COMPUTER SCIENCE AS EMPIRICAL INQUIRY: SYMBOLS AND SEARCH

Allen Newell and Herbert A. Simon
November 1975

Carnegie-Mellon University
Pittsburgh, Pennsylvania

Our research over the years has been supported in part by the Advanced Research Projects Agency of the Department of Defense (monitored by the Air Force Office of Scientific Research) and in part by the National Institutes of Mental Health.

ACM Turing Award Lecture December 1, 1975

COMPUTER SCIENCE AS EMPIRICAL INQUIRY: SYMBOLS AND SEARCH

Allen Newell and Herbert A. Simon

Carnegie-Mellon University

Computer Science is the study of the phenomena surrounding computers. The founders of this society understood this very well when they called themselves the Association for Computing Machinery. The machine -- not just the hardware, but the programmed, living machine -- is the organism we study.

This is the tenth Turing Lecture. The nine persons who preceded us on this platform have presented nine different views of computer science. For our organism, the machine, can be studied at many levels and from many sides. We are deeply honored to appear here today and to present yet another view, the one that has permeated the scientific work for which we have been cited. We wish to speak of computer science as empirical inquiry.

Our view is only one of many; the previous lectures make that clear. However, even taken together the lectures fail to cover the whole scope of our science. Many fundamental aspects of it have not been represented in these ten awards. And if the time ever arrives, surely not soon, when the compass has been boxed, when computer science has been discussed from every side, it will be time to start the cycle again. For the hare as lecturer will have to make an annual sprint to overtake the cumulation of small, incremental gains that the tortoise of scientific and technical development has achieved in his steady march. Each year will create a new gap and call for a new sprint, for in science there is no final word.

Computer science is an empirical discipline. We would have called it an

experimental science, but like astronomy, economics and geology, some of its unique forms of observation and experience do not fit a narrow stereotype of the experimental method. None the less, they are experiments. Each new machine that is built is an experiment. Actually constructing the machine poses a question to nature; and we listen for the answer by observing the machine in operation and analyzing it by all analytical and measurement means available. Each new program that is built is an experiment. It poses a question to nature, and its behavior offers clues to an answer. Neither machines nor programs are black boxes; they are artifacts that have been designed, both hardware and software, and we can open them up and look inside. We can relate their structure to their behavior and draw many lessons from a single experiment. We don't have to build 100 copies of, say, a theorem prover, to demonstrate statistically that it has not overcome the combinatorial explosion of search in the way hoped for. Inspection of the program in the light of a few runs reveals the flaw and lets us proceed to the next attempt.

We build computers and programs for many reasons. We build them to serve society and as tools for carrying out the economic tasks of society. But as basic scientists we build machines and programs as a way of discovering new phenomena and analyzing phenomena we already know about. Society often becomes confused about this, believing that computers and programs are to be constructed only for the economic use that can be made of them (or as intermediate items in a developmental sequence leading to such use). It needs to understand that the phenomena surrounding computers are deep and obscure, requiring much experimentation to assess their nature. It needs to understand that, as in any science, the gains that accrue from such experimentation and understanding pay off in the permanent

acquisition of new techniques; and that it is these techniques that will create the instruments to help society in achieving its goals.

Our purpose today, however, is not to plead for understanding from an outside world. It is to examine one aspect of our science, the development of new basic understanding by empirical inquiry. This is best done by illustrations. We will be pardoned if, presuming upon the occasion, we choose our examples from the area of our own research. As will become apparent, these examples involve the whole development of Artificial Intelligence, especially in its early years. They rest on much more than our own personal contributions. And even where we have made direct contributions, this has been done in cooperation with others. Our collaborators have included especially Cliff Shaw, with whom we formed a team of three through the exciting period of the late fifties. But we have also worked with a great many colleagues and students at Carnegie-Mellon University.

Time permits taking up just two examples. The first is the development of the notion of a symbolic system. The second is the development of the notion of heuristic search. Both conceptions have deep significance for understanding how information is processed and how intelligence is achieved. However, they do not come close to exhausting the full scope of Artificial Intelligence, though they seem to us to be useful for exhibiting the nature of fundamental knowledge in this part of computer science.

I. SYMBOLS AND PHYSICAL SYMBOL SYSTEMS

One of the fundamental contributions to knowledge of computer science has been to explain, at a rather basic level, what symbols are. This explanation is a scientific proposition about Nature. It is empirically derived, with a long and gradual development.

Symbols lie at the root of intelligent action, which is, of course, the primary topic of Artificial Intelligence. For that matter, it is a primary question for all of Computer Science. For all information is processed by computers in the service of ends, and we measure the intelligence of a system by its ability to achieve stated ends in the face of variations, difficulties and complexities posed by the task environment. This general investment of computer science in attaining intelligence is obscured when the tasks being accomplished are limited in scope, for then the full variations in the environment can be accurately foreseen. It becomes more obvious as we extend computers to more global, complex and knowledge-intensive tasks -- as we attempt to make them our agents, capable of handling on their own the full contingencies of the natural world.

Our understanding of the systems requirements for intelligent action emerges slowly. It is composite, for no single elementary thing accounts for intelligence in all its manifestations. There is no "intelligence principle," just as there is no "vital principle" that conveys by its very nature the essence of life. But the lack of a simple deus ex machina does not imply that there are no structural requirements for intelligence. One such requirement is the ability to store and manipulate symbols. To put the scientific question, we may paraphrase the title of a famous paper by Warren McCulloch [1961]: What is a symbol, that intelligence may use it, and intelligence, that it may use a symbol?

Laws of Qualitative Structure

All sciences characterize the essential nature of the systems they study. These characterizations are invariably qualitative in nature, for they set the terms within

which more detailed knowledge can be developed. Their essence can often be captured in very short, very general statements. One might judge these general laws, due to their limited specificity, as making relatively little contribution to the sum of a science, were it not for the historical evidence that shows them to be results of the greatest importance.

The Cell Doctrine in Biology

A good example of a law of qualitative structure is the cell doctrine in biology, which states that the basic building block of all living organisms is the cell. Cells come in a large variety of forms, though they all have a nucleus surrounded by protoplasm, the whole encased by a membrane. But this internal structure was not, historically, part of the specification of the cell doctrine; it was subsequent specificity developed by intensive investigation. The cell doctrine can be conveyed almost entirely by the statement we gave above, along with some vague notions about what size a cell can be. The impact of this law on biology, however, has been tremendous, and the lost motion in the field prior to its gradual acceptance was considerable.

Plate Tectonics in Geology

Geology provides an interesting example of a qualitative structure law, interesting because it has gained acceptance in the last decade and so its rise in status is still fresh in memory. The theory of plate tectonics asserts that the surface of the globe is a collection of huge plates -- a few dozen in all -- which move (at geological speeds) against, over, and under each other into the center of the earth, where they lose their identity. The movements of the plates account for the shapes and relative

locations of the continents and oceans, for the areas of volcanic and earthquake activity, for the deep sea ridges, and so on. With a few additional particulars as to speed and size, the essential theory has been specified. It was of course not accepted until it succeeded in explaining a number of details, all of which hung together (e.g., accounting for flora, fauna, and stratification agreements between West Africa and Northeast South America). The plate tectonics theory is highly qualitative. Now that it is accepted, the whole earth seems to offer evidence for it everywhere, for we see the world in its terms.

The Germ Theory of Disease

It is little more than a century since Pasteur enunciated the germ theory of disease, a law of qualitative structure that produced a revolution in medicine. The theory proposes that most diseases are caused by the presence and multiplication in the body of tiny single-celled living organisms, and that contagion consists in the transmission of these organisms from one host to another. A large part of the elaboration of the theory consisted in identifying the organisms associated with specific diseases, describing them, and tracing their life histories. The fact that the law has many exceptions -- that many diseases are not produced by germs -- does not detract from its importance. The law tells us to look for a particular kind of cause; it does not insist that we will always find it.

The Doctrine of Atomism

The doctrine of atomism offers an interesting contrast to the three laws of qualitative structure we have just described. As it emerged from the work of Dalton,

and his demonstrations that the chemicals combined in fixed proportions, the law provided a typical example of qualitative structure: the elements are composed of small, uniform particles, differing from one element to another. But because the underlying species of atoms are so simple and limited in their variety, quantitative theories were soon formulated which assimilated all the general structure in the original qualitative hypothesis. With cells, tectonic plates, and germs, the variety of structure is so great that the underlying qualitative principle remains distinct, and its contribution to the total theory clearly discernible.

Conclusion

Laws of Qualitative Structure are seen everywhere in science. Some of our greatest scientific discoveries are to be found among them. As the examples illustrate, they often set the terms on which a whole science operates.

Physical Symbol Systems

Let us return to the topic of symbols, and define a physical symbol system. The adjective "physical" denotes two important features: (1) Such systems clearly obey the laws of physics -- they are realizable by engineered systems made of engineered components. (2) Although our use of the term "symbol" prefigures our intended interpretation, it is not restricted to human symbol systems.

A physical symbol system consists of a set of entities, called symbols, which are physical patterns that can occur as components of another type of entity called an expression (or symbol structure). Thus, a symbol structure is composed of a number of instances (or tokens) of symbols related in some physical way (such as one token

being next to another). At any instant of time the system will contain a collection of these symbol structures. Besides these structures, the system also contains a collection of processes that operate on expressions to produce other expressions: processes of creation, modification, reproduction and destruction. A physical symbol system is a machine that produces through time an evolving collection of symbol structures. Such a system exists in a world of objects wider than just these symbolic expressions themselves.

Two notions are central to this structure of expressions, symbols and objects: designation and interpretation.

> *Designation*: An expression designates an object if, given the expression, the system can either affect the object itself or behave in ways dependent on the object.

In either case, access to the object via the expression has been obtained, which is the essence of designation.

> *Interpretation*: The system can interpret an expression if the expression designates a process and if, given the expression, the system can carry out the process.

Interpretation implies a special form of dependent action: given an expression the system can perform the indicated process, which is to say, it can evoke and execute its own processes from expressions that designate them.

A system capable of designation and interpretation, in the sense just indicated, must also meet a number of additional requirements, of completeness and closure. We will have space only to mention these briefly; all of them are important and have far-reaching consequences.

(1) A symbol may be used to designate any expression whatsoever. That is, given a symbol, it is not prescribed a priori what expressions it can designate. This arbitrariness pertains only to symbols; the symbol tokens and their mutual relations determine what object is designated by a complex expression. (2) There exist expressions that designate every process of which the machine is capable. (3) There exist processes for creating any expression and for modifying any expression in arbitrary ways. (4) Expressions are stable; once created they will continue to exist until explicitly modified or deleted. (5) The number of expressions that the system can hold is essentially unbounded.

The type of system we have just defined is not unfamiliar to computer scientists. It bears a strong family resemblance to all general-purpose computers. If a symbol manipulation language, such as LISP, is taken as defining a machine, then the kinship becomes truly brotherly. Our intent in laying out such a system is not to propose something new. Just the opposite: it is to show what is now known and hypothesized about systems that satisfy such a characterization.

We can now state a general scientific hypothesis -- a law of qualitative structure for symbol systems:

> The Physical Symbol System Hypothesis: A physical symbol system has the necessary and sufficient means for general intelligent action.

By "necessary" we mean that any system that exhibits general intelligence will prove upon analysis to be a physical symbol system. By "sufficient" we mean that any physical symbol system of sufficient size can be organized further to exhibit general intelligence. By "general intelligent action" we wish to indicate the same scope of

intelligence as we see in human action: that in any real situation behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some limits of speed and complexity.

The Physical Symbol System Hypothesis clearly is a law of qualitative structure. It specifies a general class of systems within which one will find those capable of intelligent action.

This is an empirical hypothesis. We have defined a class of systems; we wish to ask whether that class accounts for a set of phenomena we find in the real world. Intelligent action is everywhere around us in the biological world, mostly in human behavior. It is a form of behavior we can recognize by its effects whether it is performed by humans or not. The hypothesis could indeed be false. Intelligent behavior is not so easy to produce that any system will exhibit it willy-nilly. Indeed, there are people whose analyses lead them to conclude either on philosophical or on scientific grounds that the hypothesis is false. Scientifically, one can attack or defend it only by bringing forth empirical evidence about the natural world.

We now need to trace the development of this hypothesis and look at the evidence for it.

Development of the Symbol System Hypothesis

A physical symbol system is an instance of a universal machine. Thus the symbol system hypothesis implies that intelligence will be realized by a universal computer. However, the hypothesis goes far beyond the argument, often made on general grounds of physical determinism, that any computation that is realizable can be realized by a universal machine, provided that it is specified. For it asserts specifically

that the intelligent machine is a symbol system, thus making a specific architectural assertion about the nature of intelligent systems. It is important to understand how this additional specificity arose.

Formal Logic

The roots of the hypothesis go back to the program of Frege and of Whitehead and Russell for formalizing logic: capturing the basic conceptual notions of mathematics in logic and putting the notions of proof and deduction on a secure footing. This effort culminated in mathematical logic -- our familiar propositional, first-order, and higher-order logics. It developed a characteristic view, often referred to as the "symbol game". Logic, and by incorporation all of mathematics, was a game played with meaningless tokens according to certain purely syntactic rules. All meaning had been purged. One had a mechanical, though permissive (we would now say nondeterministic), system about which various things could be proved. Thus progress was first made by walking away from all that seemed relevant to meaning and human symbols. We could call this the stage of formal symbol manipulation.

This general attitude is well reflected in the development of Information Theory. It was pointed out time and again that Shannon had defined a system that was useful only for communication and selection, and which had nothing to do with meaning. Regrets were expressed that such a general name as "information theory" had been given to the field, and attempts were made to rechristen it as the Theory of Selective Information -- to no avail of course.

Turing Machines and the Digital Computer

The development of the first digital computers and of automata theory, starting with Turing's own work in the '30s, can be treated together. They agree in their view of what is essential. Let us use Turing's own model, for it shows the features well.

A Turing machine consists of two memories: an unbounded tape and a finite state control. The tape holds data, i.e., the famous zeroes and ones. The machine has a very small set of proper operations -- read, write and scan operations -- on the tape. The read operation is not a data operation, but provides conditional branching to a control state as a function of the data under the read head. As we all know, this model contains the essentials of all computers, in terms of what it can do, though other computers with different memories and operations might carry out the same computations with different requirements of space and time. In particular, the model of a Turing machine contains within it the notions both of what cannot be computed and of universal machines -- computers that can do anything that can be done by any machine.

We should marvel that two of our deepest insights into information processing were achieved in the thirties, before modern computers came into being. It is a tribute to the genius of Alan Turing. It is also a tribute to the development of mathematical logic at the time, and testimony to the depth of Computer Science's obligation to it. Concurrently with Turing's work appeared the work of the logicians Emil Post and (independently) Alonzo Church. Starting from independent notions of logistic systems (Post productions and recursive functions, respectively) they arrived at analogous results on undecidability and universality -- results that were soon shown to imply that all three systems were equivalent. Indeed, the convergence of all these attempts

to define the most general class of information processing systems provides some of the force of our conviction that we have captured the essentials of information processing in these models.

In none of these systems is there, on the surface, a concept of the symbol as something that designates. The data are regarded as just strings of zeroes and ones -- indeed that data be inert is essential to the reduction of computation to physical process. The finite state control system was always viewed as a small controller, and logical games were played to see how small a state system could be used without destroying the universality of the machine. No games, as far as we can tell, were ever played to add new states dynamically to the finite control -- to think of the control memory as holding the bulk of the system's knowledge. What was accomplished at this stage was half the principle of interpretation -- showing that a machine could be run from a description. Thus, this is the stage of automatic formal symbol manipulation.

The Stored Program Concept

With the development of the second generation of electronic machines in the mid-forties (after the Eniac) came the stored program concept. This was rightfully hailed as a milestone, both conceptually and practically. Programs now can be data, and can be operated on as data. This capability is, of course, already implicit in the model of Turing: the descriptions are on the very same tape as the data. Yet the idea was realized only when machines acquired enough memory to make it practicable to locate actual programs in some internal place. After all, the Eniac had only twenty registers.

The stored program concept embodies the second half of the interpretation

principle, the part that says that the system's own data can be interpreted. But it does not yet contain the notion of designation -- of the physical relation that underlies meaning.

List Processing

The next step, taken in 1956, was list processing. The contents of the data structures were now symbols, in the sense of our physical symbol system: patterns that designated, that had referents. Lists held addresses which permitted access to other lists -- thus the notion of list structures. That this was a new view was demonstrated to us many times in the early days of list processing when colleagues would ask where the data were -- that is, which list finally held the collections of bits that were the content of the system. They found it strange that there were no such bits, there were only symbols that designated yet other symbol structures.

List processing is simultaneously three things in the development of computer science. (1) It is the creation of a genuine dynamic memory structure in a machine that had heretofore been perceived as having fixed structure. It added to our ensemble of operations those that built and modified structure in addition to those that replaced and changed content. (2) It was an early demonstration of the basic abstraction that a computer consists of a set of data types and a set of operations proper to these data types, so that a computational system should employ whatever data types are appropriate to the application, independent of the underlying machine. (3) List processing produced a model of designation, thus defining symbol manipulation in the sense in which we use this concept in Computer Science today.

As often occurs, the practice of the time already anticipated all the elements of

list processing: addresses are obviously used to gain access, the drum machines used linked programs (so called one-plus-one addressing), and so on. But the conception of list processing as an abstraction created a new world in which designation and dynamic symbolic structure were the defining characteristics. The embedding of the early list processing systems in languages the (IPLs, LISP) is often decried as having been a barrier to the diffusion of list processing techniques throughout programing practice; but it was the vehicle that held the abstraction together.

LISP

One more step is worth noting: McCarthy's creation of LISP in 1959-60 [McCarthy, 1960]. It completed the act of abstraction, lifting list structures out of their embedding in concrete machines, creating a new formal system with S-expressions, which could be shown to be equivalent to the other universal schemes of computation.

Conclusion

That the concept of the designating symbol and symbol manipulation does not emerge until the mid-fifties does not mean that the earlier steps were either inessential or less important. The total concept is the join of computability, physical realizability (and by multiple technologies), universality, the symbolic representation of processes (i.e., interpretability), and, finally, symbolic structure and designation. Each of the steps provided an essential part of the whole.

The first step in this chain, authored by Turing, is theoretically motivated, but the others all have deep empirical roots. We have been led by the evolution of the