

Artificial Intelligence Research Strategies in the Light of AI Models of Scientific Discovery

Herbert A. Simon
Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract and Introduction

Some recent artificial intelligence programs whose task is to simulate the processes of scientific discovery can be taken as models of the history and processes of discovery within the AI discipline itself. Consistently with these models, AI research relies basically on the methods of heuristic best-first search. Because of its necessarily vague and open goals, it works forward inductively (rather than backward in means-ends fashion), guided by a crude evaluation function that tests running programs to identify promising directions.

AI research is empirical and pragmatic, typically working with examples rather than theorems, and exemplifying the heuristic of learning by doing. In its essential reliance on weak methods and experiment instead of proof, it is adapted to the exploration of poorly structured task domains, showing considerable contrast in this respect to operations research or numerical analysis, which thrive best in domains possessing strong formal structure.

At scientific meetings it is customary to schedule, in addition to papers reporting specific pieces of research, "addresses," "keynote speeches," and the like, which may be described as meta-papers. The task of meta-papers is not to report research but to interpret the past and to peer into the future of the discipline. This is such a meta-paper. Presumably you expect me to say where artificial intelligence has been and where it is going.

Clearly, this is not a task for human intelligence. Human beings are notoriously incapable of reviewing history -- especially history in which they have participated -- without rationalizing outrageously to make the past conform to their picture of the present. And human forecasts of the future almost always reveal much more about the forecasters' hopes, fears, desires and dreads than they do about the shape of the world to come.

Early in the history of AI, in 1957, Allen Newell and I made some predictions that became rather notorious (Simon & Newell, 1958). Skeptics and opponents of AI used them as evidence of the recklessness and irresponsibility of the advocates of AI. (Optimistic forecasts seem to attract such charges much more often than do doomsday forecasts.)

Of course our forecasts were neither reckless nor irresponsible. As we said at the time, they represented our attempt to define in concrete terms the nature of the revolution in human affairs that was going to be produced by computers in general and artificial intelligence in particular. As scientists privileged to witness the early stages of a momentous development, we felt a responsibility to interpret that development to laymen, and the predictions were our interpretation. Nor was our forecasting seriously inaccurate, if one allows a time-stretch factor of two or three -- a not unreasonable margin of inaccuracy in such crystal-ball ventures.

I cite this little piece of history not to defend my record as either a seer or a historian, but as empirical evidence for my doubts, expressed earlier, that either foresight or hindsight are fit tasks for human intelligence. Such doubts undermine the very foundations of meta-papers, including this one.

If human intelligence is unequal to the needs of history and prophecy, perhaps we should call on artificial intelligence. Perhaps we should ask what AI has to say about the processes of discovery. After all, we do have, today, a number of artificial intelligence programs that are capable of making discoveries of one kind or another -- I have in mind particularly Doug Lenat's AM program, and Pat Langley's BACON. Perhaps these programs can tell us more about the research process than human beings can.

An AI program that makes genuine discoveries, or one that solves difficult problems, provides us with a theory of the discovery process, indeed, a theory in the most concrete and explicit form that is conceivable. Since these programs reveal to us some of the essential requisites and structure of the discovery process, we can use them to illuminate the history of discovery in the domain of artificial intelligence itself, and to provide some insight into the ways in which we can best proceed in future research and development aimed at new discoveries in that field.

This is the path I propose to pursue in this paper. First, I will summarize what seem to me some of the salient characteristics of successful artificial intelligence problem-solving systems, especially those whose basic task is to make discoveries. Next, I will ask whether this list of program characteristics suggests why the process of discovery in the AI field itself has taken the particular course that it has. Finally, I will turn to the future, and ask what lessons we might learn from this experience in our continuing efforts to extend the boundaries of AI, particularly in the directions of greater capabilities for discovery and for solving ill-structured problems. If this route seems somewhat circular -- AI illuminating itself -- I remind you that circles may be either vicious or virtuous, and I will argue that this is one of the virtuous kind.

I will not try to cover every aspect of AI, and will undoubtedly overemphasize problem-solving and heuristic search at the expense of such areas as visual pattern recognition. This lapse will be the less serious to the extent that the techniques of heuristic search are today invading the domain of pattern recognition, bringing

about a greater degree of unity in outlook throughout the whole field of artificial intelligence. So I will take, as Allen Newell and I did in our Turing Lecture, heuristic search as the central paradigm for artificial intelligence (Newell & Simon, 1976).

1. AI Programs as Theories of Discovery

By a discovery program I mean a computer program whose output is not inferable in any obvious way from its input. The phrase, "in any obvious way," is essential to the definition, since we know that a program does exactly what we program it to do -- which is usually not at all the same as doing what we supposed we had programmed it to do.

2. The Nature of Discovery

Novelty, in computers as in human beings, lies in the eye of the beholder. The result is novel if it was not expected from the outset. But even this definition is ambiguous. As the numerous documented cases of independent invention attest, a discovery may be novel to the discoverer but not to the whole society, for others may already have found it. However, to produce a novelty a second time, without knowledge that it has already been discovered by others, presumably requires the same kinds of cognitive processes as were required to produce it the first time. Anything we can learn by examining the program of the original discoverer we should be able to learn also by examining the program of the reinventor.¹

It is probably true today that within any one hour period some computer program somewhere in the world has followed a path never before traversed, to produce a novel successful result. This must occur for example, more than once in almost every game played by a hobbyist's minicomputer chess program, since chess games rarely fully repeat others that have been played in the world. However, we generally do not apply the term "discovery" to every novelty of this kind, however rational or adaptive the output may be. We require, in addition, that the novelty be in some sense remarkable or socially valuable. In particular, and borrowing language from the patent law, to be an invention, a novelty must not be "obvious to a person skilled in the art." While a minicomputer playing Class D chess discovers many novel solutions to its problems, these solutions would presumably be discovered easily by strong players, hence would not qualify as inventions in the legal sense.

Even today, after a quarter century of AI efforts, it is hard to point to fully convincing examples of discoveries by artificial intelligence programs that satisfy this stricter definition, of being neither rediscoveries nor obvious to one skilled in the art. If pressed on this point, I might want to defend certain products of chess programs, programs for musical composition and visual design, and theorem-proving programs as meeting the stricter requirements of invention, but such a defense would take

me away from my main concern here.

I will draw my examples of discovery from computer programs that have mainly rediscovered what was already known, but whose discoveries are hardly trivial, and would indeed have been adjudged important if they had been genuinely new. I have in mind such examples as the discovery by Lenat's AM program of the concept of prime number and its conjecturing of the fundamental theorem of arithmetic -- that every positive integer can be represented uniquely as a product of powers of primes (Lenat, 1977). Examples of a slightly different kind are BACON's induction from empirical data of Kepler's Third Law, Ohm's Law, and the Laws of Boyle and Charles (Langley, forthcoming).

Before we take the programs that found these concepts and laws as exhibiting the essential processes for discovery, we must satisfy ourselves on one point: that the human programmers did not, in some explicit or implicit way, embed the results at the outset in the programs and their inputs. Since we have already agreed that the outputs of programs are determined by the programs (and data), what can we mean by this requirement? Simply that the derivation of the outputs from program and data be sufficiently non-obvious. This is, of course, the same criterion we apply to a mathematical theorem to determine whether it is "deep"; and it is the same as the legal requirement for invention, quoted above.

This is not to say that it is a precise criterion, statable in a formal way. The only way I know to decide whether AM or BACON, or any other program purporting to have powers of discovery (but exhibiting those powers through rediscovery) genuinely possesses such capabilities is to search the code carefully for hideaways where the conclusions may be concealed in the premises. The severity of the test will depend on how thoroughly the search is made and how strict a criterion of obviousness is applied. Since I know of no way at the present time to quantify either of these two dimensions of the test, we must still depend (as we do in evaluating the merit of scientific discoveries) on informal judgement.

From close familiarity with the AM and BACON programs, I am satisfied that these two programs pass any reasonable tests of this kind. Let me, then, comment on the structure of the programs -- on the sources of their powers of discovery. For the sake of those of you who are not acquainted with AM or BACON, I will first state briefly what each program does. As already noted, a fuller description of AM, by Doug Lenat, will be found in the Proceedings of the Fifth IJCAI (1977), and of BACON, by Pat Langley, in the Proceedings of this conference.

Lenat's AM Program AM is a system that discovers new concepts and that conjectures new relations among them. Its input consists of an initial stock of concepts (in one application, the basic notions of set theory), goals and criteria (the goal of discovering new concepts and possible relations among concepts, and criteria for evaluating the worth or interest of concepts), and heuristics for searching for new concepts. Among the criteria for judging if a concept is interesting is how closely it is related to other interesting concepts, and whether examples of it can be constructed -- not too easily, but with not too much difficulty. The search heuristics include the advice to construct examples, to

¹This claim requires some qualifications. The reinventor may possess knowledge -- not the invention itself, but knowledge relevant to it -- that was not available to the original inventor, but which makes the job easier the second time. Later, I will have more to say on this point as it applies specifically to AI.

pay particular attention to borderline examples, to particularize when examples are found too easily, to generalize when they are hard to find. This is the kind of initial information available to AM -- initial concepts, goals and criteria, and search heuristics.

The control structure of AM guides it in a best-first search; the criteria of concept worth determine which of the concepts already attained should be the starting point for the next quantum of search. On its most celebrated run, starting with the concepts of set theory, AM discovered -- among other things -- the integers, the arithmetic operations of addition, subtraction, multiplication and division, the concept of prime number, and, as I mentioned earlier, the prime number representation theorem. When it began concerning itself with numbers possessing maximal (instead of minimal) numbers of prime factors, Lenat thought it had entered on truly new ground, only to find that this territory had earlier been explored by the self-taught Indian mathematician, Srinivasa Ramanujan. Therefore, AM must be evaluated as a rediscoverer, rather than a discoverer of new mathematical truths.

Langley's BACON Program BACON is a program that induces general laws from empirical data. Given sets of observations on two or more variables, BACON searches for functional relations among the variables. Again, it carries out a form of best-first search, in which a criterion of "simple things before complex" guides what to try next.

BACON's search is highly selective; it does not try all possibilities. It arranges the observations monotonically according to the values of one of the variables. Then it determines whether the values of some other variables follow the same (or the inverse) ordering. Picking one of these other variables, it searches for an invariant by considering the ratio (resp., product) of this variable with the original one. If the ratio (product) is not constant, it is introduced as a new variable, and the process continues. Thus, the newly defined variables in BACON correspond to the new concepts in AM, and the process is driven by a search for invariants.

It is easy to see how BACON, discovering that the product of electrical current by resistance in an electrical circuit was constant, would be led to Ohm's Law. The case of Kepler's Third Law requires BACON to generate, successively, ratios of powers of the radii of the planets orbits to powers of their periods of revolution, arriving at the invariant, D^3/P^2 , after a search of a small number of possibilities.

I have mentioned only a few of the salient features of BACON. The system has at least crude means for ignoring noise as data, and a number of other interesting features, but I will leave their fuller description to the program's author. What is interesting for our purposes is that a program, equipped and organized as I have described, detects regularities in data sufficiently perceptively to rediscover important scientific laws.

AM and BACON use similar schemes of memory organization. The ability to apply the same basic processes to given information and to newly generated concepts or variables, respectively, hence to operate recursively, is guaranteed by using a homogeneous format for the storage of all data. Though the details of

the data structures are different for the two programs, both use schemas -- structures of property lists -- to describe the objects with which they deal or which they generate. The main element of rigidity in their memory organizations is that the specific properties that may occur in these schemas -- the "slots" -- are specified in advance and known to the programs.

Discovery Mechanisms The theory of discovery that emerges from an examination of how these programs work contains little that should surprise us -- unless we have been seduced by the often-repeated myth that discovery processes, being "creative," somehow stand apart from the other actions of the human mind. In AM and BACON we see discoveries being produced by precisely the same kinds of symbolic processes that account for the efficacy of other AI problem-solving programs: theorem provers, chess players, puzzle solvers, diagnosis systems. A space of possible concepts and relations (AM), or of possible invariants (BACON) is searched in a highly selective, best-first manner. The search mainly works forward inductively from the given concepts or data.

The discovery programs are distinguished from most other problem-solving systems in the "vagueness" of the tasks presented to them and of the heuristic criteria that guide the search and account for its selectivity. Because the goals are very general ("find an interesting concept or relations, "find an invariant"), the use of means-ends analysis to work backward from a desired result is not very common. By and large, the programs work forward inductively from the givens of the problem and from the new concepts and variables generated from these givens.

Both programs work at a very concrete level. AM makes a major use of examples, which it is capable of generating, in searching for new concepts. BACON works with numerical data. If we observed human scientists working in the manner of these programs, we would regard them as very pragmatic. We are reminded of Faraday's notebooks, in which he recorded, day after day, the experiments that were suggested to a curious mind by the findings of the previous day's experiment. Or, we think of Mendeleev arranging and rearranging his lists of the elements until their periodic structure begins to emerge from his worksheets.

Both programs discover, they do not prove. Their task is to find regularity and pattern in nature, not to demonstrate the necessity of that pattern. Although their heuristics appear very general and weak -- they do not rely at all on semantic information about the task domain that is being explored -- they accomplish the search tasks with a remarkably small amount of trial and error. In the best tradition of heuristic schemes, they operate without any guarantees that they will succeed, but they do succeed in finding many interesting results. We would not even know how to define completeness for programs given these kinds of ill-defined tasks.

Because the tasks addressed by these systems are poorly defined, we do not have good measures of how powerful they are. Of course, we can make our personal evaluations of the quality of their discoveries -- of how impressed we are that AM finds the prime number factorization theorem, or that BACON readily induces Ohm's Law from the data. But we do not have the precision of comparison with human performance that a

chess program gives us, or a program for medical diagnosis. The difficulty of evaluating them is compounded by the absence of a yardstick for measuring the knowledge with which they are endowed at the outset, or that is embedded in the program structures. We do not know whether BACON had the same starting point as Ohm, or whether one of them was faced with an essentially simpler problem of induction than the other. Of course the same uncertainties surround all of our attempts to evaluate human discovery also. AM and BACON pose no new methodological puzzles in this respect.

How does the behavior of these programs compare with the behavior of the human scientists who have labored in the vineyards of artificial intelligence during the past 25 years? Do AM and BACON provide a true, if rough and approximate, description of that discovery effort? And what of the future of AI? Can these discovery programs help us in either prediction or strategy? Let me turn first to the history.

3. The Discovery Process in AI

Artificial intelligence has sometimes been criticized as being atheoretical, and consequently as having no solid substance. Of course, the premise might be true but the consequent false, unless we believe that all truth takes the form of rigorously proved theorems. Artificial intelligence has certainly been short of theorems, and in a field as densely populated with mathematicians and former mathematicians as is computer science, its nakedness in this respect has not gone unnoticed.

It may be objected that I am neglecting the A* algorithm, or the various interesting properties of Alpha-Beta search, or even the theorems that Kadane and I have proved about optimal evaluation functions for best-first all-or-none search (Simon & Kadane, 1975). But these isolated examples, even if we add to them all the others known to us, do not constitute a theory of artificial intelligence. At best, they provide us with some islands of theory, separated by wide expanses of an atheoretical ocean. Moreover, the heuristics of best-first search implied by these examples were known empirically and used in running AI programs for many years before the mathematics was developed (Newell & Simon, 1956).¹

AI as Empirical Inquiry I am afraid that we must resign ourselves to the fact (or celebrate it, depending on our taste in science) that artificial intelligence has not been a branch of mathematics, but rather a field of inductive, empirical inquiry. The main strategy of investigation has been to propose tasks requiring intelligence for their performance, to write programs for handling those tasks, and to test the efficacy and efficiency of the programs by giving them a sample of tasks drawn from the domain in question. Nearly everything we have learned about artificial intelligence over the past 25 years (and much of what we have learned about human intelligence as well), has been found by following this experimental strategy. And the body of knowledge that exists in AI today is

better described as a store of experimental data and inferences drawn from them than as a collection of mathematical truths.

But the process I am describing corresponds closely to the kind of process that is carried out by AM and BACON. We have seen that both programs are inductive and experimental -- even if the product of the former's efforts are mathematical constructs and conjectures, and of the latter's, postulated functional relations among numerical variables. Neither AM nor BACON proves anything. If they produce conviction, it is the conviction of the empirical scientist, relying on some postulate of the uniformity of nature, rather than the conviction of the mathematician, relying on the certainty of the laws of logic.

Inferring Principles From Programs The problem-solving tasks that AI research has addressed during the past 25 years, like the tasks addressed by AM and BACON, seem largely fortuitous -- targets of opportunity: theorem-proving in logic and group theory, the Eight's Puzzle and Missionaries & Cannibals, chess, Euclidean geometry, medical diagnosis, mass spectrogram analysis, speech recognition, parsing natural language, to mention a few. An assiduous historian could no doubt track down the reasons why each of these domains was attempted, but those reasons would not add up to a grand strategy for artificial intelligence. Probably the choice was neither much more nor much less considered than the choice of sweet peas and fruit flies as favored organisms for genetic research.

The true comparison is between these tasks, on the one hand, and the examples generated by AM or the data sets of BACON, on the other. The central inductive problem for AI has been to generalize from the performance of programs dedicated to individual tasks some principles (empirical principles, not necessarily theorems) about the mechanisms required for intelligent problem-solving behavior. How successfully this problem has been solved can be judged by assessing how far new AI programs make use of the heuristics and structural principles of the programs already in existence, and by examining the extent to which AI textbooks are organized in terms of general principles.

On both scores there is evidence of steady progress in AI. In the first decade or two, one can find a number of reinventions of general principles by investigators who were exploring different task domains (or even, occasionally, the same task domain). For example, best-first search apparently appeared initially, as already noted, as a component of one version of The Logic Theory Machine, disappeared in early versions of GPS, which tended to be oriented toward depth-first search, and reappeared in the MATER chess combinations program (Baylor & Simon, 1966). As another example, schemas appear in programs as early as 1956, but were subsequently reinvented and rechristened "templates" or "frames" (Minsky, 1975; and Simon, 1972). During the past five or ten years, however, the main structural components of AI programs have been identified, and a reasonably consistent vocabulary adopted for referring to them.

This gradual progress toward awareness of general principles is reflected by the textbooks in the field. Early textbooks were little more than collections of

¹The Logic Theory Machine, in 1956, already incorporated best-first heuristic search, while the Alpha-Beta heuristic is to be found in chess programs as early as 1958.

examples of more or less successful problem-solving programs. Beginning with Nilsson's book, Problem-solving Methods in Artificial Intelligence (Nilsson, 1971), some general threads of organization began to appear, and specific programs were not merely described, but were analyzed for their contributions to these threads. With all this progress, the contemporary books still reflect the pragmatic and empirical foundations of the field, and resemble textbooks in geology more than they resemble treatises in analytical mechanics.

Departures from the Discovery Model There is one respect in which the history of AI research departs significantly from the trace of a computer discovery program, for in the AI world, many lines of inquiry can be pursued simultaneously -- provided that the discipline is sufficiently well populated by researchers, and that the researchers are not too much driven by fads. Hence, when we try to interpret the annals as exemplifying best-first search, we must use that term loosely. To be sure, there was a period of several years during which attempts at theorem-proving nearly dominated AI research, and a more recent period when much of the inquiry was focused on problem-solving in knowledge-rich domains. When a topic like one of these seems to be progressing rapidly, it attracts much of the field's research effort, as would be true of a best-first search system. But other lines of investigation are never wholly dormant.

What has happened when the AI research strategy has departed from the discovery model? The most instructive examples are the cases where pragmatism was sacrificed to the demand for more theory and formal development. One such case is theorem-proving, where mathematical tastes have exercised greater influence than in most other AI task domains.

From the time of Hao Wang's early and successful program for proving theorems in the propositional calculus (Wang, 1960), most theorem-proving efforts have placed great emphasis on the completeness of their programs and upon employing elegant proof methods (e.g., natural deduction and resolution) from symbolic logic. Since completeness is most easily proved for breadth-first programs that do not use pragmatically constructed selective heuristics, the mainstream of research eschewed best-first search and heuristics that lacked guarantees of completeness.

When heuristics could be used that did not threaten completeness (e.g., set of support), they were adopted readily, but heuristics possessing this guarantee were not in sufficiently long supply to prevent the exponential explosion of search trees. The net result has been a general disillusionment with the progress of theorem-proving research, and a diversion of effort to other task domains within AI. Some exceptions can be found, of course. For example, in the impressive work of Bledsoe and his associates, we see exhibited a much more pragmatic attitude towards heuristics than has been characteristic of theorem-proving research in general (Bledsoe, 1977).

AI as a Residual Domain Some years ago, Allen Newell described artificial intelligence as the domain of weak methods, a description that still seems to hold (Newell, 1969). This is not because anyone prefers weak methods to strong. No one would solve a problem by heuristic

search if he thought that the simplex algorithm of linear programming would do the job. But strong methods apply only in domains that have sufficiently rich and smooth structure to support them. The simplex method works only in a problem space that is convex, bounded by linear inequalities, and with a linear criterion function to be maximized. The method exploits the mathematical structure of the space to home in on solutions in a relatively direct and straightforward fashion.

AM, and to a lesser extent BACON, are designed to work in spaces that have little regular structure, or which have structure that is initially unknown to the program. They use the weak methods of heuristic search for the same reason that artificial intelligence has used those methods -- because not enough was known, in advance, of the shape of the problem space for stronger methods to be used.

Similarly, attempts to derive measures of computational complexity for typical AI domains have not yet yielded much of a mathematical harvest. Proofs about the dependence of amount of computation, in the worst or average cases, upon problem size depend on knowledge of structural features of the problem domain, and where such structural features are unknown or absent it becomes difficult to obtain strong mathematical results.

Necessity should not be redefined as a virtue. Yet, it makes some sense to define artificial intelligence as a residual domain -- the domain in which it has not yet been possible to substitute powerful special-purpose techniques for weak methods. At any time that such techniques are discovered for a particular subset of problems, those problems are removed from the jurisdiction of AI to that of operations research or numerical analysis. But human intelligence, applied, for instance to the discovery of new knowledge, is not limited to working in orderly domains that have strong structure, and it is the task of AI to show how intelligence works, and even to complement its working, in less well structured domains.

4. From Past to Future

If we take AM and BACON as our models of the discovery process, then we should despair of making exact forecasts of where artificial intelligence research is likely to go in the next few years. For the discovery process illustrated by those programs is myopic, its best-first search responding to intimations of opportunity. Consequently, targets will continue to shift, as they have shifted in the past, to those task domains that exhibit from time to time, most promise of movement.

Allocation of Effort One important difference has already been noted between a discovery process programmed for a serial digital computer and the social discovery process of the AI community. That community is a parallel, rather than a serial, machine. With the increase in manpower that has been attracted to the field in the past five years, the prospects are now brighter than they were earlier for maintaining sustained research activity in a number of AI domains at the same time. At the present time, for example, a more or less continuous effort of several research groups is being devoted to chess programs, to natural language understanding, to visual pattern recognition, to medical diagnosis, and to various kinds of information retrieval tasks.

The fact that a computing system has modest parallel capacity does not, however, invalidate the main features of the best-first search model. The parallel capacity is still highly limited, and does not grow exponentially (at least not for long), as it would have to in order to avoid decisions about what part of the tree to search next. The effort allocation problem for a parallel, but not exponentially growing, system is merely a little less poignant than the problem for a strictly serial system. At any given moment, several branches that are most promising for exploration have to be chosen, instead of a single branch. Hence, with limits of both manpower and funding in AI, increased activity in some directions means decreased activity in others.

For example, research on speech recognition appears to have receded again to a relatively low level of activity with the termination of the special ARPA funding, as has AI research on robotics. (I will have a bit more to say about robotics research later, but will simply observe now that the current boom in industrial robotics is only tenuously connected with the main stream of AI research, and makes only limited use of AI methods.) Automatic programming has never reached the level of attention that its potential importance and centrality to AI would seem to justify. Theorem-proving -- and problem-solving in general -- appear to be attracting relatively little effort currently.

Judged in terms of the contents of the Proceedings of IJCAI5 (I don't have the corresponding numbers for the current conference), natural language is attracting the most attention in AI research, followed closely by vision and the representation and acquisition of knowledge. These three areas together accounted for about 60% of all the papers.

The Evaluation function From the shifts in allocation of research effort, we can draw some conclusions about the evaluation function that is used to guide the best-first search. But we must note carefully whose evaluation function it is. To those of us who have been working in AI, it is obvious that the shifts in emphasis among speech recognition, robotics, and automatic programming (these especially, but not exclusively) have been determined to a much greater degree by the judgments of funding agencies as to what kinds of work were more likely to lead promptly to practical application, than by the judgments of the researchers as to what lines of inquiry held the greatest promise for advancing fundamental knowledge.

In part, this vulnerability of the research agenda to genuine or imagined priorities for applications is the price that AI pays for being a "big science" field, dependent for its progress on the availability of expensive computing equipment. But some other big science fields -- for example, radio astronomy -- have attained considerable freedom in selecting their research goals, and we can only hope that AI can gradually acquire similar autonomy as the field becomes better established and the fundamental character of the phenomena it studies more widely understood.

If I were to contrast my own personal evaluation function with the function inferred from the actual present allocation of effort, I would be inclined to give considerably more attention to the domains of robotics

(that is, the AI aspects of robotics) and automatic programming than these areas are now receiving. Later, I will have a few words to say about the reasons for my preferences.

Common Themes One factor that mitigates the possible damage done by the whims of funding agencies and the fads of AI research itself, is that there is a considerable overlap in the basic problems encountered, and in the basic AI mechanisms required to solve those problems in all the task domains where AI research is carried on. Best-first search, for example, is a recurring theme, regardless of whether we are concerned with theorem-proving, chess playing, or robot planning. Similar problems of data representation, organization, and access must be faced in almost all task domains. Many tasks call for natural language capabilities of wider or narrower extent. The context-dependence of knowledge acquired through search, and the extrapolation of knowledge from one context to another is a recurrent theme. Because of these commonalities, progress in our understanding of any new task is likely to contribute substantially to progress for other, temporarily dormant tasks.

But these benefits of commonality will be realized only if we pay explicit attention to the transfer problem. The existence of multiple parallel research efforts in different task domains increases the danger that the same principles and mechanisms will be reinvented, perhaps more than once, by specialized investigators who are unaware of work going on outside their own narrow areas. As the AI research field grows and more investigators enter it, specialization will undoubtedly grow also (it has already), and the dangers of duplication will increase correspondingly.

Perhaps the most important preventive step against reinventing wheels is to define research goals not simply in terms of constructing programs that will perform specific tasks well, but in terms of using programs as examples and test beds for generating and illuminating general principles. Computer science has its roots in both scientific and engineering traditions. For the engineer -- at least the nonacademic engineer -- the device is the thing; the proof of his pudding is in how well the system he has designed works. For the computer scientist, the device (the program) is not an end in itself, but a means for testing whether particular methods and principles, incorporated in the device, perform the functions for which they are intended. Journal referees and reviewers of funding proposals can contribute much to the development of AI by insisting on these broader goal specifications for AI research projects.

There would also appear to be room in AI research for more generalists and theorists who would devote their attention to extracting general principles by comparative analysis of programs in different task domains. Of course such activity goes on at the present time, but perhaps it would be encouraged further if we did not restrict the term "theory" to formal, mathematical developments.

It might appear that I have fallen into a contradiction. Using AM and BACON as my models of discovery programs, I pointed out the futility of trying to predict the course of discovery. Now, only a few paragraphs later, I am expressing my views about the allocation of

effort. There is, in fact, no contradiction. In best-first search, choosing an evaluation function and using it to guide the allocation of effort is unavoidable. This does not mean that one can predict where the search will lead; but a well-chosen evaluation function can indicate the most productive points at which it can start. Let me offer a few illustrations.

Research on Robots One criterion of a promising task domain is that successful AI programs in the domain will rely on important components of intelligence that have not been much explored in other research. Robotry is a promising domain, because it takes us away from planning actions in simple worlds of the imagination -- where the consequences of our actions can be deduced precisely -- into planning actions in complex real worlds, where we must be prepared to readjust our estimates of the state of the world repeatedly as our actions fall short of or beside our intentions.

Methods for matching the predicted to the actual state of the world, and for correcting the former to reflect the latter, are fundamental to the success of systems that can survive in complex environments and particularly in environments where there is much uncertainty.

When I refer to robotry research, I have in mind something rather different from the development of industrial robots that is now burgeoning in a number of countries. Most industrial robots are being designed to carry out fairly restricted ranges of tasks in factory environments that are carefully tailored to the robots. Moreover, AI techniques have not played a prominent role in these developments, most of which come out of the tradition of engineering control theory.

In this application, the residual status of AI methods is again apparent. If an environment can be sufficiently smoothed and simplified, then the methods of servomechanism and control theory may provide the best means for designing flexible devices to operate in that environment. AI methods are likely to have a comparative advantage in rough and complex environments that have to be dealt with in their raw, natural form. For this reason, research on vehicles capable of locomoting autonomously on remote planets is probably more relevant to basic issues in artificial intelligence than is research on industrial robots that are to operate in factory environments. The former kinds of systems will have to be flexibly intelligent to a much higher degree than the latter.

However, I do not want to overstate the case. As the development of industrial robots goes forward, there is a need for strong capabilities in visual pattern recognition, a domain in which artificial intelligence concepts are likely to play a role of increasing prominence. The point of my example is that we don't simply want to seize on robotry as a task domain, but want to ask what aspects of robotry call especially for AI approaches, and what light is likely to be cast on general AI concerns by research focused on those aspects.

Automatic Programming A second domain I singled out as promising for AI research today is automatic programming. Here again, the general value of the research for advancing our basic understanding of artificial intelligence depends on how the problem is defined. I have especially in mind systems that would

take ill-structured and incomplete descriptions of a desired program (of the sort we would give as instructions to a human programmer), and transfer them into executable code. Automatic programming, so defined, is an excellent domain in which to experiment with the automatic design of problem representations -- a problem we must address if we are to extend AI further into ill-structured domains.

An additional reason why automatic programming tasks deserve high priority on the research agenda is that they offer excellent opportunities for work on natural language and knowledge representation. Research in the latter two fields has sometimes suffered from vagueness in the specification of the task. To study natural language effectively we must study particular kinds of situations in which information and meanings have to be communicated for a definite purpose. The automatic programming task defines that purpose (as does also the closely related task of understanding problem instructions written in natural language). By the same token, we are apt to learn most effectively about the problems of knowledge representation in the context of a specific task domain like automatic programming.

If we accept necessity as the mother of invention, we must remember that another parent is needed too. Automatic programming deserves a high rating for its research potential only if there is reason to believe it can be done -- that our basic knowledge has reached the point where it is reasonable to talk about automatic design of task representation. I would argue that both the progress -- modest though it be -- that has already been made in automatic programming, and the progress in the design of representations for other domains provide favorable indications that we are ready for the next step (Hayes & Simon, 1974).

Local and Global Knowledge A problem that has plagued heuristic search systems from the beginning is that information gathered at one node in a search through a problem space is not generally usable by the system to guide its search in other parts of the space. The same information may have to be generated again and again at different nodes.

Partly, this is a problem of information organization, solvable through such devices as blackboard schemes (Lesser & Erman, 1977). In such schemes, information is not stored in association with the nodes at which it is generated, but is placed in a common space where it becomes permanently available to all parts of the program, and at all times during the exploration of the problem space.

But there is a deeper problem with making information more broadly available: the information may be true only in a local context. Then the boundaries of this context must be determined and associated with the information before it can be exported safely. There is still not much theory (or experience) in the AI literature as to how this is to be done, but some progress has been made toward solving the problem in connection with research on speech recognition programs and chess programs, both of which are promising environments in which to pursue this issue (Lesser & Erman, 1977; Perdue & Berliner, 1977).

Learning Systems In AI a great deal more progress has been made in constructing performance programs than in

designing programs that learn. In the early history of artificial intelligence, the topic of self-organizing systems was pursued vigorously but, as it turned out, not particularly successfully. As the best-first search progressed, the nodes associated with this topic received low evaluations, and were gradually abandoned.

Yet the topic of learning in AI is not at all dead; rather it has been redefined. In early efforts, great importance was attached to starting systems off at or near ground level. The guarantee that they were learning was that they started off knowing almost nothing. Today, we characterize learning in a somewhat different way; we look for adaptive change, and we look for that change to be recursive and cumulative.

In the broadest sense, any program is a learning program that gradually changes over time so that on each new encounter with a particular kind of task it behaves in a more appropriate way. In neither human beings nor computers should we expect to find just a very limited number of processes called "learning processes," for there generally are a multitude of ways in which a complex system can modify itself adaptively.

Learning will generally be incremental. That is, each new step in adaptation will itself improve the capacity for further adaptation. A problem-solving system becomes a learning system whenever it is designed so that problem solutions can be stored and used to contribute to subsequent problem-solving. Clearly, discovery programs like AM and BACON are learning programs, since their explicit task is to produce novel outputs and to use those outputs recursively.

With this broader definition of learning, a whole spectrum of AI systems qualify as learning systems. Learning can connote all degrees of passivity or activity of the learner. Thus, at one extreme, we have interactive systems aimed at making it easier for the programmer to add new knowledge to an information structure, where the program itself is a wholly passive learner. At the other extreme, we have adaptive production systems that are able to extract information from their experiences, and use the information to improve themselves even without explicit instruction from outside. Most systems that learn from experience are aided, of course, if the experience is organized for them in a favorable way -- in a succession of carefully graded lessons. It is the skill of a good teacher to present experience in this way.

One might ask whether it is time to revive learning as a major explicit goal of AI research. Since learning pervades almost all aspects of intelligent performance, the right search strategy is probably to incorporate learning goals in our performance systems. That seems to be a quite natural thing to do in building systems for visual pattern recognition, for example, for automatic programming, or for understanding natural language instructions.

But there are apprentices in the world as well as journeymen; and presumably the apprentice's first concern is his learning rather than his performance. So perhaps there is room, on the tree of AI research, for an active branch that works with tasks in which learning and adaptation are the central concerns. Considering the recent rapid progress that has been made in constructing adaptive production systems, good progress can be

anticipated along that branch, and I would assign it a rather favorable evaluation. But the fact that some investigators specialize in learning processes should not deter the rest of us from experimenting with learning components in our performance systems.

5. Conclusion

In this paper I have reviewed the AI community as if it were a medium-size slightly parallel processor searching its way in inductive, best-first fashion through the problem space of intelligent action. I have compared it with some of the existing AI programs that best characterize the discovery process. The comparison does not yield any great surprises, but perhaps provides some reassurance.

As a typical example of a discovery program, the AI community uses weak methods under the guidance of a somewhat imprecise evaluation function and vague ultimate goals. It tries to discover the mechanisms that enable a system like the human mind to behave purposefully, adaptively, and sometimes even effectively over a wide range of difficult and ill-structured tasks.

The search is highly pragmatic, steered and redirected by concrete empirical evidence culled from experiments with programs operating in an accidentally determined collection of task environments. The output of the research is mostly encapsulated in heuristics, not yet formalized in coherent theories of broad scope. All is confusion and mild chaos, as it should be at an exciting frontier of fundamental scientific inquiry. Although only a quarter of a century old, the search has already yielded a solid body of empirical knowledge about the nature of intelligence and the means of capturing it in programs.

6. References

- Baylor, G.W., & Simon, H.A. A chess mating combinations program. AFIPS Conference Proceedings, 1966 Spring Joint Computer Conference, Boston, April 26-28, 1966, 28, 431-447. Washington, DC: Spartan Books.
- Bledsoe, W.W. Non-resolution theorem proving. Artificial Intelligence, 1977, 9 (1), 1-35.
- Hayes, J.R., & Simon, H.A. Understanding written problem instructions. In L.W. Gregg (Ed.), Knowledge and cognition. Potomac, MD: Lawrence Erlbaum Associates, 1974.
- Langley, P. Rediscovering physics with BACON.3. Proceedings of the 6th International Joint Conference on Artificial Intelligence, forthcoming.
- Lenat, D.B. The ubiwyty of discovery. Proceedings of the 5th International Joint Conference on Artificial Intelligence, 1977, 1, 1093-1105.
- Lesser, V.R., & Erman, L.D. A retrospective view of the HEARSAY-II architecture. Proceedings of the 5th International Joint Conference on Artificial Intelligence, 1977, 2, 790-800.
- Minsky, M. A framework for representing knowledge. In P.H. Winston (Ed.), The psychology of computer vision. NY: McGraw-Hill, 1975.
- Newell, A. Heuristic programming: Ill-structured problems. In J. Aronofsky (Ed.), Progress in operations research, Vol. III. NY: Wiley, 1969.
- Newell, A., & Simon, H.A. Computer science as empirical inquiry: Symbols and search. Communications of the ACM, 1976, 19, 111-126.
- Newell, A., & Simon, H.A. The logic theory machine. IRE Transactions on Information Theory, Vol. IT-2, No. 3, Sept. 1956.
- Niisson, N.J. Problem-solving methods in artificial intelligence. NY: McGraw-Hill, 1971.
- Simon, H.A. Artificial intelligence systems that understand. Proceedings of the 5th International Joint Conference on Artificial Intelligence, 1977, 2, 1059-1073.
- Simon, H.A. The heuristic compiler. In H.A. Simon & L. Siklossy (Ed.), Representation and meaning: Experiments with information processing systems. Englewood Cliffs, NJ: Prentice-Hall, 1972.
- Simon, H.A., & Kadane, J.B. Optimal problem-solving search: All-or-none solutions. Artificial Intelligence, 1975, 6, 235-248.
- Simon, H.A., & Newell, A. Heuristic problem solving: The next advance in operations research. Operations Research, 1958, 6, 1-10.
- Wang, H. Toward mechanical mathematics. IBM Journal of Research and Development, January 1960, 2-22.