. Slagle uses the Logic Theorist framework where heuristics both decide what problem transformations to apply next, as well as transform the problems themselves. Minsky introduces heuristics in a context of search where they guide the solver gradually to a solution. He gives "hill climbing" as a typical example.[127] Feigenbaum and Feldman

---

[125]Polya (1945), pp.197-9.

[126]Newell and Simon (1972), p.888.

[127]Minsky (1961a), pp.408-12. Hill climbing is a process of exploring the region immediately about a location or state and choosing to move to the best (highest) location or state found. One then repeats the process until either a goal is found or improvement ceases.

m ion state-space search reduction in their definition and give orted rules of thumb as examples. For these the solver is ortrayed as trying one thing rather than another and is thereby lead down a different problem-solving path.[128]

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

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

Along with this list of usages we can bring forth all the key words used in defining 'heuristic' as evidence that heuristics exist to influence problem-solvers' choices. Proverbs, maxims, hints, suggestions, and advice are clearly meant to influence decision making. Principles, rules of thumb, and rules proper all

---

[128]Feigenbaum and Feldman (1963), pp.6,7.

exist to govern conduct, and in the case of problem-solving one's conduct is typically consciously selected. Furthermore, having chosen to follow a rule, one's subsequent decisions are often altered by the new face the problem now presents. Programs, procedures, methods, and strategies are all organized sets of rules which, however complex, are in effect single rules themselves. Each is but a rule which summarizes a variety of conduct for assorted circumstances and/or through a period of time. Hence they too exist to govern conduct, and the problem-solver decides to follow them or not. Finally, the other things which some heuristics have variously been called, "filters", "simplifiers", "transformers", and such, seem always to have as their purpose the restructuring of the problem situation so that one has a different set of options from which to choose.

This concludes my argument, from cited keyword definitions, and from usage, that decision guidance is the basic function of a heuristic device. I have never found this to be denied, so there is no real issue here. There have been some confusions regarding this property, however, and I will now set about exposing and resolving these. I will first need to describe how, with respect to decision making, there are two distinct ways 'heuristic' occurs in the literature. It is because of not recognizing this that some authors have given erroneous definitions of 'heuristic'.

With regard to the executive's function, which determines the overall direction of activity, a heuristic may be used actively to decide which of several rules, pieces of advice, game moves, or solutions to select, or it may be referred to passively, as one of

the rules, pieces of advice, etc. which is being offered for selection. These two categories are not mutually exclusive, nor need a heuristic belong to at least one category. For example, we saw in the case of LT how the four methods were passive heuristics selected by the non-heuristic executive, but also how they were active heuristics when they were deciding which of the theorems to consider next. An example of a heuristic that is in neither category can also be found in LT. The "similarity test" is not part of the executive since all it does is change the problem environment, not decide the course of problem-solving. And on the other hand neither is it selected from among alternative activities to perform since it is always applied and it has no competitors.

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

Having distinguished these two ways heuristics can be involved

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

I alluded earlier to an apparent contradiction in James Slagle's defining heuristics as being both active and passive.[129] I have no trouble with the passivity. Every heuristic has a "significant risk" of not being "the appropriate next step",[130] and so can in this sense be considered one alternative implicitly selected from among others, even though the others are not even mentioned. But, as we have just seen, it is definitely not true that all heuristics are part of the executive; they do not all decide what should be done next. Even though Slagle qualified his definition with "in this discussion ...",[131] I believe he was nonetheless making his own stab at a generally applicable definition.[132]

---

[129] See p.14 above.
[130] Slagle (1963), p.197.
[131] Ibid., p.192.
[132] To Slagle's credit we can say that he later adopted a more conventional definition, which I quote on p.18 above.

Along a similar vein is James Albus's claim that "A heuristic is a strategy for selecting rules, i.e., a higher level rule for selecting lower level rules."[133] So again heuristics are portrayed as part of the executive, i.e., in control of what is done next. Elsewhere he makes similar remarks:

> In most cases, the search space is much too large to permit an exhaustive search of all possible plans, or even any substantial fraction of them. The set of rules for deciding which hypotheses to evaluate, and in which order, are called heuristics.
>
> . . .
>
> [Heuristics have a] recursive nature. A heuristic is a procedure for finding a procedure.[134]

These last remarks suggest the source of his prejudice, namely, his belief that heuristics only occur in a context that fits the state-space search paradigm. Elsewhere he actually describes all problem-solving as state-space search.[135] When one has a formal state-space network defined, it is easy to imagine that all decisions can be reduced to answering "what path shall I follow?" or "what strategy shall I follow for moving down a path?". Heuristics become the strategies, and the strategies for selecting the strategies, which tell us where to go next.

In showing Albus incorrect we should first make clear that it is implausible that all problems can be made to fit the state-space scheme. As Boden observes, it is hard to define solution and intermediate states for problems like "shall I marry him?" and "how can I write a detective story?".[136] But even within this scheme,

---

[133]Albus (1981), p.284, and quoted in full above on p.20.
[134]Ibid., p.222.
[135]Ibid., pp.281-5.

heuristics can be applied to numerous background duties as opposed to direct choices of what operation to choose next. I mentioned the similarity test of LT above. Another example is Lenat's heuristics in AM, many of which contribute incrementally to prioritizing projects on the "agenda" of things to do next, without individually choosing what exactly is done next.[137]

As for Albus's apparent claim that a heuristic must not only be an executive strategy, it also has to select an executive strategy, I am not certain what is motivating him. Let's consider what Newell calls the "classical tongue-in-cheek example" of a heuristic, "always take a check — it may be a mate".[138] For me this is an individual heuristic that directly selects from amongst possible moves (paths), not from amongst possible strategies. Perhaps Albus has discovered another layer of decision in there. We choose the general heuristic, but perhaps this implies searching for an operative rule/strategy to implement it in particular case. I suspect Albus has in mind a particular model of planning in robots. But whether or not Albus can be defended I still believe we must conclude that all those who use heuristics for just immediate direct decision making do so legitimately, and that Albus's definition is too narrow. But Albus has indeed struck an important possibility for heuristic decisions — their recursive application. In my concluding chapter I will talk a little more

---

[136]Boden 1977), p.350.

[137]Lenat (1978), pp.30-3. Indeed Lenat's executive is very simple and runs without heuristics at all.

[138]Newell (1980) p.16. Newell attributes its origin to Selfridge. You may have noticed that Tonge uses it as part of his definition. See p.12 above.

about recursiveness when I discuss "meta-heuristics".

This is just a taste of the arguments and conceptions that are possible when discussing heuristics in relation to decision. I have hinted at some possibly profitable areas to investigate. Perhaps the first priority for advancing heuristics research in this area would be an accurate account of human decision making and the forms of possible machine decision making. Philosophers, psychologists, and computer scientists might collaberate on this.

IV. Conclusion

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

Although these two properties should be sufficient to eliminate the majority of non-heuristic devices, most AIers use 'heuristic' more restrictively still. They reserve the term for just those devices they have added to their experimental system in hopes of improving its performance. Although I suspect they would

relenquish this property upon a little reflection, this restricted usage is nonetheless prevalent. For instance, it is to be found in the majority of definitions given by AIers themselves. Therefore we must admit this property if we are to give an AIers definition of 'heuristic'. As for what performance improvement means we found that, contrary to many authors, it did not mean search or effort reduction, that this was only half of the equation, the other half being a possibility of improvement in solution quality.

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

establishing that heuristics could be involved in direct active decision making, or merely passively as options to execute, and that therefore some authors were incorrect in thinking that all heuristics chose what course problem-solving would follow next.

Concisely put, a heuristic in AI is any device, be it a program, rule, piece of knowledge, etc., which one is not entirely confident will be useful in providing a practical solution, but which one has reason to believe will be useful, and which is added to a problem-solving system in expectation that on average the performance will improve. As an adjective, 'heuristic' means the sum of these qualities just mentioned which are possessed by a heuristic.

All in all, I believe my analysis has defined 'heuristic', explained 'heuristic', and substantiated all the reasons for this definition in a more thorough fashion than has been the custom to date in AI.

I would now like to make a few less formal remarks concerning the significance of the concept of heuristic and the directions that further research should find fruitful.

Although AIers make regular use of heuristics, and no one denies their utility and importance, one wonders how many appreciate the extent to which the idea of heuristic can be applied. I was pleased to come across one other who imagined that all human intelligent behavior might be accounted for by heuristics. In his 1977 Computers and Thought lecture, "The Ubiquity of Discovery", Douglas Lenat suggested a model of Man where most intelligent behavior could be seen as problem-solving

behavior, and where a large set of general and special purpose heuristics guide this behavior. He reduced both everyday and scientific problem-solving to basically building models and following plans, both of which have often been reproduced by heuristic driven AI programs. Then on a second front he reduced everyday creativity and scientific inventiveness to the basic operation of judging "interestingness", and he explained how his AM program managed this operation with heuristics, and in quite a natural way. He argued that all that remains is to write programs that discover and develop their own heuristics, and he suggested some of the ways this could be done.

I believe this is a new idea in the history of ideas. Here we have Man described not as a rational animal, not as a machine, not as a soul and matter pairing, not as a stimulus response transducer, not as a collection of servomechanisms, but as a user and manager of thousands of rough and ready rules and devices that are by and large effective. This perspective has its own metaphysical impact. Methods of doing things are made concrete and seen as evolving through history with problem-solvers as merely their substrate mediums.[13] It affects our view of people. Magnanimous acts and heinous ones are equalized as merely behavior guided by subjectively plausible sets of rules. The path to great achievements and expert knowledge is seen less as a matter of genius or special talent and more as a matter of acquiring good

---

[13] I am reminded of Richard Dawkins' The Selfish Gene (London: Granada Publishing Limited, 1978) where genes are portrayed in a similar rôle.

heuristics. Education focuses less on acquiring a big set of facts and skills, and more on good thought habits and acquiring certain key heuristics, expecially those for learning other heuristics, for improving the ones one already has, and for motivating one to keep all these key heuristics active. These are but a few of the discoveries made possible by adopting the heuristic metaphysic. I expect the heuristic perspective will grow in popularity on account of its explanatory power and its potential for practical impact. Areas like education, psychology, law, philosophy, anthropology, etc. could enrich their models of Man by incorporating it.

I wish to close by talking about two topics in heuristics studies which I believe are but barely chartered but possess great potential for AI. These concern meta-heuristics and heuristic interaction. These topics point the direction in which our concept of heuristic may some day evolve.

Several heuristics researchers have seen the need for adding mechanisms to their programs for the learning of heuristics. Creating new heuristics can be done either from scratch or by transforming old methods. For this process to be mechanized a general formalism for representing heuristics is needed. To date I have only seen formalisms for heuristic evaluation functions and for production rules. Research into the types of formalism for all types of heuristic devices, with accounts of their advantages and disadvantages, is needed. Then for each formalism what we need are mechanisms expressed in that formalism for doing the following types of things: monitoring behavior and judging when a heuristic is needed (eg., gross inefficiency is present, current methods are

tedious, etc.); assembling expressions in one of the formalisms that are relevant to a problem and offer a possibility of helping solve it; testing these expressions in real or mock situations; evaluating the expressions tested; promoting the favorably evaluated expressions to the status of heuristics and integrating these effectively into the structures that hold the existing set of heuristics; continually monitoring the performance of heuristics; demoting those that are no longer suitable to the current environment; sensing when a heuristic can be improved; creating a new heuristic by generalizing a group of valuable but similar heuristics; creating a new heuristic by specializing a heuristic to suit specific situations; and so on.

These heuristic mechanisms, because they operate on other heuristics, are called meta-heuristics. Meta-heuristics offer particular promise because of the recursive potential they afford. They would enable a system to improve its level of performance at improving its level of performance. Albus, Feigenbaum, Feldman, Findler, and Lenat have expressed similar sentiments.[140]

Heuristic interaction is another subject in its infancy, which a future analysis of the concept of heuristic will likely have to deal with extensively. An example form of such interaction might be

---

[140]Albus (1981), p.222; Feigenbaum and Feldman (1963), pp.7-8; Findler (1976), p.607; Lenat (1977), pp.278-80; and Lenat (1978), pp.48,50-1.

Lenat has taken major strides in proving the value of meta-heuristics. His heuristic learning and improving program, EURISKO, has successfully employed meta-heuristics to learn heuristics for playing a naval war game, to devise new heuristics for AM to use, and to discover a heuristic for speeding up LISP programs, among other achievments. See Lenat (1983b), pp.73-86.

cooperation. Cooperation between heuristics and other heuristic and non-heuristic devices as they work towards solving a problem makes it difficult to evaluate each heuristic's contribution independently. For example, a chess heuristic may be useful when used with certain other heuristics, but harmful otherwise. Competition is another interesting form of interaction. Some heuristics may be less able at solving problems than others, but part of their activity may involve influencing other devices to eliminate their competitors or to give preference to themselves.[141] Ways of policing heuristics, administering justice, or organizing the problem-solving system so that such injustice is minimized, may need study. Another problem in heuristic interaction is how to manage a set of heuristics so as to optimally disperse them and avoid overlapping jurisdictions, but, at the same time, to cover as many of the problem situations as are likely to be encountered. This problem is a bit like trying to select players for a baseball team. One does not need five good shortstops, five good catchers, etc.; one needs one or two good players at each outfield and base position, several good pitchers, etc. Then there are problems surrounding dependence. A heuristic that makes use of other devices can be more efficiently expressed, and the system as a whole may benefit by allowing the sharing of resources. But should these subordinate devices change or be preempted by a heuristic's neighbors then this heuristic's behavior and even survival may be

---

[141]This has actually been observed by Lenat in one of the heuristics EURISKO created, which would do nothing but associate itself with any heuristic of high worth and therby acquire worth itself! Lenat (1982), p.229.

affected.

Cooperation, competition, dispersion, and dependence are but a few of the ways we may be able to interpret the behavior manifested in a problem-solving system employing multifarious devices, be they heuristic or not. The probable utility but uncertain effects of many of these devices in any so complex a system suggests that many of these devices would in fact qualify as heuristics. Hence, as AIers build ever more complex problem-solvers, the prevalence of heuristics should increase and so should the study of the forms of their interaction.

I hope my analysis of heuristic has clarified its meaning, and I hope my remarks above about Man the Heuristic Problem-Solver, meta-heuristics, and heuristic interaction have impressed the reader with the concept's richness and potential.

BIBLIOGRAPHY

Albus, James S., Brains, Behavior, and Robotics, Peterborough, N.H.: Byte Publications Inc., 1981.

Barr, Avron, and Feigenbaum, Edward A., (eds.), The Handbook of Artificial Intelligence, Vols. I & II, Los Altos, California: William Kaufmann, Inc., 1981, 1982.

Bledsoe, W.W., "Splitting and Reduction Heuristics in Automatic Theorem Proving", in Artificial Intelligence, Vol. 2, 1971, pp. 55-77.

-----, "Non-resolution Theorem Proving", in Artificial Intelligence, Vol. 9, 1977, pp. 1-35.

Boden, Margaret A., Artificial Intelligence and Natural Man, New York: Basic Books, Inc., 1977.

Chang, Chin-Liang, and Lee, Richard C., Symbolic Logic and Mechanical Theorem Proving, New York: Academic Press, Inc., 1973.

Cohen, Paul R., and Feigenbaum, Edward A., (eds.), The Handbook of Artificial Intelligence, Vol. III, Los Altos, California: William Kaufmann, Inc., 1982.

Descartes, Rene, Rules for the Direction of the Mind, and Discourse on the Method of Rightly Conducting the Reason, in The Philosophical Works of Descartes, ed. and trans. by Elizabeth S. Haldane and G.R.T. Ross, Cambridge, England: The Cambridge U. Press, 1931.

Ernst, George W., and Newell, Allen, GPS: A Case Study in Generality and Problem Solving, New York: Academic Press, 1969.

Feigenbaum, Edward A., and Feldman, Julian (eds.), Computers and Thought, New York: Mcgraw-Hill Inc., 1963.

-----, and McCorduck, Pamela, The Fifth Generation, Reading, Mass.: Addison - Wesley Publishing Co., 1983.

Findler, N.V., and Meltzer, Bernard, (eds.), Artificial Intelligence and Heuristic Programming, Edinburgh: Edinburgh U. Press, 1971.

-----, "Heuristics", in Encyclopedia of Computer Science, ed. by Anthony Ralston, New York: Van Norstrand Reinhold Co., 1976, pp. 606-7.

Gelernter, Herbert, "Realization of a Geometry-theorem Proving Machine", in Proceedings on an International Conference on Information Processing, Paris, Unesco House, pp. 273-82; reprinted in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 134-52.

Hofstadter, Douglas R., Godel, Escher, Bach, New York: Basic Books, 1979.

Hunt, Earl B., Artificial Intelligence, New York: Academic Press, Inc., 1975.

Jackson, Philip C. Jr., Introduction to Artificial Intelligence, New York: Petrocelli Books, 1974.

Kleinmuntz, Benjamin, (ed.), Formal Representation of Human Judgment, New York: John Wiley & Sons, Inc., 1968.

Korfhage, Robert R., "Algorithm", in Encyclopedia of Computer Science, ed. by Anthony Ralston, New York: Van Norstrand Reinhold Co., 1976, pp. 47-50.

Lenat, Douglas, "The Ubiquity of Discovery", in Artificial Intelligence, Vol. 9, 1977, pp. 257-87.

-----, and Harris, Gregory, "Designing a Rule System that Searches for Scientific Discoveries", in Pattern Directed Inference Systems, ed., by D.A. Waterman and Frederick Hayes-Roth, New York: Academic Press, 1978, pp. 25-51.

-----, "On Automated Scientific Theory Formation: A Case Study Using the AM Program", in Machine Intelligence 9, ed. by J.E. Hayes, Donald Michie, and L.I. Mikulich, West Sussex, England: Ellis Horwood Ltd., 1979, pp. 251-83.

-----, "The Nature of Heuristics", in Artificial Intelligence, Vol. 19, 1982, pp. 189-249.

-----, "Theory Formation by Heuristic Search. The Nature of Heuristics II: Background and Examples", in Artificial Intelligence, Vol. 21, 1983, pp. 31-59.

-----, "EURISKO: A Program That Learns New Heuristics and Domain Concepts. The Nature of Heuristics III: Program Design and Results", in Artificial Intelligence, Vol. 21, 1983, pp. 61-98.

Loveland, Donald W., Automated Theorem Proving: A Logical Basis, Amsterdam: North-Holland Publishing Co., 1978.

Minsky, Marvin L., "Steps Toward Artificial Intelligence", in Proceedings of the Institute of Radio Engineers, Vol. 9, January, 1961, pp. 8-30; reprinted in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 406-50.

-----, "A Selected Descriptor-indexed Bibliography to the Literature on Artificial Intelligence", in IRE Transactions on Human Factors in Electronics, Vol. 2, March, 1961, pp. 39-55; reprinted in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 453-75.

-----, "Matter, Mind, and Models", in Proceedings of the International Federation of Information Processing Congress 1965, Vol. 1, pp. 45-9, reprinted in Semantic Information Processing, ed. by Marvin Minsky, Cambridge, Mass.: M.I.T. Press, 1968, pp. 425-32.

-----, Computation: Finite and Infinite Machines, New York: Prentice Hall Inc., 1967.

-----, (ed.), Semantic Information Processing, Cambridge, Mass.: M.I.T. Press, 1968.

Newell, Allen, and Simon, Herbert A., Human Problem Solving, Englewood-Cliffs, New Jersey: Prentice Hall, 1972.

-----, Shaw, J. Cliff, and Simon, Herbert A., "Empirical Explorations with the Logic Theory Machine", in Proceedings of the Western Joint Computer Conference, Vol. 15, 1957, pp.218-39; reprinted in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 109-33.

-----, "The Heuristic of George Polya and Its Relation to Artificial Intelligence", a paper given at The International Symposium on the Methods of Heuristic, University of Bern, Switzerland, Sept. 15-18, 1980, to appear in R. Groner, M. Groner and W.F. Bischoof (eds.), Methods of Heuristics, Hillsdale, N.J.: Lawrence Erlbaum, forthcoming.

Nilsson, Nils J., Problem-Solving Methods in Artificial Intelligence, New York: McGraw Hill, Inc., 1971.

-----, "A Production System for Automatic Deduction", in Machine Intelligence 9, ed. by J.E. Hayes, Donald Michie, and L.I. Mikulich, West Sussex, England: Ellis Horwood Ltd., 1979, pp. 101-26.

-----, Principles of Artificial Intelligence, Palo Alto, California: Tioga Publishing Co., 1980.

Pelletier, F. Jeffry, Completely Non-Clausal, Completely Heuristically Driven, Automatic Theorem Proving, M. Sc. thesis, the University of Alberta, 1983.

-----, and Wilson, Dan, "Heuristic Theorem Proving", in Thinking: The Expanding Frontier, ed. by W. Maxwell, Philadelphia: Franklin Press, 1983.

Polya, George, How To Solve It, 2nd ed. rev., New York: Doubleday Anchor, 1957.

-----, Mathematics and Plausible Reasoning, 2 vols., Princeton, New Jersey: Princeton U. Press, 1954.

-----, Mathematical Discovery, 2 vols., New York: John Wiley & Sons, Inc., 1962, 1965.

Ralston, Anthony (ed.), Encyclopedia of Computer Science, New York: Van Nostrand Reinhold Co., 1976.

Raphael, Bertram, The Thinking Computer, San Francisco: W.H. Freeman & Co., 1976.

Russell, Bertrand, and Whitehead, Alfred N., Principia Mathematica, 2nd ed., Vol. 1, Cambridge, England: Cambridge U. Press, 1925.

Sampson, Jeffrey R., Adaptive Information Processing, New York: Springer-Verlag Inc., 1976.

Samuel, A. L., "Some Studies in Machine Learning Using The Game of Checkers", in the IBM Journal of Research and Development, Vol. 3, July, 1959, pp. 221-9, reprinted in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 71-105.

Siklossy, L., Rich, A. and Marinov, V., "Breadth-First Search: Some Surprising Results", in Artificial Intelligence, Vol. 4, 1973, pp. 1-27.

Slagle, James R., "A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus", in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 191-203.

-----, Artificial Intelligence: The Heuristic Programming Approach, New York: McGraw Hill, Inc., 1971.

Solso, Robert L., Cognitive Psychology, New York: Harcourt Brace Jovanovich, Inc., 1979.

Tonge, Fred M., "Summary of a Heuristic Line Balancing Procedure", in Management Science, Vol. 7, 1960, pp. 21-42; reprinted in Computers and Thought, ed. by Edward A. Feigenbaum and Julian Feldman, New York: Mcgraw-Hill Inc., 1963, pp. 168-90.

Wang, Hao, "Toward Mechanical Mathematics", in IBM Journal of Research and Development, Vol. 4, 1960, pp. 2-22.

Waterman, Donald A., and Hayes-Roth, Frederick (eds.), Pattern Directed Inference Systems, New York: Academic Press Inc., 1978.

Winograd, Terry, "A Procedural Model of Language Understanding", in Computer Models of Thought and Language, ed. by Roger C. Schank and Kenneth Mark Colby, San Francisco: W.H. Freeman and Company, 1973, pp.152-86.

Winston, Henry P., Artificial Intelligence, Philippines: Addison-Wesley Publishing Co. Inc., 1977.

VITA

NAME: Marc Hector Joseph Romanycia

PLACE OF BIRTH: Hafford, Saskatchewan

YEAR OF BIRTH: 1956

POST SECONDARY EDUCATION AND DEGREES:

University of Alberta
Edmonton, Alberta
1974-1978 B.A. (honours)

University of Alberta
Edmonton, Alberta
1979-1980 B.Sc.

University of Alberta
Edmonton, Alberta
1978-1983

RELATED WORK EXPERIENCE:

Teaching Assistant
University of Alberta
1978-1979

Systems Analyst
Gulf Canada Limited
1981-