2

Computing in Cognitive Science

Zenon W. Pylyshyn

Posner, Foundation of Cognitive Science, 1989

Nobody doubts that computers have had a profound influence on the study of human cognition. The very existence of a discipline called cognitive science is a tribute to this influence. One of the principal characteristics that distinguishes cognitive science from more traditional studies of cognition within psychology is the extent to which it has been influenced by both the ideas and the techniques of *computing*. It may come as a surprise to the outsider then to discover that there is no unanimity within cognitive science on either the nature (and in some cases the desirability) of the influence or what computing *is*, or at least on its essential character, as it pertains to cognitive science.

In this chapter I comment on both these questions. The first question brings us to a discussion of the role of computing in our understanding of human (and perhaps animal) cognition. I examine a variety of such roles—from the instrumental use of computers to express theories, through its role as a source of ideas, to the bold empirical claim that cognition is quite literally a species of computing. The latter position (which gets us into a discussion of what I call the *strong equivalence* thesis) cannot even be begun to be addressed until we have a much clearer understanding of what we mean by the term *computing*—that is, what family of processes we intend to cover by that term. This is the most contentious of the topics I cover, but one that cannot be avoided; an understanding of the assumptions underlying the discipline is a prerequisite to understanding recent proposals for a redirection of the goals of cognitive science (see, for example, chapters 4 and 8).

In the final section of this chapter, I examine the methodologies available for validating computational models as *strong* models of cognitive processes. Although many of these techniques are also discussed in other chapters of this book (for example, chapters 1 and 7), my discussion is intended to show how some of these methods relate to the notion of strong equivalence of processes.

Let us note that the view that computing is relevant to understanding cognition, or intelligent behavior in general, goes as far back as the idea of computing itself. Turing's (1937) original paper on computability contains a section in which Turing attempts to provide some intuitive motivation for his notion of a mechanically "effective procedure" by looking at what a mathematician does in the course of solving mathematical problems and distilling this process to its essentials. Later Turing (1950) argued that a properly programmed computer could in principle exhibit intelligent behavior. The argument rests on Turing's own discovery of the existence of a universal Turing machine, an abstract automation that can imitate any other formally specifiable computer. The relevance of the universal machine to cognitive science is raised briefly later.

Computers are relevant to cognition in many ways. Newell (1970; see also Newell 1973a) has discussed a range of views of the possible relation between computing and cognition. These vary all the way from the view that computers provide an interesting new metaphor, to the view—which I defend—that cognition is literally a species of computing, carried out in a particular type of biological mechanism. In the following I sketch two of the major ways in which computing is relevant to the study of cognition. Later I elaborate and defend both these general propositions and argue that they have been decisive in the development of cognitive science, even though there have been many arguments concerning the details—and even the foundational assumptions—behind them.

At the most abstract level the class of mechanisms called *computers* are the only known mechanisms that are sufficiently plastic in their behavior to match the plasticity of human cognition. They are also the only known mechanism capable of producing behavior that can be described as *knowledge dependent*. Because of such properties *computing* remains the primary candidate for meeting the dual needs of explaining cognition in mechanism terms and accounting for certain otherwise problematic aspects of cognition—in particular the fact that behavior can be systematically influenced by inducing differences in beliefs or goals.

At a more concrete level computers provide a way to deal with a number of problems that plague the attempt to understand cognition. Among them are the complexity of the processes underlying cognition and the need for a theory that bridges the gap from internal processing to actual instances of behavior. Such a theory is sometimes said to meet the *sufficiency condition*. This condition imposes a particularly stringent requirement on measures of the adequacy of a theory. It also forces the theorist to explicitly confront certain issues that could otherwise be taken for granted or presupposed. Chief among them are the architecture-process distinction (and the nature of the cognitive architecture) and the closely related question of the control structure underlying cognitive processing.

2.1 What is Computing?

Some Background: Formalisms, Symbols, and Mechanisms

The possibility of imitating life by artifact has intrigued people throughout history. But only in the second half of this century has the possibility of using the special type of artifact that we call a computer been considered seriously as a means of understanding mental phenomena. What is different about this latest interest is that the focus is not primarily on the imitation of movements (as was the case with early clockwork mechanisms) but on the imitation of certain unobservable internal processes. This notion only became conceivable with the gradual emergence, in several disparate areas of intellectual development, of a certain way of understanding mechanisms. This new and more abstract notion of mechanism is entirely divorced from the old-style "mechanical" considerations (such as those that preoccupied Descartes and that Chomsky has characterized as *contact mechanics*) and is concerned only with abstractly defined operations such as storing, retrieving, and alterating tokens of symbolic codes.

This notion of mechanism arose in conjunction with attempts to develop a completely formal, content-free foundation for mathematics. The Hilbert program was one of the most ambitious attempts to build up mathematics by purely formal means; without regard to questions of what the formalism was about. Some of this enterprise succeeded in the work of Frege and of Russell and Whitehead. On the other hand one of the greatest intellectual achievements of our age was the demonstration by purely formal means that the ultimate goal of complete formalization was in principle not achievable (this was done originally by Godel and subsequently by Turing, Church, Post, and others; see the collection of papers in Davis 1965).

The same work that provided demonstrations of particular in-principle limitations of formalization also provided demonstrations of its universality. Thus Alan Turing, Emil Post, and Alonzo Church independently developed distinct formalisms that they showed were complete in the sense that they were powerful enough to formally (that is; "mechanically") generate all sequences of expressions that could be interpreted as proofs and hence could generate all provable theorems of logic. In Turing's case this took the form of showing that there exists a universal mechanism, a particular Turing machine called the universal machine (UM), that could simulate any mechanism describable in its formalism. It does this by accepting a description of the mechanism to be simulated and then carries out a procedure whose input/output behavior is identical to that which would have been generated by the machine whose description it was given. We say that the UM computes the *same function* as the target machine, where "same function" means the same input/output pairs or the same *extension* of the function. There is no requirement that UM carry out the same steps as the target machine. That would be a stronger sense of equivalence.

What is interesting about the latter work, from our point of view, is that to derive such results (concerning the universality and incompleteness of certain formal systems), it was necessary to understand the notions of proof and of deduction in a formal system in terms of the manipulation of symbol tokens or marks on a piece of paper, where the manipulation was specified "mechanically" in a way that was entirely independent of how the symbols might be interpreted. Logic became a game played with meaningless symbol tokens according to certain formal rules (that is, syntactic rules).

It was the development of the notion of universality of formal mechanism, first introduced in the work on foundations of mathematics in the 1930s, that provided the initial impetus for viewing mind as a symbol-processing system. Universality implies that a formal symbol processing mechanism can produce *any* arbitrary input/output function that we can specify in sufficient detail. Put in more familiar terms, a universal machine can be *programmed* to compute any formally specified function. This extreme plasticity in behavior is one of the reasons why computers have from the very beginning been viewed as artifacts that might be capable of exhibiting intelligence. Many people who were not familiar with this basic idea have misunderstood the capacity of machines. For example, the Gestalt psychologist Wolfgang Kohler (1947) viewed machines as too rigid to serve as models of mental activity. The latter, he claimed, are governed by what he called *dynamic factors,*—an example of which are self-distributing field effects, such as the effects that cause magnetic fields to be redistributed when we introduce new pieces of metal—as opposed to *topographical factors,* which are structurally rigid. He wrote (Kohler 1947, p. 65):

> To the degree to which topographical conditions are rigidly given, and not to be changed by dynamic factors, their existence means the exclusion of certain forms of function, and the restriction of the processes to the possibilities compatible with those conditions. . . . This extreme relation between dynamic factors and imposed topographical conditions is almost entirely realized in typical machines . . . we do not construct machines in which dynamic factors are the main determinants of the form of operation.

That computers violate this claim is one of their most important and unique characteristics. Their topographic structure is completely rigid, yet they are capable of maximal plasticity of function. It is this very property that led Turing to speculate that computers would be capable in principle of exhibiting intelligent behavior. For example, he devoted an important early philosophical paper (Turing 1950) to an exposition of this idea. Turing argued that a computer could in principle be made to exhibit intelligent activity to an arbitrary degree. He claimed that a machine should qualify as being intelligent if it could successfully play the "imitation game"—that is, fool a human observer, with whom it could communicate only through a keyboard and terminal, so that the observer could not discriminate between it and another person. The possibility of a computer being able to successfully pass what become known as the *Turing test* is based entirely on the recognition of the plasticity of behavior entailed by symbolic systems, which can be *programmed* to behave according to any finitely specifiable function.

Devices that we call computers now come in a wide variety of forms—most of which appear quite different from the one that Turing developed in his mathematical analysis. It is appropriate to ask then what makes a system a computer. This is a particularly relevant question inasmuch as a working hypothesis of much of cognitive science is that the mind is literally a type of computer. One might begin by asking, In virtue of what property does the Turing machine achieve the universality or the programmability that recommended it as a model of intelligence?

Newell (1980) provides an interesting insight into one characteristic that is essential for a device to be universal or programmable. For a mechanism to be universal, its inputs must be partitioned into two distinct components, one of which is assigned a privileged interpretation as instructions or as a specification of some particular input/output function, and the other of which is treated as the proper input to that function. Such a partition is essential for defining a universal Turing machine. Thus there can only be arbitrary plasticity of behavior if some of the inputs and outputs of the system are *interpreted* (or, as Newell puts it, if they have the power to "designate" something extrinsic).

Designation is indeed one of the core ideas of computing. In computers symbols may designate in several ways: they can provide access to other symbols, they can cause an interpreter to perform the action designated by that symbol, or they may designate other extrinsic things. For example, they may designate abstract objects called numbers, or they may designate objects of reasoning (for example, objects in the world or in the imagination, propositions, predicates, and so on), or they may even designate goals. Indeed because what symbols designate need not exist (for example, unicorns or the pot of gold at the end of the rainbow) the very notion of designation, meaning "referring to," is problematic inasmuch as people usually understand "refer" to apply only when there exists a thing being referred to. That is why we usually talk about the relation of symbols and what they symbolize as semantics, or we speak of the meaning of a symbol. In any case *semantics* and *meaning* are relevant terms used to describe properties of states of computers (and people) but not of many other complex systems that are not functioning as computers (for example, the Andromeda galaxy).

Systems that have traditionally been called computers (for example, the Turing machine) share a number of properties. The view that certain of these properties are constitutive of computing (and consequently that they are also constitutive of cognition, insofar as cognition is a species of computing) is called the *classical view* (after Fodor and Pylyshyn 1988). In the next section I consider some of these properties, acknowledging that this view is by no means unanimously held among cognitive scientists (see, for example, chapter 4).

The Classical View of Computing and Cognition

In Turing's original theoretical machine and in every real digital computer, a distinction is made between the *processor* and the *memory*. The processor "writes" symbolic expressions into memory, alters them, and "reads" them. Reading certain of these symbols causes specified actions to occur, which may change other symbols. The memory may consist of a tape, a set of registers, or any form of working storage. The expressions that are written are complex symbols that are made up of simpler symbols, just the way sentences are complex symbols made up of simpler symbols in a systematic way. The processor (or, in the case of logic, the rules of inference) then transforms the expressions into new expressions in a special kind of systematic way. The *way* such symbolic expressions are transformed in a classical computer is very important. As has already been mentioned, the symbolic expressions have a semantics, that is, they are *codes* for something, or they *mean* something. Therefore the transformations of the expressions are designed to coherently maintain this meaning or to ensure that the expressions continue to make sense when semantically interpreted in a consistent way.

For example, if the expressions are numerals like 19, 1011, XIX, or expressions in some other numeral notations, they usually serve as *codes* for *numbers*. In that case when the computer transforms these expressions, they might refer to different numbers. If you can arrange for the computer to transform them systematically in the appropriate way, the transformations can correspond to useful mathematical operations such as addition or multiplication. Consider an abacus. Patterns of beads represent numbers. People learn rules for transforming these patterns of beads in such a way that the semantic interpretation of before-and-after pairs corresponds to a useful mathematical function. But there is nothing intrinsically mathematical about the rules themselves; they are just rules for moving beads around. What makes the rules useful for doing mathematics is that we are assured of a certain continuing correspondence between the formal or syntactic patterns of beads and mathematical objects (such as numbers). The way such a correspondence can be assured is illustrated by an example in the next section.

In scientific computing, as well as in the history of computer applications up to the 1970s, the most frequently encountered domain of representation was doubtlessly that of numbers, and consequently the most common transformations over expressions were those that mirror mathematical functions over numbers. But if the symbolic expressions were codes for propositions or beliefs or knowledge, as they might be if they were expressions in some symbolic logic, then the computer might transform them in ways corresponding to proofs or inferences, or perhaps to a sequence of "thoughts" that occur during commonsense reasoning. The important thing is that, according to the classical view, certain kinds of systems, including both minds and computers, operate on *representations* that take the form of symbolic codes.

There is one more important property that such symbolic codes must have, according to the classical view. In classical symbol systems the meaning of a complex expression depends in a systematic way on the meaning of its parts (or *constituents*). This is the way ordinary language, formal logic, and even the number system works, and there are good reasons for believing that they *must* work that way in both practical computing and in modeling cognition. In the case of cognition these reasons have to do with the *productivity* and the *systematicity* of thought and reasoning, two issues discussed at length in Fodor and Pylyshyn 1988.

So to summarize, the classical view assumes that both computers and minds have at least the following three distinct levels of organization:

1. *The semantic level (or knowledge) level*[1] At this level we explain why people, or appropriately programmed computers, do certain things by saying what they know and what their goals are and by showing that these are connected in certain meaningful or even rational ways.

2. *The symbol level* The semantic content of knowledge and goals is assumed to be encoded by symbolic expressions. Such structured expressions have parts, each of which also encodes some semantic content. The codes and their structure, as well as the regularities by which they are manipulated, are another level of organization of the system.

3. *The physical (or biological) level* For the entire system to run, it has to be realized in some physical form. The structure and the principles by which the physical object functions correspond to the physical or the biological level.

This three-level organization defines what I call the *classical computational* or *cognitive architecture*.

To illustrate the claim that there are different principles that apply at each of these levels, consider the following example. Suppose you have a calculator with a square root button. If you want to explain why it gives strange answers or fails to work when the batteries are low or when you cut one of the wires in it or when the temperature is too low, you have to refer to physical properties of the calculator, the physical level. If you want to explain why certain rounding errors occur in the lower-order digits of the answer, or why it takes longer to compute the answer to some problems than to others, you have to refer to how numbers are symbolically encoded and to what particular sequence of transformations of these symbolic expressions occurs (that is, to the algorithm used). This is an explanation at the symbol level. But then if you want to show that the algorithm will always give the correct answer, you have to refer to facts and theorems of number theory, that is, to the semantics of the symbols.

One might ask how it is possible for symbolic expressions and rules to keep maintaining their semantic interpretation, to keep the semantics of the expressions coherent. It is one of the important discoveries of formal logic that one can specify rules that operate on symbolic expressions in such a way that the sequence of expressions always corresponds to a proof. In computing (and in cognitive science generally) one is interested in not only logical, or truth-preserving, sequences but also sequences that preserve such semantic properties as those exhibited in heuristic or goal-directed reasoning.

The following numerical example shows how one can define an operation over symbolic expressions and a semantic mapping (which I designate *SF*) from symbols to numbers in such a way that the operation can be consistently interpreted as addition. To emphasize the generality of the following example (so that, for example, it could apply to some system other than a conventional computer), I present it in its most abstract form. Suppose we have a certain instantiation function *IF* from equivalence classes of physical states of a certain system (perhaps only the parts of the system called its memory registers) to symbolic expressions. For concreteness let us say that the expressions consist of the atomic symbols o and x arranged in strings of arbitrary length. In this example then the states of the memory registers would correspond to such expressions as o, x, ox, xo, xx, oox, oxo, oxx, xoo, xox, xxo, xxx, xooo, and so on. Each of these expressions corresponds to some possible state of each of the machine's memory registers.[2]

Let us further suppose that when a certain pattern (which I designate by #) occurs in a portion of the machine called its instruction register, the machine's memory registers change states according to a certain specifiable regularity. For example, when the portion of the machine we call register 1 is in the state that maps onto the string xox, and register 2 is in the state that maps onto the string xxo, then register 3 changes its state from whatever it was to the state that corresponds to the string xoxx.

This sort of regularity might conceivably be used to represent addition of numbers, provided that we adopt an appropriate semantic function *SF* and that the regularity meets certain requirements. In this case the required semantic function is easy to define—it happens to be the function that maps strings of o's and x's onto numbers, using the familiar binary number system. In defining the *SF* formally, moreover, we provide a way of stating the requirements that the regularity must meet if it is to be consistently interpretable as addition of numbers.

Before defining the *SF*, however, it is necessary to give a formal definition of the set of expressions consisting of x's and o's. Because we are not assuming any bound on the number of states that a register can take (and hence on the length of the strings of x's and o's), the definition of the strings must be given recursively as

(1) o is a string,

(2) x is a string,

(3) if T is a string, then so is To (that is, string T followed by o).

(4) if T is a string, then so is Tx (that is, string T followed by x).

A simpler way to express (1) through (4) is in Backus-Nauer form as T ::= o|x|To|Tx, where ::= means "is defined to be" and | means "or."

Using the definition of the strings; the semantic function can then be defined recursively as

(1) $SF(\mathrm{o}) = 0$	(the semantic interpretation of o is the number zero),
(2) $SF(\mathrm{x}) = 1$	(the semantic interpretation of x is the number one),
(3) $SF(\mathrm{To}) = 2{*}SF(\mathrm{T})$	(the semantic interpretation of a string T followed by o is twice the semantic interpretation of T alone),
(4) $SF(\mathrm{Tx}) = 2{*}SF(\mathrm{T}) + 1$	(the semantic interpretation of a string T followed by x is twice the semantic interpretation of T alone plus one).

This constitutes an example of a semantic function, defined recursively on the structure of the strings of symbols. It is analogous to Tarski's method for defining the semantics of sentences in some formal calculus in terms of their combinatorial properties. This mapping function is nontrivial. In fact it defines the semantic interpretation of a *place-value* numeral notation.

For this semantic function to be useful, however, there must be regularities in the state transitions in the computer that correspond to mathematical operations defined over the interpretations of the symbols in the intended domain. In other words there must be state transitions that *preserve the intended interpretation SF*. One such regularity, which was associated with the occurrence of # in the instruction register, has already been proposed. For # to correspond to addition (or alternatively for it to be consistently interpretable as addition), state transitions must preserve the semantic interpretation of the symbol strings under the mathematically defined operation of addition (defined, say, in terms of

Peano's axioms). In other words something like the following must be true:

If the computer is in the state characterized by the description

1. Register 1 "contains" (or *IF* maps it onto) string T_1,
2. Register 2 "contains" (or *IF* maps it onto) string T_2,
3. The instruction register "contains" (or *IF* maps it onto) #;

then the computer goes into the state characterized by

4. Register 3 "contains" (or *IF* maps it onto) the string T_3, where the relation $SF(T_3) = SF(T_1) + SF(T_2)$ holds.

In other words the (mathematically defined) sum of the semantic interpretations of the two register states must always correspond to the semantic interpretation of the state of the third register. Note that the interpretation is in the abstract domain of *numbers*, where operations such as addition are mathematically defined, whereas the symbols being interpreted (the domain of the *SF* function) are functional states, defined by *IF* as equivalence classes of physical states of the computer.

These ideas and distinctions arise in clear form in the case of conventional computers. They apply equally however, in the case of cognition, even though our subjective experience suggests that what is going on in the mind may be different. The empirical facts and the requirement of explanatory adequacy, however, demand all three distinct levels (physical, symbolic, and semantic) in the case of human cognition, just as we needed them in the computer case. Although the arguments are beyond the scope of this chapter (see Pylyshyn 1984), it appears that to explain intelligent human behavior, we need to appeal to all three levels of organization.

1. We need the knowledge level to explain why certain goals and beliefs tend to lead to certain behaviors, and why the behaviors can be changed in rational ways when new beliefs are added by telling things. For example, to explain why I am sitting here at this moment striking these particular keys of my keyboard, one must mention my beliefs about cognitive science, my beliefs about what will become of this manuscript, and my general goals of conveying true information to those who might read the book in which this chapter is intended to appear. Without this level we could not capture such regularities as, for example, the fact that if I were to have the belief that publication of the book had been canceled, I would exhibit quite different behavior *regardless of the particular "stimuli" that might have led me to have this (presumably false) belief*. This sort of semantically characterizable malleability of behavior is referred to as *cognitive penetrability* and has been used as diagnostic of behavior requiring knowledge-level explanation (for more on this see the section 2.3, as well as Pylyshyn 1984).

2. We need the symbol level to explain such things as why some tasks take longer or result in more errors than other tasks. Information-

processing psychology is full of examples of discovering that the form of the representation makes a difference to their behavior in experiments. For example, in problem-solving experiments it makes a difference whether subjects encode the fact that all the objects in a box are red or the equivalent fact that none of the objects is blue.

3. We obviously need the biological level to explain such things as the effects of drugs or jet lag or brain damage on behavior. It is also possible that we may need the biological level to explain other things as well, such as possibly the nature of cognitive development or maturation or psychopathology, and perhaps some changes that are now called learning; exactly what facts fall at each of the three levels remains to a large extent an open empirical question.

Objections to the Classical View

There has always been opposition to the view that we have symbol structures in our heads. The idea that the brain thinks by writing symbols and reading them sounds absurd to many. It suggests to some people that we have been influenced too much by the way current electronic computers work. The basic source of uneasiness seems to come from the fact that we do not have the subjective experience that we are manipulating symbols. But subjective experience has been a notoriously misleading source of evidence for what goes on in the mind. Research in human information processing reveals countless processes that clearly must be occurring (for example, parsing, inference) of which we have little or no subjective awareness.

Arguments for the necessity of positing symbol structures in human reasoning—for a "language of thought"—are given elsewhere (Fodor 1975, Pylyshyn 1984, Fodor and Pylyshyn 1988). Details of these arguments are beyond the scope of this chapter. For the present purposes, the following summary will suffice.

If the knowledge-level description is correct, then we have to explain how it is possible for a physical system, like a human being, to behave in ways that correspond to the knowledge-level principles while at the same time being governed by physical laws. The content of knowledge is related to the state of a system by a *semantic* relation, which is quite a different relation from the ones that appear on natural laws (for one thing the object of the relation need not exist). At present there is only one candidate explanation for how knowledge-level principles can be causally realized, and that is the one that builds on a set of ideas going back to the insights of Boole, Hilbert, Turing, Frege, and other logicians. It says that knowledge is *encoded* by a system of symbolic codes, which themselves are physically realized, and that it is the physical properties of the codes that cause the behaviors in question.

What Fodor and Pylyshyn (1988) have added to this general statement is an argument that the system of codes must be *structured* much like a language (as indeed it is in the various logical calculi that have been developed). The argument stems in part from the observation that both representational capacity and inferential capacity in intelligent systems is *systematic*. Representational or inferential capacities are not punctate—they do not occur in isolation; the capacity for representing certain things or for drawing certain inferences goes along with the capacity for representing other things and for drawing other inferences. For example, an intelligent system that is capable of representing certain situations (for example, that John loves Mary, or that a small red ball is in a large blue box) must also be capable—whether or not this capacity is exercised—of representing other situations involving the same conceptual components (for example, that Mary loves John or that a large blue ball is in small red box). Similarly any intelligent system that can draw certain inferences (for example, can infer from knowing that it is sunny and warm and humid that it is sunny; that is, infer P from P and Q and R) can also draw other related inferences (for example, can infer from knowing that it is sunny and warm that it is sunny; that is, infer P from P and Q).

This sort of systematicity follows automatically from the use of structured symbolic expressions to represent knowledge and to serve as the basis for inference. In other words it is a side effect of a classical architecture.[3] In contrast it is a property that must be stipulated and enforced by the theorist (that is, it is a free empirical parameter) in other nonsymbolic architectures, such as the so-called connectionist architectures.

It must be stressed that at present there exists no alternative to what Newell (1980) has called the *physical symbol system* assumption for dealing with reasoning in a mechanical way, even though there are many speculative discussions of how one might eventually be able to do without symbols. Therefore even if one does not accept the various arguments that have been given for the ultimate necessity of symbol structures, the rational strategy is to continue with the classical assumption until some better alternative comes along. At least that is the strategy adopted in every other mature science.

2.2 Computational Methodologies in Cognitive Science: The High Road and the Low Road

As I have already suggested, computers can enter into the detailed process of constructing models of cognitive processes at several levels. In practice the more fine-grained the correspondence match, the narrower the range of phenomena the model is able to cover. For this reason experimental psychologists, who have traditionally been more concerned with models that can be tested in quite specific detail against laboratory data, have generally worked with models that are relatively narrow in scope. On the other hand investigators working within the artificial intelligence tradition have been more concerned with explaining the general abilities or capacities in question and postponing the detailed empirical validation of the mechanisms and algorithms actually used in the model. These have sometimes been referred to as the "low road" and the "high road," respectively, to understanding cognitive processes. They represent different strategies for arriving at the same ultimate end: modeling human cognitive processes.[4]

David Marr was one of the most influential champions of the high road, or at least of the strategy that begins at the high end of the road. He has proposed that there are three levels at which cognitive processes may be studied. He referred to these as the level of the *computation*, the level of the *algorithm*, and the level of the *mechanism*. A theory at the first level was called a type I theory. Although the notion of a type I theory is not very well defined, Marr did give some examples, chiefly from his own work or that of his colleagues.

Any domain that has a closed formal characterization of the task or of the input/output function being carried out has a type I theory. Frequently cited examples involve the recovery of 3-D structure from a variety of types of visual cues. Thus, for example, there are at least partial theories concerning what is entailed in recovering structure from motion, stereopsis, shading, or contour information. Such theories give a precise characterization of the conditions under which the "inverse mapping" from the data in question (for example, motion of points or contours on a 2-D surface) to a 3-D structure is possible, and they formally characterize the mapping. Such theories invariably rely on recognizing certain "natural constraints" that exist in the world and are exploited by the visual system in recovering the 3-D structure.

In those cases in which there is a type I theory of some particular cognitive skill, it might be possible to determine the conditions under which that skill will succeed or fail to accomplish some particular task. For example, if we had a mathematical characterization of the relations between certain features of the light and the percepts that they engendered (that is, a type I theory of certain aspects of visual perception), then we might be able to relate the light features in question to the scene layout (via projective geometry) and determine the conditions under which perception mediated by those features would be veridical.

This in fact is what was done in modeling such processes as those involved in the perception of form from motion (Ullman 1979), of surface orientation from texture (Stevens 1981), or of stereopsis (Marr and Poggio 1979). In the first of these, for example, Ullman showed mathematically that the unambiguous recovery of 3-D shape from the motion of certain visual features on the retina (for example, random dots in the case of the kinetic depth effect studied by Wallach and O'Connell (1953)) can only be done if certain conditions are met. The mathematical function relating moving proximal features to the 3-D scene from which the features are projected is unique only if (a) there are enough distinct views and distinct features (three views and four features for orthographic projection or two views and five features for perspective projection), and (b) if the process is constrained in the possible interpretations it considers. Without (b) a unique function is not possible because the same proximal feature movements can originate from arbitrarily many different distal configurations. If the interpretation is constrained by what Ullman calls the rigidity assumption, however, then a unique interpretation is possible in very nearly just those cases where people give the true interpretation. The constraint is that the process attempts to provide an interpretation of the features as originating from points on a rigid body in motion and fails to provide any interpretation if that is not possible—it does not consider other logically possible interpretations. Although this is not yet a completely adequate type I theory (for example, it fails for biological motion, such as studied by Johansson (1975), and for perceived elastic deformations), it provides an original *computational* account of the kinetic depth effect.

Note that such a mathematical result is not based on a detailed study of the process of human perception, only on the fact that it has a certain capacity, namely, the capacity to perceive a unique 3-D structure from the motion of certain feature points (that is, on the existence of the kinetic depth effect). The mathematical result tells us the conditions under which such an accomplishment is possible. Thus it tells us something about the intrinsic requirements of that task; requirements that the visual system must somehow meet. In Ullman's case the function was also described in a constructive manner—that is, in a manner that allowed it to be computed from the sort of information that is available to a computer equipped with appropriate transducers. The latter property is also an important part of the computationalist program. Of course how the human visual system does in fact compute that function is a question whose answer depends on further empirical considerations. Notice, however, that simply knowing some of the properties of the function that the visual system computes allows one to understand why perception is generally veridical even though, contrary to Gibson, we know that the step from activating sensors to perception involves a fallible process (an inferencelike process that, however, is insensitive to general knowledge of the world). The reason it is generally veridical is that the conditions under which this quasi-inferential inverse mapping is valid, are ones that happen in fact to be frequently met in our kind of world—that is, the rigidity assumption is generally true, at least to a first approximation, in our world (though it may well not be generally true in, say, the world inhabited by fish).

What Marr was advocating is a special case of a top-down research strategy, wherein one proceeds by attempting to discover the broader outlines of a problem domain before solving some of the detailed subproblems. This sort of approach is practiced systematically in computer science, where—sometimes under the name "structured programming"—it is considered the strategy of choice in the design of computer systems. Consequently it is the strategy that characterizes artificial intelligence approaches to understanding cognition. Marr went even further to advocate that one should not worry about developing a system that exhibits the performance in question until one has at least attempted to develop a theory of the task (a type I theory), and consequently that one should work first in domains (such as perhaps vision) that lend themselves to a type I theory, rather than in domains like commonsense reasoning where there may not be such a theory. He argued that if one begins by hypothesizing a particular algorithm used by an organism without first understanding exactly what the algorithm is supposed to be computing, one runs the danger of simply mimicking fragments of behavior without understanding its principles or the goals that the behavior is satisfying.[5] This is similar to Chomsky and others' methodological injunction not to hypothesize learning mechanisms for the acquisition of certain skills until one has a good theory of the steady-state skill itself.

Although few people in cognitive science take a position as extreme as Marr's, there continue to be differences in style of approach in cognitive science research. There are differences between people who are concerned with generality and with the search for general principles, as opposed to those who wish to account for experimental variance. There are also differences between approaches that place top priority on the sufficiency criterion, and hence construct working programs that cover some domain of skill, as opposed to those who are concerned with deciding between one or two general options (for example, deciding whether a certain phenomenon, say, the recognition that a stimulus is a member of a previously memorized set, is the result of a parallel search, a serial self-terminating search, or a serial exhaustive search).

To some extent which of these strategies is followed depends on the area of research or the particular empirical phenomena being investigated. Thus the study of early vision is frequently pursued by attempting to implement algorithms and explore their entailments. Problems associated with language understanding and discourse processes are often pursued within that tradition as well. On the other hand the study of learning, memory, and problem solving has been successfully approached by both the high road and the low road. Insofar as the empirical phenomenon of interest can be attributed to some particular isolated mechanism or process, it may be possible to establish empirically the nature of that process by carrying out a series of experiments. But to the extent that the phenomenon arises from the interaction of many processes, it may not be possible to explain it without a more general model that embodies the entire set of relevant processes. The pitfalls of attempting to answer general questions by isolating effects and by attributing phenomena to particular features of the process have been well documented by Newell (1973c), who argued (as the title of his paper says) that "You can't play twenty questions with nature and win."

Despite these pitfalls it appears to be possible to study certain specific subprocesses in detail in some cases without building large-scale models. Indeed the area of cognitive science sometimes known as information-processing psychology has been dominated by the empirical validation of minimodels. The analysis of cognitive processes into stages using mental chronometry (see, for example, Posner 1978) is a good example. The methodology for such fine-grained analysis of cognitive processes is discussed in chapter 7. To take a specific example, it appears to be possible to study aspects of short-term memory without developing large-scale models (see, for example, Sperling 1967). Indeed because the models are so small-scale, theorizing in this area has typically not involved implementing models in the form of computer programs.

But even here one must be cautious in concluding that there is nothing to be gained by actually implementing small-scale models. Newell (1973b) provided an excellent example of how the attempt to design a computer model to account for certain empirical phenomena of short-term memory can itself lead to new hypotheses that might otherwise not have arisen. In that particular example the attempt to implement a model in an independently motivated architecture led to a particular way of accounting for Sternberg's (1970) short-term-memory–scanning results, the so-called decoding hypothesis, which involves neither exhaustive nor self-terminating search (the two options that had been under investigation in much of the experimental research) and contains both parallel and serial components (two options that also had been assumed to exhaust the possibilities).

The Control Issue

The commitment to the construction of a model meeting the sufficiency condition, that is, one that actually generates token behaviors, forces one to confront the problem of how and under what conditions the internal representations and the rules are invoked in the course of generating actions. These are question that concern the *control* of the process. Although they form a central topic of study in computer science, they were virtually never raised in a cognitive psychology that was not constrained by computational sufficiency. Indeed one of the main criticisms that was leveled against the early work by cognitive psychologists like Tolman was that their theories dealt only with the organism's representations (mental maps), but had no way of saying how these representations would lead to action. For example, in an early critique of this cognitivist approach Guthrie (1935, p. 172) wrote, "In his concern for what goes on in the rat's mind, Tolman has neglected to predict what the rat will do. So far as the theory is concerned, the rat is left buried in thought; if he gets to the food-box at the end, that is his concern, not the concern of the theory."

There is much more to understanding control structures than knowing how operations are sequenced. We are so used to thinking of procedures as sequences of instructions that continue their fixed course until some conditional branch operation detects a specified condition that alternative organizations do not readily spring to mind. Yet this is just one type of organization of control—one in which control is *passed* along a linear sequence from operation to operation; when one operation finishes it passes control to the next operation in line. In computer science and artificial intelligence, however, there is a great deal of interest in very different control schemes—ones that may change psychologists' thinking about the range of possibilities available for converting representations into actions.

In what follows I briefly survey some of the issues that arise when one considers the problem of controlling the way that processes unfold in response to representations, rules, and the contingencies of the environment. My purpose is not to describe the range of control structures that are currently being studied in computer science, but merely to provide some intuitive sense of what some of the distinctions are in this field and to suggest that cognitive science has much to learn from this area of development. Considerations such as these are not likely to be raised without a commitment to the realization of the process model on a computer. And because control issues are one of the central areas of study in computer science, progress in developing computational models of cognitive processes will very likely depend on technical ideas originating in computer science (and more particularly in artificial intelligence).

One of the earliest breakthroughs in understanding the nature of control was the articulation of the idea of feedback from the environment to be controlled. With this a certain balance was restored between a device and its environment: Although only the device is credited with having a goal, the responsibility for its behavior is shared. At times when the environment is passive, the initiative appears to come primarily from the device, whereas at other times the environment appears to intervene, and the initiative seems to go in the opposite direction. This notion of the responsibility for initiation of different actions is fundamental to the understanding of control. In the case of most computer programs, the most common idea has been that of control moving from point to point, or from instruction to instruction, in a largely predetermined way. Such sequencing of instructions makes the notion of *flow* of control quite natural, and branch instructions make it equally natural to think of *passing* or *sending* control to another locus. When control passing is combined with a primitive message-passing facility (for passing arguments), *subroutines* become possible. And because subroutines can be nested—that is, subroutines can themselves send control to still lower subroutines with the assurance that control will eventually find its way back—the notion of a *hierarchy* of control also emerges. Miller, Galanter, and Pribram (1960) saw the psychological importance of the idea of hierarchical subroutines; they called them test-operate-test-exit, or TOTE, units and suggested that they should be viewed as the basic theoretical unit of psychology, replacing the ubiquitous reflex arc. This idea has been very influential in shaping psychologists' thinking about cognition.

There are a number of good reasons why a hierarchical system of control is such a powerful concept. By keeping the interactions between routine and subroutine simple (in terms of both when control is passed and what messages are sent along with it), it becomes easier to think of each routine as a nearly independent subsystem; and that makes the whole system easier to add to, modify, and understand (see the classical discussion of the importance of hierarchical organization in nature in Simon 1969). Each routine in the hierarchy can be thought of as defining some (sub)goal in an overall *goal-directed* system. Passing control to a subroutine amounts to activating a subgoal, and control is returned when that subgoal is consummated. So powerful an idea is this that its shortcomings were largely overlooked for many years.

As early as 1962, however, Allen Newell (Newell 1962) pointed out some of the rigidity in such an organization. So long as each subroutine is a narrow "specialist," such as a routine for searching a list, the usual highly restricted communication between routine and subroutine works well; you can just pass the arguments and a return address to that routine and give it control. It will then return with an answer when it is finished. But if the subroutine is not such a narrow specialist, it might help to be able to communicate each task in more flexible terms. Furthermore it might help if the subroutine's progress could be monitored along the way to prevent it from using up an unwarranted amount of time and resources (for example, memory) on some relatively minor task or on a task that some other process might be able to determine was doomed to fail. Likewise it would help if the subroutine could report its results more flexibly; especially if it could report what went wrong in cases where it failed. How to convert these desiderata into efficient computational form has been one of the main design concerns in developing artificial intelligence programming languages.

A variety of different control structures can be characterized in terms of two distinctions: (1) between *sending* control (where the initiative lies with the *old* locus) and *capturing* control (where the initiative lies with the *new* locus), and (2) between *directing* a message to one specified recipient and *broadcasting* it to all routines or "modules" at once. For example, in the standard subroutine-hierarchy case, control is always *sent* (by the routine that already has it), and a message (containing parameters and a return address) is *directed* specifically to the routine that is being given control; and when the subgoal is achieved, control is *sent* back, along with a result message. In pattern-invoked procedure calls, such as used in Planner or Prolog, when a task needs to be done, a message describing the goal is broadcast, and control is then captured by some module designed to respond to that particular goal message. This is also the basic idea of what is sometimes called a *blackboard* control structure, of which the old Pandemonium system (see, for example, Lindsay and Norman 1977) and the newer Hearsay-II speech recognition system (Erman, Hayes-Roth, Lesser, and Reddy 1980) are examples.

Production systems are special cases of pattern invoked procedure calls. In production systems messages are also broadcast, and control is captured. But when the production finishes, it again just broadcasts a message. Its basic control cycle is called a *recognize-act* cycle, in contrast with the more traditional *fetch-executive* cycle of conventional computing. The current work on production systems is described in chapter 3.

The distinction between whether processes are invoked explicitly by commands, indirectly by the occurrence of other events, or implicitly by certain conditions being met is important in computer science. The distinction is closely related to the difference between a test and an "interrupt" (the latter of which can occur in arbitrary relation to a process). The distinction between data-invoked, or event-invoked, processes (characteristic of so-called demon procedures, which include "if-added" or "if-altered" or "if-examined" procedures that are evoked by various distinct types of computational events), and explicit process-invoked procedures (characteristic of what are sometimes called servant or if-needed procedures) is an important recurring theme in the study of control regimes.

Many of the concerns in designing new architectures reduce to the following three questions: (1) how to enable flexible and effective communication among different processes or modules, (2) how to ensure that all relevant information (and as little as possible irrelevant information) is brought to bear in making decisions or inferences, and (3) how to withhold and release the making of decisions until the appropriate times. The second question has received a great deal of research and has led to some proposals for organizing knowledge so that its relevance to particular topics is easily determined (for example, "Frames," "Scripts," "schemas"). The third question is also of special concern to psychologists (see, for example, Bransford and Johnson 1973) who have demonstrated experimentally that many inferences are carried out in advance of being required for some particular task (for example, at the time utterances are heard, as opposed to at the time their content is needed for some decision). Making decisions or executing procedures must sometimes be withheld until the appropriate context is available. Several proposals for dealing with such linguistic problems as referential opacity rely on this notion of withholding execution pending the appropriate context. For instance, Davies and Isard's (1972) discussion of language comprehension places considerable emphasis on the importance of withholding the evaluation of procedures that attempt to identify the referents of various parts of an utterance until the appropriate time. Thus there is a growing recognition among investigators interested in the problems of cognitive psychology that a variety of questions related to control must play a more prominent role.

For psychologists it has primarily been the attempt to provide a running system that has forced such issues to the fore. Without the need to think in terms of a running system, people have typically focused on what are sometimes called "permissive" rules—such as the rules of logic or of grammar—which specify the relations among representations that are permissible. In that case there is no need to be concerned with the conditions under which particular rules are invoked or with the implications of such control issues for the cognitive architecture.

There is no denying that the system of permissive rules is important. Without a distinction between what Chomsky calls a competence theory and a performance theory, or what McCarthy and Hayes (1969) refer to as the epistemological and the heuristic problems of intelligence, we can find ourselves simply mimicking the most frequent behavior rather than inferring the underlying mechanisms. Yet according to the computational view, understanding a process also requires having a theory of what makes the process unfold as it does on particular occasions, and that in turn requires that issues of control and of the appropriate cognitive architecture be addressed as well.

The Empirical Status of Computational Models: Levels of Correspondence and Strong Equivalence

Regardless of whether one takes the high road or the low road, in cognitive science one is ultimately interested in whether the computational model is empirically valid—whether it corresponds to human cognitive processes. "Corresponding," however, is not a straightforward notion; correspondence can occur at many levels. If a computational process is to be a serious candidate as an explanatory model of mental processing, one is owed some account, as explicit as possible, of how this model relates to the domain of phenomena it is to explain. Specifying the empirical claims entailed by the model is the task of the theory that the model instantiates. Such a theory might, for example, simply claim that the model realizes the same input/output function as the organism being modeled, that it is perhaps a theory of that function, or a type I theory in Marr's terms. As we have seen in discussing the high-road methodology in the previous section, even at this most general level of correspondence, such a theory can make a substantial contribution to understanding the process by providing a theory of the demands of the task.

A stronger claim might be that the model realizes some particular function using the same *method* as the person being modeled. The notion of a method is not very well defined nor consistently used, even among computer scientists. However, it generally entails something more specific than just input/output equivalence. For example, we talk of the relaxation method for solving equations of interacting constraints, Newton's method for locating minima and maxima of a function, the Fourier transform method of computing the effect of a certain filter on a waveform, and so on. These provide a more specific indication of the nature of the process than we would get if we knew only the input/output function.

To specify in greater detail what sequence of steps the system went through would be to provide something like an *algorithm* for the process. The notion of algorithm is somewhat better established in computer science than is the notion of method. For example, there are a variety of well-known algorithms for various kinds of numerical approximations to functions (which are in fact cataloged and published), for parsing context-free languages (for example, the Early algorithm), and so on. Algorithms for sorting and merging lists are another major area of study (see, for example, Knuth 1968), as are algorithms for table lookup.

There are of course even finer levels of comparison between computational processes. For example, an even more specific level of comparison for computational models is that of a *program*: the encoding of a particular algorithm in some programming language. Even finer levels of comparison between computational systems are possible when implemented on actual computers. For example, we could have identical programs that were run on physically different computers, and so on. Thus there is plenty of scope in the possible claims that a theory might make about the level of correspondence between model and empirical domain, or about what properties of the model could be said to have "psychological reality." Clearly, if the computational system is to be viewed as a model of the cognitive *process*, rather than as a simulation of cognitive behavior, it must correspond to the mental process in more detail than is implied by weak equivalence. On the other hand it is equally clear that because computers are not only made of quite different stuff from brains, but the details of how they realize particular operations (say by certain register transfer paths, and by using binary mechanisms and bit-shifting operations) are different from the ways brains work. The correspondence between computational models and cognitive processes seems to fall somewhere in between these extremes.

The general assumption in cognitive science has been that the appropriate level of comparison corresponds roughly to the intuitive notion of algorithm. From our point of view we can think of two computational systems as strongly equivalent, or as being different realizations of the same algorithm or the same cognitive process, if they can be represented by the same program in some theoretically specified computer. A simple way to put this is to say that we individuate cognitive processes in terms of their expression in the canonical language of this theoretical machine.

The functional (as opposed to anatomical) structure of the machine—or what we call its *functional architecture* or just simply its *architecture*—represents our theoretical definition of the right level of specificity (or level of aggregation) at which to view cognitive processes. It is the level at which data structures (or states) of the model are semantically interpreted, with the semantic domain being the cognitive one (that is, in which the states represent the things that are the objects of thought and reasoning, what subjects are thinking *about*). A description of the functional architecture also sets out the functional properties of the cognitive system that are determined by its structure (rather than by the instantaneous contents of its memory), properties such as the functional resources that the brain makes available (for example, which operations are primitive, how memory is organized and accessed, what sequences are allowed, what limitations exist on the passing of arguments and on the capacities of various buffers, and so on). Specifying the functional architecture of a system is very much like providing a manual defining some particular programming language.

Thus one way to address the issue of the appropriate level of comparison between a model and a cognitive process—or the notion of strong equivalence of processes—is by providing a specification of the functional architecture of a "cognitive computer," which I will henceforth refer to as the *cognitive architecture*. Much of the work described in Pylyshyn 1984 is concerned with developing this idea—with providing constraints on an adequate cognitive architecture and showing its relevance to the computational view of mind. For the present purpose the notion of functional architecture will be introduced by discussing the role that this idea plays in computer science and by introducing the closely related, though in many ways unique, role it is required to play in cognitive science. The following discussion illustrates the point that *any* notion of correspondence stronger that weak equivalence *must* presuppose some underlying functional architecture. Thus the question is not whether we need to worry about the cognitive architecture in developing computational models. Rather the issue is whether we can be content to leave it as an implicit assumption—largely conditioned by the functional architecture of currently available computers—or whether we ought to make it explicit and endeavor to bring empirical criteria to bear in constraining it.

Algorithms and Cognitive Architecture

Cognitive algorithms, the central concept in computational psychology, are understood to be executed by the cognitive architecture. According to the strong realism view that many of us have advocated, a valid cognitive model must execute the *same algorithm* as that carried out by subjects. But now it turns out that *which* algorithms can be carried out in a direct way depends on the functional architecture of the device. Devices with different functional architectures cannot in general *directly* execute the same algorithms.[6] But typical commercially available computers are likely to have a functional architecture that differs significantly in detail from that of brains.[7] Hence we would expect that in constructing a computer model of the mental architecture will first have to be *emulated* (that is, itself modeled) before the mental algorithm can be implemented.

For an algorithm to serve as a model of a cognitive process, it must be presented in some standard or canonical form or notation, for example, as a program in some programming language. What is typically overlooked when we do this is the extent to which the class of algorithms that can be considered is conditioned by the assumptions we make regarding what basic operations are possible, how these may interact, how operations are sequenced, what data structures are possible, and so on. Such assumptions are an intrinsic part of our choice of descriptive formalism because descriptive formalism defines what I have been calling the functional architecture of the system.

Yet the range of programming languages or functional architectures that are conveniently available is actually quite narrow. Most available architectures are register-based, in which symbols are stored and retrieved by their numerical or symbolic "addresses," control is transferred sequentially through a program (except for "branching" instructions), and operations on symbols are accomplished by retrieving them from memory, placing them in a designated register, applying one of the primitive commands to them, and then storing the resulting symbol back into memory. Although there are variants of this basic pattern, the main idea of a sequential process proceeding through a series of "fetch," "operate," and "store" operations has been dominant since the beginning of digital computation (see the previous section for a discussion of alternatives being examined in computer science research). This goes for both hardware and software (see a discussion of the latter in Backus 1978).

Because our experience has been with such a narrow range of architectures, we tend to associate the notion of computation, and hence of algorithm, with the particular class of algorithms that can be executed by architectures in this limited class. However, this is misleading because, as I noted, different architectures permit different algorithms to be executed.

This point is best illustrated by considering examples of several simple architectures. Perhaps the most primitive machine architecture is the original binary-coded Turing machine. Although this machine is universal, in the sense that it can be programmed to compute any computable function, anyone who has tried to write procedures for it will have noticed that most computations are extremely complex. More important, however, the complexity of the sequence of operations it must go through varies with such things as the task and the nature of the input in ways that are quite different from that of machines with a more conventional architecture. For example, the number of basic steps required to look up a string of symbols in such a Turing machine increases as the square of the number of strings stored.

In contrast to this architecture, in what is called a register architecture (an architecture that has what is usually referred to as random access memory, in which retrieving a symbol by name or by "reference" is a primitive operation), the time complexity for looking up a symbol in a table can, under certain conditions, be made independent of the number of strings stored. Because of this a register architecture can directly execute certain algorithms (for example, the hash-coding lookup algorithm) which are impossible in the Turing machine—in spite of the fact that the Turing machine is universal and therefore can compute the *same function* as the algorithm or be programmed to be "weakly equivalent" to the algorithm. In other words it can compute the same lookup function, but not with the same complexity profile and hence not by using the same hash-coding algorithm.

Now of course a Turing machine could be made to mimic the sequence of states that the register machine goes through by first arranging for the Turing machine to compute the functions realized by *each individual operation* of the register machine or in other words to simulate each individual step that the register machine takes in executing its algorithm. But in that case the Turing machine would first be *emulating* the architecture of the register machine and then executing the algorithm in the emulated architecture, a very different matter from computing it directly by the Turing machine.

The distinction between directly executing an algorithm and executing it by first emulating some other functional architecture is crucial to cognitive science. It bears on the central question of which aspects of the computation can be taken literally as part of the model and which aspects are to be considered as mere implementation details (like the color and materials out of which a physical model of the double helix of DNA is built). We naturally expect that we shall have to have ways of implementing primitive cognitive operations in computers, and that the details of how this is done may have little empirical content.

From the point of view of cognitive science, it is important to be explicit about *why* a model works the way it does and to independently justify the crucial assumptions about the cognitive architecture. That is, it is important for the use of computational models as part of an explanation, rather than merely to mimic some performance, that we not take certain architectural features for granted simply because they happen to be available in our computer language. We must first explicitly acknowledge that certain noncomputational properties originate with certain assumed properties of the cognitive architecture, and then we must attempt to empirically motivate and justify such assumptions. Otherwise important features of our model may be left resting on adventitious and unmotivated assumptions.

This issue frequently arises in connection with claims that certain ways of doing intellectual tasks, for example, by the use of mental imagery, bypasses the need for explicit representation of certain logical or even physical properties of the represented domain and bypasses the need for inefficient combinatorially explosive processes like logical inference. The issue is frequently stated in terms of hypotheses that one or another mental function is carried out by an "analog" process. The whole issue of analog versus digital processing is complex and has not in general been well understood (see the discussion in chapter 7 of Pylyshyn 1984). For the present I consider only simple cases involving the claim that some process was carried out by "direct readout" or otherwise noncognitive or noninferential means. From the present perspective this would be interpreted as the claim that some cognitive function was actually part of the cognitive architecture.

Consider cases such as the following: People have occasionally suggested that subjects do not need to have knowledge of relational properties such as, say, transitivity, in making certain inferences, such as in three-term series problems (for example, "John is taller than Mary and John is shorter than Fred. Who is tallest?"). According to this view, all subjects have to do is arrange the three items in order (either in a list or in an image) and read the answer off—they simply *notice* which object is first (or last) in the list. But of course even if a subject can solve the problem in this way, that does not mean that tacit knowledge[8] of formal properties (for example, transitivity) of the relation "taller than" is not needed.

There are at least two reasons why one might have to postulate knowledge of formal relations. First, the decision to represent "taller" by something like "further on the list" must have been based on the tacit recognition that the two relations were of the same formal type (a list would not, for example, have been suitable for representing the relation "is married to"). Second, although ordering three names in a list and then examining the list for the position of a particular name may seem straightforward and free from logical deduction, a little thought will show that the ability to carry out this operation mentally, as distinct from physically, presupposes a great deal about the available primitive mental operations. In particular, appealing to the existence of a "mental list" (or some such structure) involves certain assumptions about the properties that such a structure *intrinsically* possesses. For example, if the subject has a mental representation of items A, B, and C and reasons (according to the theory) by placing A and B in a certain order and then adding C next in the sequence, the model must assume that placing C next to B leaves the relation between A and B unchanged and the relation of A to C (with B between them) remains the same with respect to the relevant represented relation (that is, tallness) as that between A and B.

Assumptions such as these are justifiable only if there exists an operation in the cognitive architecture that has the same formal mathematical properties (that is, falls under the same system of logical axioms) as the relations "taller" and "further along the ordering." Furthermore even if such an operation is part of the cognitive architecture, one is still not entitled to assume that the use of this capacity requires no further appeal to tacit knowledge of logical constructs like transitivity, as the first point shows. To take another timely example, matrix data structures have frequently been used to represent the spatial properties of images (see, for example, Kosslyn and Shwartz 1977, Funt 1980). This is a convenient way to represent spatial layout, partly because we tend to think of matrices in spatial terms anyway. In addition, however, this structure seems to make certain consequences available without any apparent need for certain deductive steps involving reference to knowledge of geometry. For example, when we represent the locations of imagined places in our model by filling in cells of a matrix, we can "read off" such facts as which places are adjacent, which places are left of, right of, above, or below a given place, and which places are in between a given pair of places. Furthermore when a particular object is moved to a new place, its spatial relations to other places need not be recomputed. In an important sense this is implicit in the data structure. Such properties make the matrix a much more natural representation than, say, a list of assertions specifying the shape of objects and their locations relative to other objects.

But, as in the example of solving the three-term series problem without apparently using logical inference rules (by using a list), such properties of matrices arise from the existence of certain formal properties that are part of the architecture of virtually all contemporary computers. Such properties are not, however, constitutive of computing. They would not, for instance, be available in a Turing machine architecture. For a matrix data structure with the desired properties to be realizable, the architecture must provide at least the primitive capacity to address the content of a representation by place—that is, it must be possible to name a location and to ask for the content of a named location. This itself requires what is called a register architecture (or some other kind of location-addressable store).

Furthermore in this architecture it must also be possible to primitively generate the names of places adjacent to a given place (that is, it must be possible to do this without appealing to other representations or to tacit knowledge of geometry or anything else that would involve intermediate inferential steps). This is necessary if we want "scanning" of the representation to be a (nondeductive) primitive operation. In addition there must be primitive predicates that, when applied to names, evaluate the relative directions of places corresponding to those names (for example, two-place predicates such as *right-of* must be primitive in the architecture). This in turn implies that there are at least two independent total orderings over the set of names. In addition if the relative distance between places is to be significant in this representation, then there might have to be further primitive operations that can be applied to place names so as to evaluate, say, relative size (for example, the predicate *larger-than*).

This whole array of formal properties is available in all common computer architectures because they all use numerical expressions for register (that is, place) names and have built-in primitive arithmetic operations. But these properties are part of such architectures for reasons that have nothing to do with the theoretical needs of cognitive science. When these features are exploited in building cognitive models, we tacitly assume that such operations are part of the cognitive architecture of the mind—an assumption that clearly needs to be independently motivated and justified. Arguments have rarely been provided for any such proposals. Among the few suggestions for such abstract architectural features that I have seen are due to Brouwer (1964) and Nicod (1970), who for quite different reasons proposed that *succession* be viewed as a cognitive primitive, and G. Spencer Brown (1969), who proposed that drawing a (binary) distinction (a sort of universal figure-ground conceptual separation) is a primitive operation of the mind. Of course a great deal of the recent cognitive science research program—at least since Newell's seminal paper on production systems (Newell 1973b)—has been concerned with proposing specific features of the cognitive architecture (see, for example, chapter 3).

In choosing a particular architecture, one makes a commitment concerning which functions are the free parameters that can be tailored to fit specific situations, and which are fixed over a certain range of influences or are primitive subfunctions shared by all processes in a certain class. Restrictions on the availability of certain primitive computational functions is a virtue if our goal is to provide an explanation. The more constrained a notation or architecture, the greater the explanatory power of resulting models.

This is exactly the problem of reducing the degrees of freedom available for fitting a model to observations. Each function that can be attributed to the functional architecture, rather than to the flexibly alterable program, attains the status of a constant rather than that of a free empirical parameter in the model. It provides a principled rationale for why on some particular occasion the model takes one particular form as opposed to other logically possible ones. It is precisely the lack of such a rationale that makes some computational models ad hoc. One goal in developing explanatory cognitive models then would be to fix as many properties as possible by building them into the fixed cognitive architecture. Opposing this goal, however, is the need to account for the remarkable flexibility of human cognition. This in turn leads us to attribute the behavioral regularities to the way in which the architecture is used—that is, to the programs and knowledge that allow the relatively rigid architecture to be exploited in generating behavior that is highly plastic. The stimulus-independence of cognition provides one of the strongest reasons for attributing much of its manifest behavior to tacit knowledge of various kinds rather than to the sorts of fixed functional properties that have frequently been proposed.

2.3 Methodologies for Assessing Strong Equivalence

How does one distinguish between regularities that are attributable to properties of the cognitive architecture and those attributable to the nature of the cognitive process and its representations? Twenty-five years ago many of the techniques for assessing strong equivalence (for example, mental chronometry) were not available—or rather their use in this context was not understood. If someone had undertaken to analyze the notion of strong equivalence at that time, much of what we now believe is germane would not have been included. Although we can expect such techniques to continue to evolve, it may be useful to list a few provisional methods that are pervasive in the work of information-processing psychologists.

Intermediate State Evidence

As an example, recall that strong equivalence requires that a model be expressed at a level of aggregation in which all the basic representation states are revealed, because each such state is essential in the representational story. Thus the transitions between representational states must themselves not involve any representational states; these transitions must be realized directly by the cognitive architecture. Hence any evidence for the existence of such intermediate representational states is evidence for the nonprimitiveness of the subprocess in question. There are various methods for obtaining such evidence.

One of the earliest methods for discovering intermediate states in problem solving involves the recording of subjects' expressed thoughts while solving the problem (Duncker 1935). Newell and Simon (1972) developed this technique, which they call *protocol analysis*, to a high level of precision. Although the method can only be used with certain slow and deliberate types of problem-solving tasks (including problems involving visual imagery; compare Baylor 1972, Farley 1974, Moran 1973), it does provide evidence for intermediate states that might otherwise be unavailable for constraining the model. When combined with additional intermediate observations, such as protocols of movements obtained from video recordings (Young 1973) and records of eye movements (Just and Carpenter 1976), this method can yield extremely useful data. The use of this method is discussed in greater length in chapter 1.

The existence of intermediate representational states can sometimes also be inferred in more indirect ways. A good example occurs in psycholinguistics, in the study of real-time sentence processing. There is some indirect evidence for the availability of certain components of syntactic analysis in the course of sentence comprehension (Frazier and Fodor 1978, Marslen-Wilson and Tyler 1980, Forster 1979). Any evidence of the availability of intermediate states of a process to any other process (that is, any evidence that the workings of the process are "transparent" to another part of the system) can be taken as evidence that such a process is not primitive but has a further cognitive decomposition.

In the remainder of this section I consider two other empirically based criteria for deciding whether certain aspects of behavior regularities ought to be attributed to properties of mechanisms—that is, to the cognitive architecture—or to the representations and processes operating on them. Both are, as suggested, sufficient though not necessary conditions—that is, they can ideally tell you when a function requires a more complex cognitive analysis, but cannot tell you that you have gone far enough because there may be various sources of indirect evidence for the need for further decomposition. The *first* of these criteria derives from computational considerations and defines a notion of strong equivalence of processes referred to as *complexity-equivalence*. This criterion is frequently associated with the use of reaction-time measures, or with such on-line measures as those that assess the attention demand of tasks (for example, measuring the performance on a secondary task).

The *second* criterion helps us to decide whether a particular empirical phenomenon ought to be attributed to the architecture or to goals and beliefs. It relies on the assumption that we can identify certain clear cases of phenomenon that should be accounted for at the knowledge level, that is, in terms of the representations alone, rather than in terms of properties of the cognitive architecture. Phenomena that depend in a rational way on subjects' goals, beliefs, and utilities are a case in point. For example, in psychophysics we assume that if a measure (such as a threshold) changes systematically as we change the payoffs (that is, the relative cost of errors of commission and of omission), then the explanation of that change must be given at the knowledge level—in terms of decision theory—rather than in terms of properties of sensors or other mechanisms that are part of the architecture. In general showing that certain empirical phenomena are sensitive to goals and beliefs (or are what I call *cognitively penetrable*) is prima facie evidence that they should not be attributed to properties of the architecture.

Relative Complexity Evidence and Complexity-Equivalence

Recall that in the example discussed previously, there was at least one property of the hash-coding algorithm that needs to be preserved by any strongly equivalent process—and which would not be preserved if the same function were to be realized on a traditional Turing machine. That property is the *relation* between (or the form of the function that characterizes the relation between) the number of steps that it would take to look up a symbol in a table and the total number of symbols stored there. The hash-coding algorithm, implemented on an architecture with a primitive facility to retrieve symbols by name (what is commonly referred to as a random access or register architecture), is able to look up symbols with a number of steps that is (to a first approximation) independent of the number of entries in the table. By contrast if this algorithm were emulated on a Turing machine, the number of steps that it would take would increase as the square of the number of strings stored on the tape (so that the function relating the number of steps and the number of items stored would be a second order polynomial).

The relation between number of primitive steps taken and certain properties of the symbolic input is generally considered to be an essential invariant property of what one intuitively thinks of as different realizations of the same algorithm. For example, one would clearly not count two processes as realizing the same algorithm if one of them computed a function in some fixed time, independent of the size of its input, whereas the other was combinatorially explosive—so that the time it took increased exponentially as some property (for example, length) of the input was varied. The total amount of time or the total number of steps taken is not important for assessing equivalence of algorithms, because that depends on the particular machine on which the algorithm is running. What is important is the nature of the relation between such things as time or number of steps taken and properties of the input, such as its length. Because of this there are certain apparent differences among programs that do not matter for purposes of what I have called their *complexity-equivalence* (Pylyshyn 1984).

In cognitive science the most common way of assessing relative complexity is by measuring relative reaction times, that is, by observing the form of the function relating the time taken for a task and certain parametric properties of the task (for example, size of the input). Although the utility of this measure also rests on methodological assumptions, it has turned out to be one of the most valuable sources of relative complexity evidence in the cognitive science toolbox. Examples of its use are discussed in chapter 7, as well as in many other chapters. In a following section of this chapter, I return to a consideration of the status of measures such as reaction time when I take up the question of whether it is possible in principle to decide on the correct computational models by using behavioral data alone.

The set of programs that are complexity-equivalent clearly represents a refinement of the set of programs that compute the same input/output function, and hence complexity-equivalence represents a restriction of the weak equivalence relation. Although complexity-equivalence captures an important aspect of the intuitive notion of "same algorithm," it is not by itself sufficient to define strong equivalence. It is in other words a necessary but not sufficient condition for strong equivalence.

Cognitive Penetrability

A second class of methods for studying strong equivalence assumes that what I have been calling cognitive phenomena are a natural scientific domain that can be explained entirely in terms of the nature of the representations and the structure of programs running on the cognitive architecture. If that is to be true, then the cognitive architecture itself must not vary in ways that demand the same sort of *cognitive* explanation. It must in other words form a cognitive "fixed point," so that differences in cognitive phenomena might be explained by appeal to the arrangements (sequences of expressions and of basic operations) among the fixed set of operations and to the basic resources provided by this architecture. Though the architecture might vary as a function of physical or biochemical conditions, it should not vary directly and in logically coherent ways with changes in the content of the organisms' goals and beliefs. If the cognitive architecture were to change in ways requiring a cognitive rule-governed explanation, it could itself no longer serve as the basis for explaining how changes in rules and representations produce changes in behavior. Consequently the input/output behavior of the hypothesized primitive operations of the cognitive architecture must not itself depend on goals and beliefs, and hence on conditions that change the organism's goals and beliefs, it must be what I refer to as *cognitively impenetrable.*

This is usually a straightforward criterion to apply in practice. To determine whether certain empirical evidence favors certain hypothesized architectural properties of the cognitive system, the natural question to ask is whether the evidence is compatible with some other, different, architectural properties. One way to do this is to see whether the empirical phenomena in question can be systematically altered by changing subjects' goals or beliefs. If they can, then this suggests that the phenomena tell us not about the architecture but rather about some representation-governed process—something that in other words would remain true even if the architecture were different from that hypothesized.

For example, this appears to be the case with certain kinds of imagery phenomena, such as the linear relation between reaction time and the distance on a mental image that is mentally "scanned" (for more on this case, see Pylyshyn 1981). That is because the linear increase can be made to disappear by changing the instructions, for example, by asking subjects to imagine a situation in which they do not believe there would be any increase in reaction time as a function of distance (that is, in which they believe there would be no relation between time and distance in the *real* situation that they are to imagine).

In general, showing that a certain phenomenon is cognitively penetrable provides strong reason to interpret that phenomenon as arising from the nature of the representations and from cognitive processes operating over these representations. In practice there is always the question of exactly which stage of the process is affected by the instructions, but this is not a problem unique to the penetrability criterion. Being able to determine whether some phenomenon is due to properties of the architecture or of the representation-governed process is critical to assessing strong equivalence because it gives us a way of determining whether we have broken down the processing steps into the right primitive elements.

Can We Decide which Model is Correct from Behavioral Data? The Behavioral Indeterminacy Claim

Finally, I end this section on methodology with a brief discussion of the claim that strong equivalence may in principle not be possible in psychology (or that it can only be achieved by appealing to the facts of biology or by the gratuitous importation of esthetic or other sorts of subjective judgments). Weak equivalence is, by definition, equivalence with respect to input/output behavior. Consequently one might suppose that if all we had in cognitive science was observed behavior, one could not hope to determine which computational model is the correct one beyond choosing one of the weakly equivalent ones. Indeed this point of view has frequently been advocated (compare Anderson 1978, Townsend 1974). Note that this is supposed to be indeterminism beyond the usual scientific indeterminism, wherein a finite body of data never uniquely determines the true theory.

First, it should be pointed out that nobody wishes to exclude psychobiological data as a source of evidence in evaluating theories of cognition (see, for example, the discussions by Sejnowski and Churchland in chapter 8). But whether or not they are included has no bearing on the indeterminism arguments because cognitive models are not models of how the brain realizes processes in neural tissue, they are theories that describe *cognitive* mechanisms that process *cognitive representations*. Neurophysiological evidence can and sometimes does help to decide psychological issues, but contrary to what some people appear to believe, this sort of evidence is just as indirect and fallible as the measurement of reaction times; we can no more directly observe a cognitive mechanism by the methods of biology than we can by the methods of psychophysics (or for that matter by introspection). If that is the case, how can we hope to do better than selecting one of the set of models that are weakly equivalent (other than perhaps by appeal to such external criteria as parsimony and elegance—as some have suggested (Anderson 1978))?

The answer that I have suggested (Pylyshyn 1978b) for this sort of indeterminability claim is this: Although in a sense all we have is behavior, not all behavior is of the same kind from the point of view of theory construction. By distinguishing between different kinds of behavioral measures and by interpreting these measures in different ways that are independently motivated, we can do very much better than weak equivalence.

Notice that placing different interpretations on observed behavior is routine in all experimental psychology. For example, an investigator may collect primary observations in a certain domain—say, concerning the behavior of a person in solving a problem. There are observations that a constructive theory of that domain might be expected to account for by generating similar behavior as its output. But the investigator typically also collects observations of a secondary kind (which might even be called, without too serious a distortion of terminology, metabehavioral observations), from which certain properties of the problem-solving process itself might be inferred. This is the case, for example, when subjects provide "thinking-out-loud" protocols. It is also the case when observations are made that are interpreted as *indexes* of such things as processing complexity or the attention demand of the task. In such a case it is not expected that a theory or a model would actually generate such behavior as part of its *output*. Rather the idea is that the model should generate the primary (output) behavior in a manner that reflects certain real-time processing properties indexed by observations in the secondary class.

Consider the following example in which the developing methodology of cognitive science has led to a gradual shift in the way an important aspect of observed behavior is interpreted. The example concerns what is probably the most widely used dependent measure in cognitive psychology, namely, reaction time. This measure has sometimes been interpreted as just another response, to be accounted for by a cognitive model in the same way that the model accounts for such response records as the sequence of the buttons that were pressed. Since Donders's pioneering work (carried out in the 1860s and reprinted as Donders 1969) it has also been widely interpreted as a more-or-less direct measure of the duration of mental processes (Wasserman and Kong 1979). I have argued (Pylyshyn 1979a, 1984) that neither of these interpretations is correct in general; reaction time can be viewed in general as neither the computed output of a cognitive process itself nor a measure of the duration of a mental event or mental operation.[9]

If reaction time were thought of as simply another response, then it would be sufficient if our computational model simply calculated a predicted value for this reaction time, given the appropriate input. But that would not be sufficient if the computation is to be viewed as a literal model of the cognitive *process*. Contemporary cognitive scientists would not view a system that generated pairs of output strings, interpreted as the response and a number designating the time taken, as being an adequate model of the underlying process, no matter how well these outputs fit the observed data. That is because they wish to interpret the model as computing the output *in the same way* as the subject (that is, by using the same algorithm).

It has become customary in cognitive science to view reaction time the same way that measures such as galvanic skin response or plethysmograph records, or measures of distractibility (see, for example, Brown 1962) are viewed, namely as an *index*, or an observable correlate, of some aggregate property of the process. In particular reaction time is frequently viewed as an index of what I call *computational complexity*, which is usually taken to correspond to such properties of the model as the number of operations carried out. A process that merely computed time as a parameter value would not account for reaction time viewed in this particular way because the parameter would not express the computational complexity of the process.

I have discussed several cases in which it is possible to decide which of two different algorithms is being used by examining the relative number of primitive steps that they took when given different inputs. Now if there is some reason to believe that the amount of (real) time it takes is proportional to (or at least a monotonically increasing function of) the number of such primitive steps of the algorithm, then measures of relative time taken might provide the evidence needed to decide between the putative algorithms. But in this case we need to have independent reason to believe that reaction time is a valid index of the number of primitive steps of the cognitive architecture. Such independent reasons are frequently available, as, for example, when regularities inferred on the basis of the assumption that reaction time is a reliable indicator of processing complexity are corroborated by other methods. When such patterns of consistency keep showing up under converging methodologies, we then have a prima facie reason to expect such methods to be valid, other things being equal (compare Posner 1978).

Nonetheless it should be kept in mind that when we draw inferences about the nature of the algorithm from reaction-time data (or any other *physical* measurement), we always depend on the validity of ancillary hypotheses. Such hypotheses could in principle be false. There are many situations in which measurements of properties of the underlying physical events may tell us little about the algorithm. They might instead tell us either about the way in which the process is physically (that is, neurophysiologically) instantiated on some particular occasion or in one particular individual, or they might tell us about subjects' tacit knowledge or about the nature of the task itself. For example, I have argued (Pylyshyn 1981, 1984) that many of the phenomena of mental imagery research (for example, the so-called mental scanning results of Kosslyn 1980) are of just this sort. In these cases it appears that the particular reaction-time patterns observed are the direct result not of properties of the architecture but of subjects' tacit knowledge of what would happen in the imagined situations and their ability to duplicate aspects of these situations (for example, their duration) imaginally. The argument for this is based on the cognitive penetrability criterion: if the pattern of behavior can be altered in a rational way by changing subjects' beliefs about the task, then we have prima facie evidence that the process involves inference. Moreover we have little reason to hypothesize special-purpose architectural properties if we can account for the pattern of reaction times merely in terms of subjects' beliefs together with their psychophysical ability to generate the relevant time intervals. This of course does not apply in cases where it is clear that subjects' performance is not explicable in terms of their beliefs—as, for example, when they are operating at the limits of their ability, as is the case with most studies carried out in the information-processing tradition, including some of the imagery manipulation experiments, such as those involving "mental rotation" (as in Shepard and Cooper 1982).

There are other sorts of cases where observed reaction times do not tell us much about the nature of the cognitive architecture. For example, Ullman (1984) has suggested that the reason certain kinds of visual processes (which he calls *visual routines*) are carried out serially is not because of the nature of the cognitive architecture but because the nature of the task itself requires it. In that case the fact that the process is serial cannot be attributed to requirements imposed by the architecture (at least not entirely), though it does show that the architecture is capable of serial operation.

2.4 Conclusion: The Domain of Cognitive Science

What makes some area of study a natural scientific domain is the *discovery* (not the stipulation) that some relatively uniform set of principles can account for phenomena in that domain. It is never the case that we can stipulate in advance precisely what will fall into that natural domain. Nor can we stipulate in advance what the class of principles is that will define the domain; the evolution of the boundaries of a scientific domain is a gradual process, requiring provisional conjectures as one proceeds.

Cognitive science has been viewed as the study of the natural domain of cognition, where cognition includes prototypical phenomena of perception, problem solving, reasoning, learning, memory, and so on. At present the working hypothesis appears to be that what these have in common is that they involve intelligent activity in some general sense.

A bolder hypothesis is that cognition is the domain of phenomena that can be viewed as natural information processing, which in current terms means that it is computational, that being the only notion of autonomous mechanistic information processing we have. This in turn means that phenomena in this domain can be explained on at least three distinct levels, as suggested by what I call the classical view. According to this hypothesis (which we might call the *computational realist* view), we cannot stipulate in advance which empirical phenomena will turn out to be cognitive in the technical sense (meaning susceptible to a computational explanation). It would be both surprising and troublesome if too many of what we pretheoretically took to be clear cases of cognition ended up being omitted in the process. But it would also not be entirely surprising if some of our favorite candidate cognitive phenomena got left out. For example, it could turn out that consciousness is not something that can be given a computational account. Similarly certain kinds of statistical learning, aspects of ontogenetic development, the effect of moods and emotions, and many other important and interesting phenomena could simply end up not being amenable to a computational account. Substantial components of such phenomena could, for example, require a noncomputational explanation, say, in terms of biochemistry or some other science.

In that regard it *could* turn out that certain phenomena might not arise from symbol processing, contrary to earlier assumptions. In that case connectionists' claims that symbol systems are not needed (see, for example, Rumelhart, McClelland, et al. 1986) could turn out to be right for those phenomena. On the other hand there are very good reasons for maintaining that *reasoning* and other *knowledge-dependent* or rational process require symbol processing, and moreover that these processes are extremely pervasive in the phenomena that have been studied in cognitive science. Only time and further research will tell which phenomena might be better explained by models that do not conform to the classical notions of computing.

Notes

I thank Allen Newell for his extensive comments on a draft of the manuscript.

1. Although the cognitive science community tends to use the term *knowledge* quite freely in discussing semantic·level principles, it is sometimes worth distinguishing those semantic entities that are knowledge, from those that are goals, percepts, plans, and so on. The more general term *semantic level* is used in contexts where such distinctions are important. Philosophers even talk about the "intentional" level or "intentional" objects, but because the use of that terminology tends to raise a large, ancient, and not entirely relevant set of issues, we shun that term here.

2. Note that although no bounds are assumed on the length of the expressions, the arguments presented hold even if the expressions are bounded—so long as the length is sufficiently great that the expressions have to be treated as though they were not atomic, which is true in any useful computer. Because of this *IF* instantiates the expressions *by means of* instantiations of the elementary symbols o and x and the concatenation relation. Similarly operations on expressions (such as the operation #) are defined in terms of operations on individual symbols.

3. I use the term *architecture* here somewhat loosely because it has not yet been defined. In a following section this notion is discussed in greater detail because it is one of the central computational ideas in cognitive science.

4. Of course there are also those studying cognition who deny any concern with modeling human cognitive processes; they wish only to create computer systems that carry out some intelligent task. Yet there is reason to think that such people are also implicitly developing theories of human cognition, inasmuch as facts about human cognition are being brought in as part of the task definition (see the discussion of implicit empirical constraints in artificial intelligence research in Pylyshyn 1978).

5. Although in discussing the distinction between a computational theory and an algorithm Marr draws the analogy between mathematical theories, such as the theory of Fourier analysis, and particular algorithms, such as the fast Fourier transform (FFT) algorithm, the examples from his own work in vision do not appear to fit that analogy. In fact what is called a "theory of the computation" (or a type I theory) is typically a theory that links a function (such as computing structure from motion or shading) to a teleological story. Marr was preoccupied with the question What is this computation *for*? or What useful information about the world does it provide the organism? This, however, does not provide the basis for a principled distinction between levels. It is clearly just a useful heuristic for encouraging the theorist to look for independent motivations and broader functional units when formulating a theory in some domain.

6. We can take this claim as a point of definition for present purposes, although there are some technical issues here that would have to be addressed in a more detailed discussion. The criterion for being the same algorithm is closely linked to the idea of *direct* execution. For example, we can trivially change an algorithm (say, by adding a fixed number of redundant operations such as "no ops" to each original operation), yet for our purposes we may not want to count this variant as a distinct algorithm—that is, we may want to count any machine executing the variant as carrying out the same process as the original machine executing the original algorithm. To develop this idea, we may need concepts such as that of a canonical description of a process (for example, along the lines that I have sketched in Pylyshyn 1984).

7. Although, as I have already suggested, the working hypothesis of most of cognitive science is that whatever the functional architecture turns out to look like in detail, it will continue to fall into the class of symbol-processing, or classical, architectures as originally envisaged by Turing and as is true of all systems we call digital computers today. The reason is that to have sufficient representational power as well as the right kind of semantically coherent behavioral plasticity, the computational system must read, write, and transform structured symbolic expressions that have combinatorial semantic properties (these points are argued at some length in Fodor and Pylyshyn 1988).

8. The term *tacit knowledge* is used here in the usual way to refer to real knowledge that subjects have even though they are not aware of having and using it—an unproblematic idea in contemporary cognitive science, where it is taken for granted that subjects need not have awareness or "metaaccess" to most cognitive structures and processes. The term has nothing to do with Polanyi's (1964) use of the same phrase.

9. The reason that reaction times cannot be viewed as measuring the duration of a mental operation is simply that when one speaks of a certain mental operation, such as comparing a stimulus with an item in memory, one is referring not to any one particular occurrence of this operation in the brain but in general to the class of all occurrences—all event *tokens*—that would constitute the same event-*type*—for example, the event-type "compare stimulus *S* with item *X* in memory." Thus in referring to the CAR function in Lisp, one is not referring to any particular occasion on which that function is evaluated. Because we distinguish between the operation as a *computational* event and the particular *physical* events that carry it out on particular occasions, all occurrences of a particular operator

need not (by definition) have a unique duration associated with them, just as they need not have a unique size or location in the brain (or a computer) associated with them. Because of this, one does not in general speak of the duration of a CAR operation, although of course each particular event-token that constitutes the execution of a CAR operation does have a unique duration on each occasion. For some reason the idea of distinguishing token events from event types is alien to many psychologists, so this point is frequently lost (see, for example, Wasserman and Kong 1979).

References

Anderson, J. R. 1978. Arguments concerning representations for mental imagery. *Psychological Review* 85:249–277.

Backus, J. 1978. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. *Communications of the Association for Computing Machinery* 21:613–641.

Baylor, G. W. 1972. A treatise on the mind's eye: An empirical investigation of visual mental imagery. Ph.D. diss., Carnegie Mellon University, Pittsburgh, PA. Ann Arbor, MI: University Microfilms, no. 72-12,699.

Bransford, J. D., and Johnson, M. K. 1973. Considerations of some problems of comprehension. In W. Chase, ed. *Visual Information Processing*. New York: Academic Press.

Brouwer, L. E. J. 1964. Intuitionism and formalism. In P. Benacerraf and H. Putnam, eds. *Philosophy of Mathematics*. Englewood Cliffs, NJ: Prentice-Hall.

Brown, G. S. 1969. *Laws of Form*. London: Allen Unwin.

Brown, I. D. 1962. Measuring the "spare mental capacity" of car drivers by a subsidiary auditory task. *Ergonomics* 5:247–250.

Davies, D. J. M., and Isard, S. D. 1972. Utterances as programs. In B. Meltzer and D. Michie, eds. *Machine Intelligence 7*. Edinburgh: Edinburgh University Press.

Davis, M., ed. 1965. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Hewlett, NY: Raven Press.

Donders, F. C. 1969. On the speed of mental processes (1868–1869). *Acta Psychologica* 30:412–431.

Duncker, K. 1935. On problem solving. *Psychological Monographs* 58(270):5.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. 1980. The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys* 12:213–253.

Farley, A. 1974. A visual imagery and perception system. Ph.D. diss., Carnegie Mellon University, Pittsburgh, PA.

Fodor, J. A. 1975. *The Language of Thought*. New York: Crowell.

Fodor, J. A., and Pylyshyn, Z. W. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition* 28:3–71.

Forster, K. I. 1979. Levels of processing and the structure of the language processor. In W. E. Cooper and E. C. T. Walker, eds. *Sentence Processing: Psycholinguistic Studies Presented to Merrill Garrett*. Hillsdale, NJ: Erlbaum.

Frazier, L., and Fodor, J. D. 1978. The sausage machine: A new two-stage parsing model. *Cognition* 6:291–325.

Funt, B. V. 1980. Problem-solving with diagrammatic representations. *Artificial Intelligence* 13(3):201–230.

Guthrie, E. R. 1935. *The Psychology of Learning*. New York: Harper.

Johansson, G. 1975. Visual motion perception. *Scientific American* 232:76–88.

Just, M. A., and Carpenter, P. A. 1976. Eye fixations and cognitive processes. *Cognitive Psychology* 8:441–480.

Knuth, D. E. 1968. *Fundamental Algorithms. The Art of Computer Programming*. Vol. 1. Reading, MA: Addison-Wesley.

Kohler, W. 1947. *Gestalt Psychology, An Introduction to New Concepts in Modern Psychology*. New York: Liveright.

Kosslyn, S. M. 1980. *Image and Mind*. Cambridge, MA: Harvard University Press.

Kosslyn, S. M., and Shwartz, S. P. 1977. A data-driven simulation of visual imagery. *Cognitive Science* 1:265–296.

Lindsay, P. H., and Norman, D. A. 1977. *Human Information Processing: An Introduction to Psychology*. 2d ed. New York: Academic Press.

Marr, D., and Poggio, T. 1979. A computational theory of human stereo vision. *Proceedings of Royal Society, London B* 204:301–328.

Marslen-Wilson, W., and Tyler, L. K. 1980. The temporal structure of spoken language understanding. *Cognition* 8(1):1–71.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, eds. *Machine Intelligence* 4. Edinburgh: Edinburgh University Press.

Miller, G. A., Galanter, E., and Pribram, K. H. 1960. *Plans and the Structure of Behavior*. New York: Holt, Rinehart and Winston.

Moran, T. 1973. The symbolic imagery hypothesis: A production system model. Ph.D. diss., Carnegie Mellon University, Pittsburgh, PA.

Newell, A. 1962. Some problems of basic organization in problem-solving programs. In A. Yovitts, G. T. Jacobi, and G. D. Goldstein, eds. *Self-Organizing Systems*. New York: Spartan.

Newell, A. 1970. Remarks on the relationship between artificial intelligence and cognitive psychology. In R. Banerji and M. D. Mesarovio, eds. *Theoretical Approaches to Non-Numerical Problem Solving*. New York: Springer-Verlag.

Newell, A. 1973a. Artificial intelligence and the concept of mind. In R. C. Schank and K. Colby, eds. *Computer Models of Thought and Language*. San Francisco: W. H. Freeman.

Newell, A. 1973b. Production systems: Models of control structures. In W. Chase, ed. *Visual Information Processing*. New York: Academic Press.

Newell, A. 1973c. You can't play twenty questions with nature and win. In W. Chase, ed. *Visual Information Processing*. New York: Academic Press.

Newell, A. 1980. Physical symbol systems. *Cognitive Science* 4(2):135–183.

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

Nicod, J. 1970. *Geometry and Induction*. Berkeley: University of California Press.

Polanyi, M. 1964. *Personal Knowledge*. London: Routledge and Kegan Paul.

Posner, M. I. 1978. *Chronometric Explorations of Mind*. Hillsdale, NJ: Erlbaum.

Pylyshyn, Z. W. 1978. Computational models and empirical constraints. *Behavioral and Brain Sciences* 1:93–99.

Pylyshyn, Z. W. 1979a. Do mental events have durations? *Behavioral and Brain Sciences* 2(2):277–278.

Pylyshyn, Z. W. 1979b. Validating computational models: A critique of Anderson's indeterminacy of representation claim. *Psychological Review* 86(4):383–394.

Pylyshyn, Z. W. 1981. The imagery debate: Analogue media versus tacit knowledge." *Psychological Review* 88:16–45.

Pylyshyn, Z. W. 1984. *Computation and Cognition: Toward a Foundation for Cognitive Science*. Cambridge, MA: MIT Press. A Bradford Book.

Rumelhart, D. E., and McClelland, J. L., and the PDP Research Group. 1986. *Parallel Distributed Processing*. Cambridge, MA: MIT Press. A Bradford Book.

Shepard, R. N. and Cooper, L. A. 1982. *Mental Images and Their Transformations*. Cambridge, MA: MIT Press. A Bradford Book.

Simon, H. A. 1969. *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

Sperling, G. 1967. Successive approximations to a model for short-term memory. *Act Psychologica* 27:285–292.

Sternberg, S. 1970. Mental scanning: Mental processes revealed by reaction time experiments. In J. S. Antrobus, ed. *Cognition and Affect*. Boston: Little, Brown.

Stevens, K. A. 1981. The visual interpretation of surface contours. *Artificial Intelligence* 17:47–74.

Townsend, J. T. 1974. Issues and models concerning the processing of a finite number of inputs. In B. H. Kantowitz, ed. *Human Information Processing: Tutorials in Performance and Cognition*. Hillsdale, NJ: Erlbaum.

Turing, M. A. 1937. On computable numbers, with an application to the Entscheidungs-problem. *Proceedings of the London Mathematical Society* 42:230–265.

Turing, M. A. 1950. Computing machinery and intelligence. In *Mind*; reprinted in 1964 in A. R. Anderson, ed. *Minds and Machines*. Englewood Cliffs, NJ: Prentice-Hall.

Ullman, S. 1979. *The Interpretation of Visual Motion*. Cambridge, MA: MIT Press.

Ullman, S. 1984. Visual routines. *Cognition* 18:97–159.

Wallach, H., and O'Connell, D. N. 1953. The kinetic depth effect. *Journal of Experimental Psychology* 45:205–217.

Wasserman, G. S., and Kong, K. L. 1979. Absolute timing of mental activities. *Behavioral and Brain Sciences* 2(2):243–304.

Young, R. M. 1973. Children's seriation behavior: A production system analysis. Ph.D. diss., Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.