ELSEVIER

Artificial Intelligence 77 (1995) 95–127

Artificial Intelligence

Artificial intelligence: an empirical science

Herbert A. Simon*

Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Received August 1993; revised May 1995

Abstract

My initial tasks in this paper are, first, to delimit the boundaries of artificial intelligence, then, to justify calling it a science: is AI science, or is it engineering, or some combination of these? After arguing that it is (at least) a science, I will consider how it is best pursued: in particular, the respective roles for experiment and theory in developing AI.

I will rely more on history than on speculation, for our actual experience in advancing the field has much to tell us about how we can continue and accelerate that advance. Many of my examples will be drawn from work with which I have been associated, for I can speak with greater confidence about what motivated that work and its methods (and about its defects) than I can about the work of others. My goal, however, is not to give you a trip through history, but to make definite proposals for our future priorities, using history, where relevant, as evidence for my views.

1. Artificial intelligence as science

AI deals with some of the phenomena surrounding computers, hence is a part of computer science [27]. It is also a part of psychology and cognitive science. It deals, in particular, with the phenomena that appear when computers perform tasks that, if performed by people, would be regarded as requiring intelligence—thinking.

Artificial intelligence began in the 1950s as an inquiry into the nature of intelligence. It used computers as a revolutionary tool to simulate, indeed exhibit, intelligence, thereby providing a means for examining it in utmost detail. "B.C.", before computers, the only observable examples of intelligence were the minds of living organisms, especially human beings. Now the family of intelligent systems had been joined by a new genus, intelligent computer programs.

* E-mail: has@a.gp.cs.cmu.edu.

SSDI 0004-3702(95)00039-9

1.1. The multiple goals of AI

As the papers reprinted in Feigenbaum and Feldman's classic, *Computers and Thought*, reveal, the AI thrust, from its very beginnings, was at least three-pronged. One goal was to construct computer programs (e.g., the Logic Theorist) capable of exhibiting intelligence, and thereby, to begin building a theory of intelligent systems. (The original Carnegie-Rand name for the endeavor was "complex information processing"; our group later accepted the alternative "artificial intelligence", which had become the established usage among both friends and foes of the activity.)

A second goal was to construct programs (e.g., GPS) that exhibited intelligence by using processes like those used by humans in the same tasks. Here the aim was to achieve a theory of how the human mind can behave intelligently. The third goal was to construct intelligent programs (e.g., Tonge's assembly line balancing program) that could supplement or complement human intelligence in performing some of the world's work. (In the body of this paper, I will refer to systems in this third category as "expert systems", enlarging somewhat the usual denotation of that name.) The systems described in Sections 2 and 3 of Part 1 of *Computers and Thought* focus upon the first of these three objectives; those described in Part 2, upon the second; and those in Sections 4–6 of Part 1 upon the third.

Almost from its very birth, then, AI was a multicelled organism. Its foundation was the capability for building systems that exhibited intelligence, either as pure explorations into the nature of intelligence, explorations of the theory of human intelligence, or explorations of the systems that could perform practical tasks requiring intelligence. Surrounding these operative AI systems there gradually grew up corresponding bodies of theory. But we should not think of the programs as isolated from the theories. Quite the contrary. For example, the Logic Theorist *embodies* a theory: the theory that achievement of intelligence in solving problems requires a physical symbol system capable of heuristic search. likewise, GPS embodies the theory that means–ends analysis is a powerful heuristic commonly employed by people for problem-solving search.

We can extract from such programs verbal statements of the theoretical principles, as Allen Newell and I did in our 1975 Turing Award address, but the programs provide the basic operational definitions of what the principles mean. The theory is no more separable from the program than classical mechanics is from the mathematics of the laws of motion. Different implementations of the verbal statements are different theories, which exhibit different properties when the programs are run. I will have more to say later about the relation of programs to theories and how runs of programs are used to test theories. For the moment, I will simply remind you that:

> The moment of truth is a running program.

1.2. Social fragmentation of AI

Side by side with the growth of programs aimed at the triple objectives of understanding intelligence, understanding the human mind and building and

understanding expert systems, there have grown up communities of researchers concerned with these objectives. Some of the researchers are interested in more than one of the three goals, a few with all three; but social structures have formed that emphasize and enforce the separateness of the three endeavors rather than their common concerns.

I will not describe these social structures in detail, except to mention that, in fact, there are four, not three, principal groups; for at least two rather distinct groups of researchers focus upon the "pure" theory of intelligence. One subgroup, found mainly in computer science departments, is often associated with colleagues who are interested in program verification and/or computational complexity. Another subgroup identifies with "cognitive science"; some of its members are to be found in psychology, some within AI groups in computer science, some in philosophy, some in linguistics, some in anthropology and miscellaneous other areas.

Over the years, the distance separating all four AI groups has gradually increased. They attend different professional meetings, ranging from the American Psychological Society, through the Cognitive Science Society, the AAAI, to ACM and engineering societies like IEEE. They limit their reading and citations more and more to the journals published by their groups. They receive their training in different academic disciplines and subdisciplines, each passing on to the next generation its own specialized version of the enterprise.

In assigning a broad definition to AI, covering all of these groups, I reveal my belief that, in spite of diversity of goals, there is a common core that makes continuing communication among them highly desirable. I believe that each one of these groups can get substantial help in advancing its goals by drawing on the work of the others, and that the advantages of interaction (and the serious disadvantages of fragmentation) have been frequently demonstrated over the whole history of AI. I believe that AAAI and the Cognitive Science Society share primary responsibility for opposing and turning back the forces of dissociation.

My reasons for believing in complementarity of the several goals will emerge as I proceed. I would just like to state two of the reasons now, in a preliminary way.

First, the way in which humans achieve intelligence (I will call it "heuristic search") is quite different from the way in which computers performing numerical analysis and similar tasks typically do it. (We might call the latter method, "brute force disciplined by mathematics.") The distinction is not black and white, but obvious none the less. The human methods, I believe, are absolutely essential for intelligent response to relatively ill-structured problems, hence we must understand them regardless of whether our aim is to understand the human mind or intelligence in general.

Programs for solving linear programming problems, inverting matrices or solving partial differential equations depend heavily on the size and speed of computers, but use algorithms based on the known, and usually rich, mathematical structures of their problem spaces to reduce the amount of search required. These search reduction principles do not always optimize search, but almost always preserve the property of completeness—they are guaranteed to find the solution to any desired degree of approximation. Because of the mathematical

regularity of the spaces searched, it is often possible to prove theorems about the sufficiency and sometimes the efficiency, of algorithms.

Human problem solving seldom shares any of these properties. Nevertheless, humans, whose computational abilities are puny compared with those of modern super-computers or even PCs, are *sometimes* able to solve, with little computation, problems that are very difficult even by computer standards—problems having ill-defined goals, poorly characterized and bounded problem spaces, lack of strong and regular mathematical structure. People solve such problems by the shrewd use of heuristics and at the expense of giving up guaranteed completeness of search and optimality of the solutions attained.

What I am calling "disciplined brute force" had its origins in numerical analysis and its successor, the theory of computation, which has had some extension from numerical to symbolic systems. Many of us do not believe that the methods of disciplined brute force can achieve the same range of application and flexibility that humans attain. Unless and until it is demonstrated that they can, we have every reason to explore vigorously the human heuristic search techniques as a source of ideas for intelligent systems, and equal reason to understand the underlying mechanisms that give these techniques their power in situations where brute force, even disciplined brute force, fails. These concerns are central both to understanding how intelligence, human or not, can be applied to ill-structured problems, and to constructing expert systems for solving such problems.

Artificial intelligence has always had a special interest in this important "residual" area, where programs rely on heuristic search, without guarantees of completeness, and often use satisficing criteria of success. Given this interest, they have every reason to keep in close touch with the progress of research in human intelligence, which is gradually demystifying the nature of human "shrewdness", "intuition" and even "creativity".

A second reason for close communication among all of the sub-strands of AI is that the principal means of progress in our field is to find tasks requiring intelligence for their performance, and then to see what kinds of processes are sufficient to perform these tasks. Over the years, exploration of a steadily widening domain of tasks has disclosed a continually richer array of mechanisms for intelligent behavior. And as the great diversity of existing computer programs that play chess illustrates, even for a single task there may be many ways to skin the cat. Comparison of alternative programs can cast much light on the underlying principles of intelligence. Human behavior provides a valuable range of difficult ill-structured tasks where the peculiar characteristics of human intelligence are regularly exhibited.

2. Artificial and natural objects

Artificial objects, including computer programs, are what they are because they were designed to be that way. This fact has led some to claim that there can be no

science of artificial objects, but only an engineering technology. Those who hold the most extreme form of this view look to the discovery and proof of mathematical theorems about intelligent systems as the only genuine route to a science of AI, and denigrate the role of system building and experiment as "only engineering".

But the claim that artificial objects are divorced from empirical science, and that they do not lend themselves to natural-science methods of research is fallacious. An artificial object is as fully bound by the laws of nature as any natural object. Automobiles are as subject to the law of gravity and the conservation of energy as glaciers are. Scientific laws limit the set of possible objects, natural or artificial. No object, artificial or natural, that does not obey these laws—satisfy these constraints—can exist [36, Chapter 1].

2.1. *Internal constraints*

The natural laws that determine the structure and behavior of an object, natural or artificial, are its *internal constraints*. An artificial system, like a natural one, produces empirical phenomena that can be studied by the methods of observation and experiment common to all science.

It might be objected that a system designed deliberately to behave in a desired way can produce no surprise or new information. This objection shrugs off our enormous ignorance of natural law and of the effects produced by natural laws operating on complex systems. The world of artificial (and natural) objects is full of unanticipated consequences, because of the limits both of empirical knowledge and of computational power. The case, in AI, for studying many different kinds of systems empirically is essentially identical to the case, in biology, for studying many species of organisms. In neither case can we capture more than a miniscule portion of the richness and complexity of the real world by attempting to deduce it from first principles.

Often the most efficient way to predict and understand the behavior of a novel complex system is to construct the system and observe it. Because AI programs are also computational models, we can use the programs themselves as their own models, an advantage for the field of AI that is unique in science. In AI, the theory not only *models* but simultaneously *exhibits* the behavior of the phenomena under study.

The "natural" sciences also depend, for their progress, on building artificial systems and studying their behavior, for this is the essence of the experimental method. The natural scientist constructs a system in which the operation of certain natural laws is thought to be prominent; then observes the phenomena produced by the system, and especially how these phenomena change with changes in the system parameters. So Galileo rolls balls down inclined planes or over the edges of tables, and measures the time of the roll or the length of the flight as a function of the angle of the plane.

To experiment is to use the artificial to study the natural. To design an AI

system and observe how its behavior changes with changes in the design is to perform an experiment. Most of what we know about artificial intelligence has been learned by carrying out experiments of this kind, thereby making AI a thoroughly experimental science.

2.2. External constraints

A system, artificial or natural, must conform not only to the internal constraints imposed by natural law, but to two sets of external constraints as well. The system can only come into existence under conditions that are defined by natural law; and it can only survive and operate effectively in suitable environments. These initial and boundary conditions are the system's *external constraints*. Synthetic chemistry—like AI, a science of design—is devoted to determining (by actual synthesis) the external constraints that operate upon chemical molecules.

By manipulating the external constraints, we can often determine what functions a system must perform in order to survive, and how its various components carry out these functions. Nature, according to Darwin, generates systems or modifies existing ones; then tests their ability to survive in the ambient environment. The artificer does exactly the same thing, except that the generator (the design process) is more purposeful, and the tests (the purposes the designer has in mind) may go beyond biological fitness.

As the functional requirements imposed by the environment introduce a teleological component into all systems in the same way that the designer's purposes do, the difference between the natural and the artificial fades and then vanishes. The constraints imposed by nature on living organisms derive from the same natural laws as those that face design.

2.3. Science and engineering

We see that, far from striving to separate science from engineering, we need not distinguish them at all. But if we insist upon a distinction, we can think of engineering as science for people who are impatient. The Darwinian processes of biology depend on the chance of mutations and crossover to produce new designs. Although there is also a large element of chance in human design processes, chance is moderated by heuristics that use prior knowledge, what is already known about the systems of interest, to generate and combine elements in a very selective way, greatly increasing the odds that the product will be functional.

While the scientist is interested specifically in creating new knowledge, the engineer is interested also in creating systems that achieve desired goals. Apart from this difference in motives, there is no need to distinguish between computer scientists and computer engineers, or AI scientists and engineers. We can stop debating whether AI is science or engineering; it is both.

3. Research by system synthesis

It is time to relate these generalities to the discipline of artificial intelligence. Regardless of our reasons for pursuing AI, its main research method is to build and study systems that exhibit intelligence. The basic paradigm is to:

> Select a task incorporating a feature of intelligence that is of substantial practical importance or that exhibits features and complexities that have not yet been simulated by AI systems. Build a system exhibiting this feature of intelligence. Examine the behavior of the system in different task environments and with different initial conditions.

The Logic Theorist incorporated some simple methods of heuristic search, which were tested on the task of discovering proofs for theorems. GPS incorporated means–ends analysis, which was tested in a variety of simple problem-solving domains. SHRDLU [41] had means for processing natural language strings and extracting their semantic meanings, which were tested in a blocks world. EPAM [11] has mechanisms for recognizing, remembering and learning to discriminate, which were tested in a range of experimental settings drawn from the research literature on verbal learning. NAVLAB [30] has mechanisms for determining the position of a vehicle and steering it, which are tested by driving it on roads. In what sense are these kinds of design projects experiments?

3.1. System design as experimentation

An experiment manipulates the independent and dependent variables of a particular system. What are the independent and dependent variables in an AI system? The dependent variables clearly are measures of the performance of the system: how intelligently it behaves both in terms of the range of tasks it can handle and its skill and efficiency in handling them.

Defining the independent variables takes a little more care. Suppose that we are studying a particular version of GPS. First, there is the core of the system: in the case of GPS, principally its basic symbol-processing capabilities and its mechanism for means–ends analysis. A little more peripherally, it also contains strategies for heuristic search; it may incorporate best-first search, for example, or depth-first search. Still more peripherally, it contains information about particular task domains, including productions that notice differences between situations (current situation and goal situation) and productions that select move operators relevant to reducing the differences that are noticed.

Changes in any or all of these components can be regarded as changes in the internal constraints on GPS. Or, we can think of the symbol-processing capabilities and the means–ends mechanism as internal constraints, and the remaining components as initial conditions. Both internal constraints and initial conditions can be treated as independent variables for experimental purposes. In addition,

the task environments with which we confront GPS define the external constraints on its behavior, constituting another set of independent variables.

In the earliest experiments in AI, with systems like LT, GPS, STUDENT and the others that are reported in *Computers and Thought*, task domain and domain knowledge were held constant, while the principal independent variables were the core of the system itself, and often its strategies. The question to be answered was: "What basic symbolic capabilities and heuristics will enable a system to exhibit intelligence in a task domain that is difficult for humans?"

Procedures for evaluating outcomes were not elaborate. Did the system solve the problems with moderate computing effort? Did it behave selectively, in comparison with a brute-force search? At what level of problem difficulty could it operate (as compared, say, with human skills)?

Today, when an artificial intelligence project is aimed at extending AI to a new class of task domains, matters remain much the same. The BACON system [21] for scientific discovery takes a set of data from an experiment. BACON contains basic symbol-manipulating capabilities and a small set of heuristics for inducing laws from data and inventing new theoretical concepts. Experimentation consists in exploring the range of tasks over which it can and can't discover the regularities in data, the reasons for its successes and failures (i.e., the relation between its capabilities and the characteristics of the corresponding task environments), and the degree of selectivity of its search.

In such a line of experimentation, whether with GPS or BACON, initially, the principal independent variable is the core of the system itself and its strategies. What changes in the system will improve its performance in a task, and what changes are required to handle new tasks? As system performance improves, emphasis may shift from manipulating the characteristics of the system to testing a fixed system over a range of task environments. How flexible and general is the system?

In research within the so-called expert–novice paradigm, on the other hand, the initial conditions, that is, the system's domain knowledge, is the central independent variable. The main interest is in learning how much knowledge, organized how in memory, is needed for expert performance. In research on generality, the task domain is the central independent variable. Is there a small core of mechanisms that can carry the main part of the load over all the tasks?

Extendability

Two themes are visible in much AI experimentation. One is extendability; the other is generality over tasks. New ideas in AL are often tried out on "toy tasks"—tasks that are better structured and less difficult than the real-life tasks we would like to handle. The Tower of Hanoi is a toy task; medical diagnosis is a real-life task.

Learning a language is a real-life task; but we may build an AI system, for example, Siklóssy's ZBIE [34], that, while demonstrating its ability to acquire simple syntax and semantics, for one reason or another is not yet ready to handle

the full scope of a natural language. We illuminate a real-life task by simulating a toy subtask.

We may wish to consider ZBIE a candidate theory of language learning, but our confidence in its veridicality will depend on the prospect of extending its capabilities to acquiring a complete natural language. If ZBIE's limits appear to be due to the short time it has had for learning or to physical limits on memory size, we will be more sanguine about its extendability than if we can see that additional or different mechanisms will be needed for the extension. Of course the final test of its strength as a theory will be the actual attempt to extend it.

We should remember that most theories in physics are only tested in relatively simple laboratory situations, and in fact, many important phenomena can only be observed clearly under highly controlled conditions. So we must be careful not to impose in AI requirements for theory verification much stronger than those imposed in other sciences. That would surely guarantee that we would never reach beliefs about anything significant.

One sound reason for caution about the upward scalability of programs that handle toy subtasks successfully is that in AI we are always faced with the specter of combinatorial explosion of search. But as we have gained confidence in our ability to build and use large knowledge bases to increase selectivity in programs like DENDRAL [23] and INTERNIST [31], and as we have succeeded in building increasing numbers of systems that operate at (human) professional levels of performance, the specter becomes less menacing. And as I will discuss later, such warnings as NP-completeness do not threaten combinatorial explosion for most of the problems we actually seek to solve.

Generality over tasks

AI is most interested, as it should be, in discovering those mechanisms of intelligence that apply to a wide range of tasks. GPS was designed to separate the task-independent from the task-dependent components of the program, and Ernst and Newell [8] undertook an extensive research activity to demonstrate that it could solve problems in a dozen or more environments without alteration of the task-independent component of the program. The theory of problem solving that GPS represents is that component of the program.

Similarly, the EPAM program [11] contains a core of mechanisms for recognition, memory and learning. To perform any task within its capabilities, it must acquire (or be given) an appropriate body of knowledge, stored as initial conditions in its memory, as well as strategies derived from the task instructions. It is primarily the core mechanisms that we regard as the EPAM theory; and it is these mechanisms that should remain invariant as EPAM is extended to new tasks.

However, it should not be supposed that the theoretical content of programs is limited to their cores. Knowing how much knowledge, and what kind of knowledge, is required by a program to extend it to real-life tasks is also an important part of AI theory. It would be of enormous interest today to know what knowledge, how organized, would be required for a chess program to play at

grandmaster level without needing to search more (100 branches?) than a human grandmaster.

Research within the expert–novice paradigm has focused specifically on determining the knowledge bases required for high level performance. Artificial intelligence is concerned with understanding both those general heuristic processes (and other bases for intelligence) that are applicable to many domains and those more specialized processes that permit high levels of performance to be reached in particular domains.

3.2. The physical symbol system hypothesis

The first task of AI research was to determine whether intelligent behavior could be obtained at all with symbolic list-processing systems. The repeated successes that the field achieved in a variety of task domains led Al Newell and me, in our 1976 Turing Lecture, to offer a hypothesis to explain this strong common foundation of the whole gamut of intelligent devices and programs. We called it *The Physical Symbol System Hypothesis*:

> A physical symbol system (PSS) has the necessary and sufficient means for general intelligent action.

As the hypothesis is a familiar one, I need not recount in detail the defining characteristics of a physical symbol system. A PSS is simply a system capable of storing symbols (patterns with denotations), and inputting, outputting, organizing and reorganizing such symbols and symbol structures, comparing them for identity or difference, and acting conditionally on the outcomes of the tests of identity. Digital computers are demonstrably PSSs, and a solid body of evidence has accumulated that brains are also. The physical materials of which PSSs are made, and the physical laws governing these materials are irrelevant as long as they support symbolic storage and rapid execution of the symbolic processes mentioned above.

The PSS Hypothesis asserts that the external constraints imposed by any task requiring intelligence can be satisfied by, and only by, a PSS. Since different tasks impose quite different constraints, the claim that being a PSS is necessary and sufficient for intelligence may seem surprising. Its truth depends essentially on the generality and adaptability of PSSs like computers and brains. Of course we are speaking here not of mathematical truth (e.g., Turing computability) but of the empirical fact that computers and brains, appropriately instructed, can exhibit intelligence over a wide range of tasks, *employing only acceptable amounts of computation to do so*.

There is some dispute today about the Physical Symbol System Hypothesis, hinging on the definition of the term "symbol". If we define "symbol" narrowly, so that the basic components in connectionist systems or robots of the sort advocated by Brooks are not regarded as symbols, then the hypothesis is clearly wrong, for systems of these sorts exhibit intelligence. If we define symbols (as I have, above) as patterns that denote, then connectionist systems and Brooks' [2]

robots qualify as physical symbol systems. In any case, the hypothesis is an empirical one, whose fate will continue to be decided by empirical evidence about the mechanisms employed by systems that exhibit intelligence, regardless of where we draw the definitional boundary of "symbol".

4. Theories of intelligence

Putting aside now the design task of developing specific intelligent systems, we turn to the task of developing the theories of those systems together with theories of the design process. I must preface what I am going to say with a discussion of what the term "theory" means, or should mean.

4.1. What is a theory?

There is an unfortunate confusion, encouraged by the similarity between the words "theory" and "theorem", between theories in an empirical science, on the one hand, and formal deductive theories, on the other. This confusion probably originated with, and certainly has been encouraged by, the great success of Newtonian mechanics in getting much from little. From a few basic premises, in particular the three laws of motion, all sorts of important consequences are derived mathematically about how matter behaves in the real world.

In many minds, this success has created the illusion that physics is nearly a branch of mathematics (recall the innumerable textbooks on *Rational Mechanics*); and has created strong urges in the other sciences to emulate this royal road to empirical truth by reasoning. Economics provides perhaps the most flagrant examples of the use of logic unfettered by observation to reach unwarranted conclusions about the real world, but examples are not absent from the other sciences, including computer science.

Very little of the physics of complex systems (the atmosphere, the ocean, condensed matter) has this highly deductive flavor. Mathematics there is, in generous portions, but it is surrounded by boundary conditions and initial conditions that are grounded in empirical observation. It is sometimes forgotten that special relativity theory was motivated by anomalies of observation—especially the incompatibility between the Galilean invariance of the laws of mechanics and the Lorentz invariance of Maxwell's laws. Similarly, Planck's law of black-body radiation was motivated by the failure of Wien's law to explain the intensities of spectral lines in the face of new observations of infra-red radiation obtained when bolometers were extended into that spectral range. Carefully observed phenomena are still the starting point for theory in physics.

When we turn from physics to sciences like biology and geology, and even chemistry, the priority of observed phenomena over conclusions reached via long chains of inference from general axioms becomes even more evident. Not only do most of the known regularities in these sciences derive from extensive observation and experimentation, but many of the regularities, especially the most important,

are not quantitative, but qualitative. In our Turing Address, Al Newell and I called such generalizations *laws of qualitative structure* (QS laws). The germ theory of disease, we observed, is such a qualitative and inexact law, as is the cell theory, and for that matter, the theory of evolution by natural selection. If there are any equations in "The Origin of Species", they are exceedingly inconspicuous.

The germ theory of disease says something like: "If you diagnose a disease, look for a microorganism (of course, you won't always find one)." The cell theory says something like: "Most organisms are made up of one or more (perhaps many more!) membrane-bounded structures called 'cells' that are remarkably similar across species in basic structure, for example, all having nuclei (actually, of course, only eukarytes have them)." Both theories are qualitative, approximate, even vague. These central theories soon become surrounded by crowds of particulars, of vary degrees of precision and generality, describing laws of organization and process. Most of these are QS laws; relatively few are quantitative.

When we are dealing with complex systems, whatever the science, theories almost all have this kind of complexity and messiness. To some extent they can still be modelled mathematically—or portions of them can if their application is limited to simple cases. For even smaller, simpler subsystems, the mathematical formulations can sometimes be solved in closed form. More often, the behavior of complex systems has to be studied by computer modelling and simulation, with little or no help from theorems. Even in the more theoretical portions of physics today, problems are seldom solved in closed symbolic form; more often they are solved numerically with many hours of computing; and physicists are the world's largest users of supercomputers.

Because of these properties of complex systems, a term that describes essentially all the AI systems of interest, we find that in AI the principal theories take two forms, which at first view seem diametrically opposed: there are precise theories in the form of computer programs, and fuzzier theories of the form that Allen Newell and I dubbed "laws of qualitative structure". Let us examine each of these in their application to AI.

4.2. Computer programs as theories in AI

In the physical sciences, systems of differential equations provide *the* major tool for expressing precise theories of system behavior. For predictions in any given situation, the differential equations must be supplemented by empirical estimates of system parameters and initial and boundary conditions.

In cognitive science, computer programs, which from a formal standpoint are simply systems of difference equations, perform exactly the same role as systems of differential equations do in physics. The only distinction between differential and difference equations is that the former treat time as a continuous variable while the latter treat it as a discrete variable—the system changes state with each computation cycle. Since the basic time interval represented by the system can be set to any value, this is a distinction without an important difference. (In fact,

when we make numerical computations on differential equation systems, we routinely approximate continuous by discrete time.)

Simple systems of differential and difference equations in real or complex numbers can be solved in closed form to yield general theorems of system behavior for any values of the parameters. As we have seen, when matters get a little more complex, or when the symbols in the equations are not numerical, solutions in closed form can no longer be obtained, and the system is studied by carrying out simulations for various values of the parameters. As the results apply only to the particular parameter values used in the simulations, we return again to the kinds of qualitative generalizations that characterize all complex systems.

In interpreting programs as theories, we must take care to define what characteristics of the programs represent the theory, what characteristics are to be regarded as irrelevant "notation", and what parts constitute boundary conditions and initial conditions for a particular application of the theory. These same questions arise in natural science theories, but perhaps take a particular form in AI that is worth examining.

In the earlier discussion of experimentation, we saw that a running AI program contains a definition of the goal and knowledge about the task domain as well as problem-solving processes. It also contains strategies, some of which may be task-specific. When we say that GPS is a theory of problem solving, we are speaking of the core program, including at least some of the more general, and task-independent strategies.

For example, the EPAM program is a theory of human perceptual and memory processes [11]. To test its predictions in any given task situation, it must be given the stimuli plus relevant knowledge assumed to be in memory already at the time the task is performed and the strategies used by the subject to perform the task. How this information got into memory and why and how particular strategies were adopted are also appropriate targets of scientific inquiry, but they are not part of the core EPAM theory.

If we strip all of the domain-specific content from EPAM, or a medical diagnosis program or a chess-playing program, what remains is usually a small set of fairly simple mechanisms. Likewise, BACON [21], which is capable of discovering scientific laws and new theoretical concepts for a wide range of physical and chemical phenomena, consists of a half dozen domain-independent heuristics for generating hypotheses for consideration, and a simple search-control heuristic.

Theory versus programming details

In the natural sciences, there is usually relatively little confusion between the theory and the notation in which it is expressed. Maxwell's equations can be written in old-fashioned coordinate notation or in more modern vector notation. Everyone agrees that in either form it is the same theory. Even in a more complex and subtle case, everyone agrees that Heisenberg's matrices, Schrödinger's wave equations, and Dirac's abstract algebraic formulation all represent the same theory: quantum mechanics.

In theories implemented by running programs, there is still some ambiguity as to how far down in the hierarchy of programming formalisms the theory extends. Pretty clearly, the fact that a program is written in Common Lisp is not part of the theory it expresses. But how about the fact that it is written in *some* form of Lisp? Or that it is written in a list-processing language and not an algebraic language? Since speed of execution is not an irrelevant consideration in judging the degree of intelligence in a performance, the theory in an AI program is not wholly independent of its programming implementation. Surely it is not irrelevant that most AI programs are written in list-processing languages of some kind, and that the processes in most of them are implemented as productions.

One way in which we can make clearer the substantive content of our programs is to indicate as definitely as we can the domain primitives, which can then be distinguished from the primitives of the programming language. However, this distinction is not always easily made.

Beyond the pragmatic sentiments I have just expressed, the relations between intelligence, on the one hand, and list processing and productions, on the other, have perhaps not been adequately elucidated. One might even want to strengthen the definition of a physical symbol system to include a requirement of list-processing capabilities, especially the ability to form associations and labelled associations (descriptions), as well as the ability to act on recognition (productions). Almost all intelligent programs make essential use of these abilities.

In describing intelligent programs and those of their components that are of theoretical interest, we are usually able to characterize these components in a qualitative, if inexact, fashion. This brings us back to laws of qualitative structure in AI. Much of our communication about our theories takes this shorthand form, preferably backed up by the harder currency of running programs. The shorthand allows us to ignore program details that are irrelevant to the theory; it does not provide guarantees that the informally described system will perform as advertised.

I will have more to say later about methods for describing and evaluating programs. But before we go into questions of interpretation and evaluation, let us turn to the other principal form of theory: laws of qualitative structure.

4.3. *Laws of qualitative structure in AI*

As we seek to develop theory in computer science, and specifically in AI, we should be looking for laws of qualitative structure and regularities of organization and process that characterize them. Our search for them will of necessity be primarily empirical and experimental. In the case of AI the search will be carried out by designing complex systems that embody these laws (quantified with particular parameter values, of course) and operating them under a wide range of conditions, using a variety of observations and measurements to characterize their behavior.

In our Turing Address, Newell and I proposed, in addition to the Physical

Symbol System Hypothesis, a second QS law, the Heuristic Search Hypothesis (HS):

> Problems are solved (when intelligence is required for solution) by searching selectively (heuristically) through a problem space (i.e., a problem representation).

The Heuristic Search Hypothesis

The HS Hypothesis is nearly as broad as the PSS Hypothesis, and equally qualitative. It is nevertheless quite powerful in what it excludes. It denies that problems are generally solved by exhaustive search through large problem spaces, or without the help of knowledge of the structure of the problem space. This knowledge is used by converting it into search control heuristics.

The HS Hypothesis is augmented by a number of more specific QS laws that characterize some relatively general and useful search heuristics, and by some principles of effective search control. With respect to search heuristics, we have learned that such processes as hill climbing and means–ends analysis provide powerful bases for selectivity in many task domains, and we know a good deal about the conditions under which heuristics like these are or aren't effective. Sometimes (e.g., [9]), we are even able to characterize these conditions formally.

For example, hill climbing is a reliable method only when local maxima are also global maxima, and hill climbing must be supplemented by other criteria when this condition is not met; means–ends analysis works only if the problem space is factorable in a certain sense (when operators can be ordered so that differences removed by operators of high priority are not reinstated by those of low priority [7,19]).

With respect to search control, special-purpose heuristics have been devised for particular classes of task domains: for example, alpha-beta search for game environments. The contrasts in performance among depth-first, breadth-first, and best-first strategies are fairly well understood. Most of this knowledge also takes the form of QS laws, although a few mathematical theorems are scattered through the literature (e.g., theorems about the efficiency of the A* search algorithm, measured by length of solution path; about efficiency, measured by expected computing effort, about criteria for optimal best-first search [38]).

Twenty or more years of research on expert systems has produced a third very general QS law, problem solution by recognition (REC):

> Expert systems solve frequently occurring problems largely by the process of recognition.

That is to say, an expert system (computer or human) possesses a set of productions capable of noticing cues in everyday problems and thereupon evoking the knowledge stored in memory that is relevant for dealing with the situations marked by the cues. Recognition plays a central role, for example, in medical diagnosis, whether human or automated, and in the early expert system, DENDRAL, which interprets mass spectrograph data in order to elucidate chemical structure [23].

Recognition processes are implemented, for example, in the discrimination net of EPAM and the Rete nets of production system languages. Using recognition processes enables the expert to draw upon large bodies of data, suggesting another closely related QS law, the Knowledge Principle:

> A system exhibits intelligent understanding and action at a high level of competence primarily because of the *specific* knowledge that it can bring to bear: the concepts, facts, representations, methods, models, metaphors, and heuristics about its domain of endeavor. (Lenat and Feigenbaum [22])

These examples, and especially the QS laws, PSS, HS and REC, show in what large measure our general knowledge about problem solving in AI is embedded in laws of qualitative structure which have been induced from specific expert systems modelled as computer programs. All of these programs are PSSs, and their components that implement HS and REC heuristics are easily identified. In comparison with the QS laws, the mathematical theorems that the field has created to date fade into insignificance.

The same picture emerges when we turn to other subdisciplines within AI: learning, for example. A number of systems have been constructed that learn from their own problem-solving efforts, or from the successful problem-solving efforts of others in the form of worked-out examples of problem solutions. The adaptive production systems of Waterman, Neves and others [24, 40], belong to this line of work, as do the explanation-based learning systems of Mitchell, and the chunking procedures of Soar [25]. What we have learned about learning from the construction of such systems is perhaps best summed up by the QS law of Learning from Examples (LE):

> If a production system is provided with detailed examples of problem solutions, showing the intermediate steps, then means–end analysis or some related method of causal attribution can be used to create automatically new productions capable of solving problems of the same general kind.

In yet another AI domain, more than three decades of experience in building systems for automatic translation of natural language has produced a substantial body of knowledge about the requisites of such systems, including the QS law:

> Satisfactory translation of natural language requires not only knowledge of a lexicon and syntax, but also a substantial body of semantic knowledge to provide context for resolving ambiguities.

Similar examples of QS laws can be extracted from other domains within AI. In this respect AI resembles most other scientific disciplines. In biology, for example, empirical knowledge is typically embodied in descriptions of the structures and processes of specific species of organisms, combined with more general QS laws describing general mechanisms (e.g., metabolism, processes connecting DNA with proteins, immune reactions). Increasingly, these mechanisms are being modelled symbolically with computer programs; and computer simulations are being compared with the findings of experimental manipulations.

4.4. Dealing with complexity

Currently, the topic of complexity attracts a great deal of interest, but there remains a question of what can be said meaningfully about complexity in full generality. There is more promise for theories that deal with particular aspects or forms of complexity. The mathematical theory of chaos treats the complexity of nonlinear dynamic systems whose long-term behavior is unpredictable. The theory of systems that possess many interacting components deals with another form of complexity. Theories of computational complexity deal with a third form.

Thus, while bare "complexity" may be a category too broad to support theories with any considerable content (as "general systems" proved to be), one can be optimistic about the possibility of building theories and discovering QS structure that characterize various kinds of complex systems. Let me illustrate these possibilities for just two rather closely related aspects of complexity: the hierarchical structure of many-component systems; and seriality and parallelism in complex systems.

Hierarchy

It has been observed empirically for a long time that most many-component systems—both those observed in nature and those devised by man—have a hierarchical architecture [36]. That is to say, viewed from the top down, they can be divided into subsystems that are divided in turn into subsubsystems, and so on, until we reach a level of primitives which we do not wish to, or cannot, decompose further. The First Commandment of structured programming is to respect this hierarchy by working from the top to the bottom. The Second Commandment is to minimize the interaction between different substructures (No GOTOs!). The Third Commandment is to make each level of the hierarchy insensitive to the structure of the levels below, so that adjoining levels interact only through inputs and outputs.

It would seem that structured programming was already invented many eons ago by Nature, for these commandments are observed pretty well in a vast majority of natural systems. Organisms are made of systems (digestive, respiratory, circulatory and so on); systems are made of organs, organs of tissues, tissues of cells, cells of organelles, organelles of proteins, proteins of amino acids, amino acids of atoms and so on. At a still more minute scale, we run all the way down through atomic nuclei to elementary particles, quarks, and possibly strings. At the other end of the scale, the universe contains galaxies, which contain stars, which may have planetary systems.

Hierarchy yields several laws of qualitative structure, and even some precise mathematical generalizations. An example of the former is a QS law that hierarchical systems will evolve more rapidly than non-hierarchical ones, and a law that the long-run dynamics of such systems depends (approximately) only on higher level structure, while the short-run, high frequency, dynamics is determined nearly independently within each subsystem.

One of the mathematical generalizations is a formalization of the latter QS law,

and provides algorithms for the computations [5, 36]. These laws, both qualitative and quantitative, are closely related to the commandments of structured programming mentioned above.

Hierarchy may be viewed as a powerful antidote to computational complexity. The amount of computation required to determine the (approximate) behavior of a hierarchical system can be expected to increase only linearly with the number of primitives; and if the subsystems at any given level are and remain identical (e.g., identical cells in tissues of an organism), to increase only logarithmically.

Since the possibilities for parallel computation in a system are inversely related to the number and strength of precedence constraints, and the latter are related to the intensity and frequency of interaction of parts, we would expect that hierarchical organization would be conducive to parallelism, provided that the lines of hierarchy were used to guide the boundaries of the parallel subsystems. That is to say, we would expect to gain from a capability for parallelism between components that do not have high-frequency interaction.

These claims would have to be made much more precise before they could be taken literally, but they illustrate one way in which one might approach the design of parallel systems. It is interesting that principles like these are instantiated in most human organizations, which are almost always hierarchical (I am referring not to hierarchy of authority but to boxes-within-boxes departmental arrangements), and with the hierarchy arranged so that larger units have relatively infrequent occasion to interact with each other.

As with the other topics we have discussed, we see that empirical research has to play a major role in the study of complexity, but that there is also room for mathematical theory that will at least handle simplified models of the complex real phenomena and give guidance to the conduct and interpretation of experiments. To a major extent, we will reach an understanding of complex systems through building and testing them.

4.5. The role of formal theory

I have already made a number of comments about the role of formal theories in artificial intelligence, observing that precise mathematical theorems have played only a modest role in AI, and are unlikely to play a central role in the foreseeable future. As in most other empirical sciences, the theories of greatest import and impact have been laws of qualitative structure, supported by detailed experiments and simulations.

On the other side of the matter, "precise mathematical theorems" is not synonymous with "formal theories". Computer programs meet the same standards of precision as do the symbolic expressions of other parts of mathematics. What distinguishes them from some mathematics that has been applied to the simpler problems in the physical sciences is that they do not usually admit solutions in closed form (i.e., theorems). Consequently, as we have seen, the principal technique for drawing inferences from them is to run them in appropriate task environments and evaluate their behavior.

If we think of theorems and simulation simply as two kinds of formal treatment, we obtain the former when we simplify and abstract the real world to fit the mathematical tools we have available. We obtain the latter when we take into account more of the world's complexity. In computer science in general, and in AI in particular, we are usually operating in areas of greater complexity than those in which theorems can be proved. This is not a virtue; it is simply a fact of life. We should treasure the occasions when theorems of some generality, power and relevance can be proved.

Initial and boundary conditions in programs

Some feel uncomfortable because programs seem so much more complex than Newton's laws of motion, or Maxwell's equations, or even the laws of quantum mechanics. Some comfort can be gained from the point made earlier, that a substantial part of this complexity in AI programs arises because they incorporate not only basic general mechanisms for performing their tasks, but also a great deal of knowledge pertaining to particular task domains, and strategies and heuristics specific to those domains. The domain-specific knowledge and strategies correspond to the initial and boundary conditions of theories in other sciences.

Theory primitives versus programming details

I have also observed that even the computer code representing this kernel does not signal where the AI theory leaves off and pure programming convenience takes over. Where that boundary lies is a substantive, i.e., experimental, question.

Specification languages for theories

A different way to clarify the theoretical claims embedded in programs is to define relatively formal languages that are not as precisely implemented as programming languages, but that describe the theory in a form that allows anyone skilled in the art to program it. Examples of this method of generalizing while retaining a good deal of precision will be found in the languages used in our book, *Scientific Discovery*, to describe the BACON program and the other discovery programs discussed there; and in several formalisms used in *Human Problem Solving* to describe GPS and other problem-solving programs.

Some efforts are now under way [4] to define standard specification languages that could be used to define theories in a formal way, short of full implementation; but it remains to be seen whether a single language can do the job or whether a variety of languages will be needed to accommodate radically different representations used in handling different cognitive tasks, or the same tasks with different strategies.

And, finally, the mechanisms that programs incorporate can usually be stated even more succinctly, if less precisely, as laws of qualitative structure.

5. Evaluating intelligent systems

Evaluating the success of an artificial intelligence research effort can be relatively simple or it can be complex. When the Logic Theorist (LT) demonstrated that a rather primitive heuristic search, with a modest capability for selectivity, could find proofs for many theorems in *Principia*, a basic work on logic, that fact alone told us a great deal about intelligence. The significance of the result depended on the task being nontrivial for humans. It depended also on the fact that the program required modest amounts of computation (almost trivial amounts by today's standards), but amounts comparable to what we might think a human brain could provide. It depended on the fact that LT's heuristics, though simple, made its search highly selective as compared with brute-force search.

Similar statements can be made about molecule identification systems like DENDRAL, Winograd's SHRDLU, medical diagnosis systems like MYCIN and INTERNIST, or scientific discovery systems like AM and BACON. What makes these systems centrally interesting for AI is that they perform tasks that, in humans, require professional levels of intelligence and knowledge, and in doing so, exhibit a combination of knowledge base, computing power and heuristics sufficient for the task. We can echo Samuel Johnson's statement about the dancing dog: "The marvel is not that it dances well; the marvel is that it dances at all." Demonstrating the range of tasks requiring intelligence that can be programmed for computers and describing the nature of these programs are major goals of the AI enterprise. When we speak of evaluating such programs, our main focus should be on understanding them.

5.1. The purposes of evaluation

How complex and difficult a matter it is to evaluate a system will depend upon our goals. As we have just seen, if our purpose is to advance the pure theory of intelligence, our first aim will be to construct systems that exemplify intelligence of different forms in different task environments. Evaluating whether they do what we expected them to do may be relatively simple. But there are other occasions when evaluation must be more elaborate and principled.

Simulating human intelligence

When our interest lies in understanding human mental processes, showing that the programs can do tasks that human professionals are capable of is only the first step. If we wish to claim that BACON teaches us something about how human scientists make discoveries, then we must also compare BACON's processes with data, from field or history or laboratory, describing the processes actually used by scientists.

Designing expert systems

When the interest lies in creating expert systems, like DENDRAL or MYCIN, that can complement, supplement or replace activities of human experts, the

mechanisms employed and the measures of success change again. In building expert systems the processes used by our systems will not be limited to human processes; but we will need to compare the quality of the programs' performances with human performance, and with the performances of other expert systems in the same domain, along all dimensions of concern: e.g., quality of solutions, error rate, cost, user-friendliness, and so on.

Extending theory

In both cases just mentioned there is more to system synthesis than merely designing and evaluating specific systems for particular uses. There is an interest in improving designs (or simulations). The research includes determining what characteristics systems must have and what general principles they must embody to enable them to perform their tasks and perform them effectively.

Thus, in the domain of artificial intelligence, we need theories of the characteristics and underlying principles of systems capable of holding information in memory and retrieving it when needed, and theories of problem-solving systems, systems for inducing concepts, systems for learning, systems for navigating and operating in the external world (robots), systems for understanding human speech so on. The more powerful these theories, the more we can anticipate what properties systems in these domains must possess to exhibit intelligence (humanoid or other), and the better the systems we can design.

Improving the design process

In addition, research may aim at improving the efficiency of the design and evaluation processes themselves. As design and evaluation are intelligent processes, hence come within the scope of artificial intelligence and the PSS Hypothesis, this kind of research is not distinct from the research previously mentioned. Indeed the theory of designing can best be regarded as a special part of the theory of solving problems. It can be studied by creating and studying systems for automatic design.

5.2. Evaluating expert system designs

In the case of expert systems, evaluation of a particular design is often very pragmatic: Does the new system perform better and/or more efficiently than systems already available? The greater the superiority, the more easily it can be demonstrated. Analogously, although statistics were reported about how rapidly the early computers of the 1940s performed certain computations, the most important news was that they performed them. This was enough information to support their continued development. So it has been with steam engines, automobiles, airplanes, radios and all other major engineering innovations.

This is not to argue that evaluation is unimportant for the advancement of technology; but it *is* to argue that at the frontiers of a new technology, very crude qualitative evaluation may be enough to point the way. The design process, with its constant modification of the emerging system to meet difficulties and failures in

performance, incorporates in itself a severe regime of evaluation. Moreover, considerations of extendability, visible to the designers who are familiar with the design details, but not revealed in the performance of early designs, may be more important in pointing toward fruitful directions for R&D than statistics of performance.

We are faced with the celebrated recipe for rabbit stew, which begins: "First catch the rabbit." First design a system that has the desired general capability, at least at a minimal level. Having accomplished that, improvement of the design and final evaluation may be very difficult, but at least it has a foundation on which to proceed.

The immediate research goal is sometimes to build a system that will be of practical use. More often, the goal is to use design and evaluation as a basis for building AI theories of the kinds suggested in the previous sections. The tasks are selected for feasibility and for the light they can throw on the general principles of organization and operation of intelligent systems, paving the way for construction at a later date of systems having real-world utility.

In the beginning, tasks were selected that were relatively simple and well structured, and that called for little real-world knowledge. Standard environments like chess, the Tower of Hanoi and the Blocks World provided us with situations within which we could experiment and reach understanding of the properties and operation of fundamental problem-solving mechanisms.

With growing success in designing such systems (and growing size and speed of the computers available for simulation), the research gradually extended to tasks calling for large amounts of real-world knowledge and tasks where the initial goals and constraints were less well defined: interpreting mass spectrograms, diagnosing disease. They have taught us, among other things, how knowledge must be organized in memory and processed in order to permit intelligent response to knowledge-rich task environments.

Robotics tasks, that is, tasks in which a system must deal with an actual real-world environment, are of growing importance to AI research, for they compel attention to kinds and levels of complexity and uncertainty that can be finessed in laboratory test beds. (What I am calling "robotics" is not limited to tasks calling for sensing and physical response; a scheduling system that handles an actual flow of factory orders and responds to genuine information about completion, machine down-times, cancellations, data errors, etcetera, is also a "robot" for these purposes.)

By now, a very wide range of tasks has been explored, including many that, when performed by human beings, call for professional-level expertise, for learning, and even for those qualities we call "intuition", "insight" or "creativity".

Which of these kinds of benchmark tasks and test beds offer the greatest promise for future AI research? I can only answer, "All of the above". A recent article in the *AI Magazine* [15] provides a thoughtful discussion of the issues, and, especially as the authors reveal their disagreements along with their points of agreement, illustrates vividly the complementarity of different sorts of test beds

for the AI enterprise. If I had to express a preference, I would generally endorse Steve Hanks' remarks about "the dangers of experimentation in the small", not because I think such experimentation is unnecessary, but because its manageability and cleanliness sometimes seduces us into neglecting domains of real-world complexity, and refusing to face squarely the issues of extendability.

What is a success, and what a failure, in expert system design? Three kinds of criteria have been evoked: comparison with human performance, measurement of performance on a standard set of tasks from the domain of interest, and comparison with a theoretically determined upper bound of performance. But before we take up these possible solutions to the evaluation problem, we need to say something about what we mean by a "good" or "effective" expert system.

Dimensions of effectiveness

Three dimensions of effectiveness come immediately to mind: quality of performance, range and flexibility, and computational efficiency. We can judge a chess program, for example, along the first and third dimensions by its strength of play and by the time it takes to make a move. Of course there will often be a trade-off between these two criteria. With respect to range and flexibility, a program to play chess is of no use for other tasks, whereas the General Problem Solver can attempt any task for which a suitable representation and table of connections between operators and differences can be devised. "Quality of performance" is itself a multi-dimensional criterion, which may include such components as reliability, graceful degradation, user-friendliness and others.

Comparisons with human performance

As soon as it was shown that one can invert a large matrix more rapidly with a digital computer than wit a desk calculator, people began to do just that. Within a very short time the computer was so much more powerful than the calculator, even in economic terms, that sophisticated evaluation was not required to demonstrate the superiority. The same may be said, in general, about expert systems having more of an AI flavor than those used for matrix inversion.

Levels of human performance provide useful benchmarks for measuring the quality of expert systems, as long as system performance lies within the human range. Human performance not only provides a metric through that range, but also calibrates the breadth and flexibility of system performance over diverse tasks. Such measures can be used whether or not the expert system imitates human processes.

Comparison on standard tasks

It is convenient to have available a set of benchmark standard tasks sampled in some way from a domain's population of tasks [12, 33]. Standard tasks can be used to evaluate a system at various stages of its development, and especially to compare the power of competing systems. The difficulty lies in defining an appropriate standard.

Suppose we wish to evaluate a medical diagnosis system. (For an excellent

recent example, see an evaluation of the QMR system by N.B. Giuse, et al. [13].) We might be able to list the diseases for which we want the system to work. But the symptoms that signal the presence of these diseases are highly variable. Our sample of tasks needs to be a sample not only over diseases, but over patterns of symptoms, including patterns where several ailments are present simultaneously. Should we weight our sample by frequencies of symptom patterns in the human population? Should we weight it by the seriousness of the diseases that the symptoms signal? Some of these questions might be answered, at least in principle, by using a decision-theoretic approach to the sampling problem.

But perhaps we are making the problem more difficult than it really is. For many practical purposes we can assume that any relatively broad sample will rank different expert systems in roughly the same order as any other broad sample. At best, this will only be approximately true and is helpful only if our main interest is in arriving at such a ranking. If we wish to use the test tasks as a guide to system improvement, then their inclusiveness, and the possibility of relating their components to specific system components may be more important than their representativeness.

With all of these qualifications, standard sets of tasks can generally provide useful benchmarks for evaluating the performance of expert systems and for comparing different systems. A good example is the set of tests that ARPA set on several occasions for the evaluation of speech recognition systems [26]. Different systems were tested on the same corpus of speech, and evaluated with respect to both speed and accuracy. Such tests can be designed to evaluate over specified ranges of vocabulary and subject matter.

Similarly, chess programs are routinely evaluated by competition between programs and with human players, and their strength specified by ELO ratings, using the same scale that is used to rate human chess players.

Objections have sometimes been raised to this kind of "horse racing" as diverting attention from underlying scientific principles to mere ways of "winning". But the objections lack force, for to win means to demonstrate a powerful system, and the tests of effectiveness provide rich information not only about who is stronger, but also about the specific strengths and weaknesses of the designs, the very kinds of empirical information that lead to understanding. These competitions discourage projects from searching under street lamps where it is easier to find things—however irrelevant the discoveries may be to the goal.

Theoretical bounds on performance

Theories of computational complexity have given us some theoretically-motivated bounds on system performance. One disadvantage of the existing theories is that most of them have focused on criteria that facilitate proving mathematical theorems about complexity, and these may not be the criteria we would be using to evaluate our systems. Another case of the street lamp.

The criterion about which it has been easiest to prove theorems is worst-case performance in the limit, as some measure of problem size, N, grows indefinitely

large. By this criterion, a system whose expected worst-case computation time grows exponentially with N is inferior to one whose time grows only polynomially with N. Moreover, any system that does not guarantee completeness (reaching a solution for every problem in finite time) automatically fails the worst-case test.

Surely we would prefer to measure quality by expected computation times rather than worst-case times, but to determine the former, we have to define a probability measure over the population of problems. For this, and other reasons of mathematical feasibility, it has seldom proved possible to estimate expected computation times, and most of the existing theorems use a worst-case criterion.

One of the pioneers in complexity theory, Michael Rabin, as early as 1974 [32] described the dilemma I have just presented, expressed his dissatisfaction with the limits of existing theory, and offered some remedial suggestions (including defining algorithms of limited applicability and allowing computation with occasional errors). What he did not suggest, and what I would offer as solution for the dilemma, is the idea of using empirical criteria, based on actual computing experience, of "what works", in those usual cases where theorems of the desired kind are unavailable. That is, in fact, what both AI and numerical computation have done from the beginning.

For many purposes, we will prefer to use computational systems that, though sometimes failing to solve problems, *usually* solve them in a short time and solve a large fraction of the problems presented in an acceptable (if not always short) time. Fig. 1 illustrates hypothetically some of the alternative ways in which we might want to evaluate systems. System A in the figure solves 60 per cent of the problems presented to it in 1 minute, but appears to reach asymptote after an

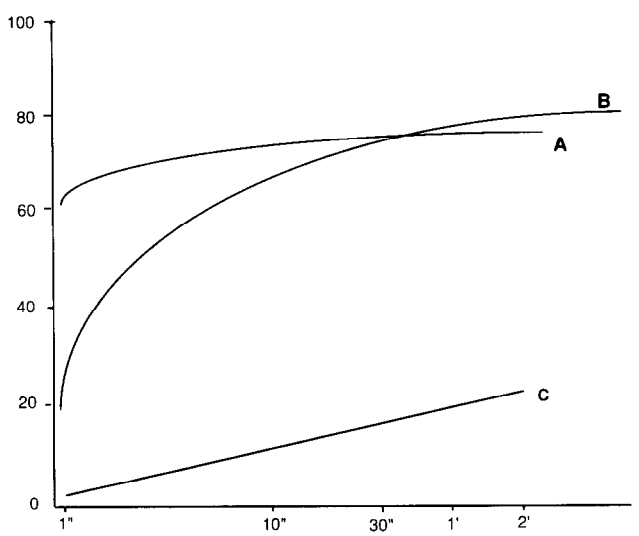

Fig. 1. Problem solving power of three algorithms.

hour with only 75 per cent of the total solved. System B reaches 60 per cent only after 30 minutes, but solves 75 per cent with an hour, and appears to reach asymptote above 90 per cent. System C is guaranteed to solve all of the problems sooner or later, but solves only 10 per cent in 30 minutes, and 20 per cent in two hours.

It is not evident which of these systems is to be preferred, and preference will depend on computing costs and on how serious the consequences are of failing to solve a problem within a given computing time. When the problems to be solved are very large compared with our computing resources, we will seldom want to rank alternative systems according to how long it will take them to solve every problem.

Because of the relatively poor match between the mathematical tractability of criteria for evaluating programs, on the one hand, and the practical significance of the criteria, on the other hand, mathematical theories have not taken us very far in system evaluation, and we have to rely principally on empirical evaluation methods to guide design.

Closely related to issues of computational complexity is the question of the scalability of a design. Designs that work well in a small scale do not always scale up well. Here, because algorithms that are exponential in problem size explode rapidly, the standard measures of computational complexity may be of some value. However, the practical question is not how systems scale in the limit, but what computation they require when used on problems of the sizes that occur in practice. It is not particularly interesting to know whether a program for natural language processing could handle a vocabulary of ten billion words; it is usually much more interesting to know whether it can handle a vocabulary of one hundred thousand words (or, for some applications, even 100 words).

Systems can also be evaluated for goodness of solution. In operations research (linear programming, integer programming, etcetera) there is a tradition of evaluating the efficiency of a program by the times it requires to reach optimal solutions, but in many situations we might prefer a system that would usually reach a very good (not necessarily optimal) solution in a short time to one that would find the optimum, but only after much computation. In systems that have to respond in real time, for instance, computation ordinarily returns the solution that is "best so far" when the time limit is reached, or a "satisfactory" solution as soon as one is obtained. It may be either impossible, or simply wasteful, to wring out the last drop of approximation to the optimum solution.

There has also been a tradition (e.g., in relation to the A* search algorithm) of seeking to minimize the number of steps to solution; whereas in many domains, the number of steps to solution is of little interest; what is wanted is to conserve on computation time required for finding the solution [38]. If we are proving difficult theorems, we more often set the goal of finding a proof than of finding the shortest proof. Shortest path to solution and shortest expected computation time to solution are completely different criteria, and it is usually the latter, not the former, that is relevant.

5.3. Use of statistical tests in evolution

The AI literature has made rather little use of standard statistical methods for testing hypotheses. One can argue both sides of the question of whether this is unfortunate or fortunate. I must confess to some rather strong attitudes on the matter formed in the course of long experience with hypothesis testing theory, as a user and observer of typical usage and, at times, as a contributor to the theory. I can only provide these views in brief summary here, with some pointers to the published literature.

Tests of statistical significance are widely used in psychology and biology, very little in physics. (The classical "probable errors" of physics are usually estimates of the accuracy of instruments, not tests of the probability that an observed phenomenon could have occurred by chance.) Tests of significance are legitimately used to test whether a variable produces any effect (as compared with the null hypothesis that the observed "effect" was produced by chance). The presence or absence of statistical significance says nothing whatsoever of whether the effect is important in magnitude: significance and importance are unrelated quantities.

As mathematical statisticians agree unanimously, tests of statistical significance cannot be used to test whether a model fits data. (For a brief explanation of why they can't, see [14] or [35] and the references cited there.) One appropriate alternative to reporting such tests is to report percentage of variance explained (R^2). In addition, regression coefficients (*not* correlation coefficients), showing how much a dependent variable changes with change in an independent one, provide measures of the *importance* of the independent variable. Even the latter statement holds only if the equations being analyzed are *structural* equations, reflecting the underlying causal structure of the phenomena [17].

As AI is primarily concerned with evaluating systems (i.e., models), statistical hypothesis testing is unlikely to play a very useful role in the enterprise, although a greater attention to measuring the magnitude, hence importance, of the effects produced by changes in systems operating in particular kinds of environments would be very desirable.

6. Theories of human intelligence

Throughout this paper I have emphasized the direction of AI research that is concerned with the general theory of intelligence, and have not had much to say about models of human cognitive processes: psychological models.

In spite of the shared method of research, the design and evaluation of systems, there is no immediately obvious reason why there should be any close connection between research directed at designing intelligent systems and simulating human cognition, or between the corresponding theories of intelligence. It could be that, because of the radical differences between electronic devices and brains, pro-

grams designed to be efficient expert systems would be totally different in architecture and process from systems designed to simulate human thinking.

To some extent, this is the case. The strongest chess-playing programs (designed specifically as expert systems) do not play chess in the same way as human grandmasters. Their problem-solving searches are far more extensive and far less selective than the searches of human chessmasters [39]. However, a few chess programs of a different kind have been written (e.g., the NSS program, MATER, PARADISE) with the aim of understanding human play by imitating it.

As of 1993, the most powerful of the programs designed as expert systems play chess at a formidable grandmaster level; the most powerful of the programs designed as cognitive simulations are modest amateurs. The programs designed as expert systems conduct enormous searches, well beyond human capabilities, before making their moves, and generally make use of only a moderate store of chess knowledge. The cognitive simulations explore only a few (perhaps hundreds, but not millions) of the branches of the game tree and make use of heuristic chess knowledge to select the branches to be explored.

But there is another side to the matter. While expert systems and cognitive simulations are subject to quite different internal constraints (the physics of computers versus the biology of brains), when they are performing the same tasks they are subject to the same external constraints— the same task demands. To the extent that the task requirements are numerous and heavy, these requirements may reveal themselves in fundamental similarities between the programs that perform them, however dissimilar the means of implementation at the lowest hardware level—the inner constraints.

Programs constructed to simulate human behavior are evaluated differently from programs constructed simply to perform efficiently tasks requiring intelligence. The former are tested by comparing their behavior with the behavior of human beings in the same task environments—exactly as any theories are tested whose function is to describe and explain empirical phenomena in some domain.

There is nothing about the methodology that distinguishes it at a general level from the methodology in any empirical science. The general paradigm is to use the theory to predict the empirical phenomena, observe the phenomena, compare the predictions with the observations and revise the theory to bring about better agreement of theory with data. As there is a large substantive literature on symbolic models of cognition (e.g., [1, 28]) as well as a literature on methodology (e.g., [6, 25]), I will not discuss the topic further here.

7. The future of artificial intelligence

At the outset of my remarks, I said that I was going to prescribe for the future of AI on the basis of what the past has taught us. A number of important lessons of the past should now be reasonably plain, and I need summarize them only briefly.

7.1. The fundamental strategy: pushing the frontier

Our main task in AI continues to be to explore a wider and wider range of activities for whose performance intelligence is essential. We need to identify aspects of intelligence that we have not yet succeeded in handling, and attack each in turn as soon as we have any ideas about how to proceed. The attack must consist, as in the past, of building systems that actually perform the tasks and produce the phenomena associated with them. We need to evaluate our programs for their efficiency, and for scope and scalability.

It is not hard to identify some task domains that should be high on the priority list, although no claims can be made for the completeness of such a list. My own candidates would be two that are already receiving considerable attention, and another that is still pretty much on the frontier. These three candidates are: machine learning, robotics and representation (including change of representation).

Machine learning

Machine learning is in a vigorous state of activity with its own journal and specialists within AI. We might better say "specialists, alas", for there is danger that the specialization will delay the impact of new discoveries about learning mechanisms on mainstream AI. Both serial and connectionist learning techniques are being explored, a healthy competition that will teach us in which domains each is effective.

There have been some achievements in theory—notably theorems about the convergence of learning algorithms—but what I have said about theorems and theories in the body of my paper is as applicable to learning as to other topics in AI. Running programs and their evaluation, and QS laws are the key to progress. Pat Langley (personal communication) believes that there is today a reasonable balance between empirical work and theory in machine learning. A count of papers presented at a recent machine learning conference showed that of 44 papers, 39 contained experimental evaluations of specific programs with explicit measures of performance. On the other hand, the conferences on computational learning theory, probably populated more by computer scientists from theory than by specialists in artificial intelligence, present mainly theoretical papers.

Robotics

Robotics is also a vigorous field, and also in danger of being too far separated from mainstream AI. I viewed with mixed feelings the establishment of a separate graduate program in robotics in my own university. One of the common criticisms of much mainstream AI research is that no distinction is made, in modelling problem situations, between the actual, real-world, situation and the model of the situation stored in computer memory. As robotics cannot afford this luxury of confusing the model with external reality, it must incorporate in its systems feedback channels that can correct the models periodically to reflect reality more accurately. Of course, this distinction can be attained in AI modelling, by keeping

in memory both an abstracted model and a simulated "real world", but the virtue of robotics is that it makes the distinction a necessity instead of an option, and continually reminds the system builder of the complexity of the *real* real world—the one outside the computer.

Representation

It is still typically the case that before a computer can exhibit intelligence in handling any task it must be provided with a representation of the task domain: a problem space that specifies the kinds of objects and phenomena in the problem states, and the kinds of operators that are available for changing one problem state into another. Some work has been done, but only a modest amount, to show how problem representations can be generated from external information.

The UNDERSTAND program [16] for example, can generate problem representations suitable as inputs to the General Problem Solver from verbal descriptions of problems (simple puzzles). The ISAAC program [29] can generate representations for specific physics problems from natural language descriptions in textbooks. The Soar program of Newell, Laird and Rosenbloom [25], has some capability for modifying its representations as it moves to new problem spaces. Korf [19] specified some procedures for representation change in a restricted class of abstract problems. Kaplan and Simon [18] have proposed a method that would produce the critical change in representation needed to solve the Mutilated Checkerboard problem.

This sample of what has been done reminds us of how much remains to be done. What would a program be like that could invent the calculus, or even one that could select the calculus as the appropriate representation for dealing with a particular problem? Through what sequence of steps was Heisenberg's matrix representation for quantum mechanics obtained, or Schrödinger's wave representation?

At the most general level, the topic of representation leads us to consider the differences between reasoning in words or in formal linguistic representations (logic, mathematics) and reasoning from pictures or diagrams. Understanding the properties of these representations is a basic issue not only within AI, but also in the whole domain of human–computer interaction.

7.2. Methodology

In arguing that the past progress in AI came largely from constructing ever more sophisticated and complex intelligent programs to perform increasingly difficult and ill-structured tasks, I should not like to leave the impression that the methodology we have used has been faultless and that there is not room for great improvement in it. Wherever theory can be extracted from our models and the phenomena they produce, we should extract it—just as is done in other empirical sciences. But perhaps the greatest opportunity for improvement in method is to make our experiments cumulative by (1) attending more systematically and carefully to system evaluation, and (2) paying a great deal more attention to

comparison between systems as a basis for understanding the mechanisms, and their interactions, that account for outcomes. Again, in discussing these matters, I will address mainly the expert system side of AI and not comment on the psychological side, where methodology has perhaps reached a somewhat higher level of sophistication.

Standard tasks for evaluation

To evaluate systems in various task domains, we need standard sets of tasks that will be used repeatedly by many investigators studying different systems. I mentioned earlier the standards that DARPA has used to evaluate progress in speech understanding. It could be a very appropriate activity for our professional association to establish, through committees, such standards for a range of areas. Not only would this create a consensus on benchmarks for investigators, but it would also lead to a valuable dialogue on what constitutes good performance, how the criteria depend on the presence or absence of real-time constraints on computation, the relation between performance standards and theories of computational complexity, and so on.

Extraction of general mechanisms and principles

The aim in evaluating systems is not simply to establish which is better at any given moment, however interesting such contests (as among chess computers) may be. Because we name, and thereby trademark, our systems, what we usually report in the literature are comparisons between large complex systems, each consisting of a substantial number of interacting mechanisms. After we have learned that one system has solved more problems than another, or solved them faster, we do not yet know the reasons for the superiority. We do not even know in what respects they use quite similar or quite different mechanisms to accomplish their results.

The comparison of systems does not end when we have determined which performs better on particular kinds of tasks. That is just the beginning of a comparison of the mechanisms that each system employs and the contributions of these mechanisms to system performance on different kinds of tasks. What we are seeking, as always, are QS laws that can guide the design of the next system, and that can advance the general theory of intelligence and intelligent systems.

If, in my account of the past and future of AI research, I have emphasized empirical methods over formal theory, it is because I have sensed, in our meetings and in the pages of our journals in recent years, a kind of theory envy that sometimes sacrifices attention to complex but real problems in favor of attention to over-simple problems that are amenable to exact mathematical treatment. Some distinguished members of our profession have even challenged and demeaned the very concept of experimental computer science.

My own scientific record is strewn with examples of mathematical work, some of it relevant to real problems, some of it perhaps not. The issue is not whether to replace mathematics with experiments, or vice versa; it is to secure and maintain a tolerance throughout our discipline for a plurality of approaches to our deep

scientific problems; and a dedication to improving each of these approaches in its own terms.

Acknowledgments

This research was supported by the National Science Foundation, Grant No. DBS-9121027; and by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539. Reproduction in whole or in part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

References

[1] J.R. Anderson, *The Architecture of Cognition* (Harvard University Press, Cambridge, MA, 1983).
[2] R.A. Brooks, Intelligence without representation, *Artif. Intell.* **47** (1991) 139–159.
[3] A.W. Burks, H.H. Goldstine and J. von Neumann, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument* (Institute for Advanced Study, Princeton, NJ, 1946).
[4] R. Cooper, J. Farringdon and J. Fox, Towards a systematic methodology for cognitive modelling, Unpublished manuscript.
[5] P.J. Courtois, *Decomposability: Queuing and Computer System Applications* (Academic Press, New York, 1977).
[6] K.A. Ericsson and H.A. Simon, *Protocol Analysis* (MIT Press, Cambridge, MA, 1993).
[7] G.W. Ernst, Sufficiency conditions for the success of GPS, *J. ACM* **16** (1969) 517–533.
[8] G.W. Ernst and A. Newell, *GPS: A Case Study in Generality and Problem Solving* (Academic Press, New York, 1969).
[9] O. Etzioni, Why PRODIGY/EBL works, in: *Proceedings AAAI-90*, Boston, MA (1990) 916–922.
[10] E.A. Feigenbaum and J.A. Feldman, *Computers and Thought* (McGraw-Hill, New York, 1963).
[11] E.A. Feigenbaum and H.A. Simon, EPAM-like models of recognition and learning, *Cognitive Sci.* **8** (1984) 305–336.
[12] J. Gaschnig, Performance measurement and analysis of certain search algorithms, Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA (1979).
[13] N.B. Giuse et al., Evaluating consensus among physicians in medical knowledge base construction, *Methods Inf. Med.* **32** (1993) 137–145.
[14] L.W. Gregg and H.A. Simon, Process models and stochastic theories of simple concept formation, *J. Math. Psychol.* **4** (1967) 246–276.
[15] S. Hanks, M.E. Pollack and P.R. Cohen, Benchmarks, test beds, controlled experimentation, and the design of agent architectures, *AI Mag.* **14** (4) (1993) 17–42.
[16] J.R. Hayes and H.A. Simon, Understanding written problem instructions, in: L.W. Gregg, ed., *Knowledge and Cognition* (Erlbaum, Potomac, MD, 1974).
[17] W.C. Hood and T.C. Koopmans, eds., *Studies in Econometric Method* (Wiley, New York, 1953).
[18] C.A. Kaplan and H.A. Simon, In search of insight, *Cognitive Psychol.* **22** (1990) 374–419.
[19] R.E. Korf, Toward a model of representational changes, *Artif. Intell.* **14** (1980) 41–78.
[20] P. Langley and D. Kibler, The experimental study of machine learning, Unpublished Tech. Report, NASA Ames Research Center, Moffett Field, CA (1991).

[21] P. Langley, H.A. Simon, G.L. Bradshaw and J.M. Zytkow, *Scientific Discovery* (MIT Press, Cambridge, MA, 1987).
[22] D.B. Lenat and E.A. Feigenbaum, On the thresholds of knowledge, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 1173–1182; also *Artif. Intell.* **47** (1991) 185–250.
[23] R.K. Lindsay, B.G. Buchanan, E.A. Feigenbaum and J. Lederberg, DENDRAL: a case study of the first expert system for scientific hypothesis formation, *Artif. Intell.* **61** (1993) 209–261.
[24] D.M. Neves, A computer program that learns algebraic procedures by examining examples and working problems in a textbook, in: *Proceedings 2nd National Conference of the Canadian Society for Computational Studies of Intelligence*, Toronto, Ont. (1978) 191–195.
[25] A. Newell, *Unified Theories of Cognition* (Harvard University Press, Cambridge, MA, 1990).
[26] A. Newell et al., Speech understanding systems: final report of a study group, Department of Computer Science, Carnegie Mellon University (1971).
[27] A. Newell, A.J. Perlis and H.A. Simon, What is computer science?, *Science* **157** (1967) 1373–1374.
[28] A. Newell and H.A. Simon, Computer science as empirical inquiry: symbols and search, *Comm. ACM* **19** (1976) 113–126.
[29] G.S. Novak, Representation of knowledge in a program for solving physics aproblems, in: *Proceedings IJCAI-77*, Cambridge, MA (1977) 286–291.
[30] D.A. Pomerleau, J. Gowdy and C.E. Thorpe, Combining artificial networks and symbolic processing for autonomous robot guidance, *J. Eng. Appl. Artif. Intell.* **4** (1991) 961–967.
[31] H. Pople, Problem solving: an exercise in synthetic reasoning, in: *Proceedings IJCAI-77*, Cambridge, MA (1977).
[32] M.O. Rabin, Theoretical impediments to artificial intelligence, in: *Proceedings IFIPS Conference* (1974).
[33] A.M. Segre, C. Elkan and A. Russell, Technical note: a critical look at experimental evaluations of EBL, *Mach. Learning* **6** (2) (1991) 183–196.
[34] L. Siklóssy, Natural language learning by computer, in: H.A. Simon and L. Siklóssy, eds., *Representation and Meaning* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
[35] H.A. Simon, On judging the plausibility of theories, in: Van Roostelaar and Staal, eds., *Logic, Methodology and Philosophy of Sciences III* (North-Holland, Amsterdam, 1968).
[36] H.A. Simon, *The Sciences of the Artificial* (MIT Press, Cambridge, MA, 2nd ed., 1981).
[37] H.A. Simon, Cognitive architectures and rational analysis: comment, in: K. vanLehn, ed., *Architectures for Intelligence* (Lawrence Erlbaum, Hillsdale, NJ, 1991).
[38] H.A. Simon and J.B. Kadane, Optimal problem-solving search: all-or-none solutions, *Artif. Intell.* **6** (1975) 235–248.
[39] H.A. Simon and J. Schaeffer, The game of chess, in: R.J. Aumann and S. Hart, eds., *Handbook of Game Theory* (Elsevier, Amsterdam, 1992) 1–17.
[40] D. Waterman, Generalization learning techniques for automating the learning of heuristics, *Artif. Intell.* **1** (1970) 120–170.
[41] T. Winograd, *Understanding Natural Language* (Academic Press, New York, 1972).