CHAPTER 2

The Symbol Level and the Knowledge Level

Allen Newell

What I'm going to do today is try and summarize some of the thinking I've done about symbols and about knowledge. This is an appropriate place since the conference is on foundations of cognitive science. In some sense (I notice this mostly when I'm talking with Zenon) I seem to have an anti-philosopher's point of view and I seem to disparage this kind of activity—but then I find myself engaging in it quite a bit and that produces a certain amount of dissonance—so I have to try and understand that.

I now have at least a small idea about why I have this dissonance. Try thinking about it this way. These efforts are really a sort of commentary on the science. We try to describe and redescribe what we see going on. We're trying to draw the picture in very general terms because it's a useful and pleasant thing to do. But we don't understand a lot about the process of science as we go through it. A good example, by the way, is one raised in Jerry Fodor's talk, namely, the whole area of the New Look in perception (which by now is the very, very old look in perception because it goes back really quite a way). In psychology we have an immense amount of methodology to help us through. We have the sort of local apparatus of the experiment and all the things we beat into our graduate students about how to make the experiments tight—as if, in some sense, by agreeing with all this methodology each of those experiments will survive for a while. In fact the history of all scientific experiments and all papers about scientific experiments in psychology is that they decay rapidly over time. It always turns out that there are all these ways in which they can be attacked seriously if you simply demand of a psychologist not a 200-millisecond response to a paper but give them a week or a year. They would then find all kinds of ways in which the study is flawed, no matter how carefully the methodology was done. So we have this contrast that we just don't understand between local rigor and long-term lack of rigor in the papers. We don't really understand it, so we just keep doing this commentary, trying

to get it right. I do think this kind of activity plays an important role, and maybe that's why I keep doing it. It's really part of what one thinks of as a paradigm or a research program in Lakatos's sense; we need articulations to build our own heuristics on how to go about doing research.

The reason I have this anti-philosopher's point of view is the following: If we have trouble doing the philosophy—if a commentary isn't right, if you can shoot holes in it and there's all kinds of things wrong with it—then progress is to be made not by, in some sense, working at that high level of description but by just going back down and waiting for the science to go on. The picture here is something like this: Here's all the stuff that's going on down in the pits and here's a piece of general description and there's trouble with it because there always is. What will clarify it turns out not to be more thinking at this level but to wait until the work goes on a little longer, and then you make a fresh induction and maybe now you will see it clearly because it has cleared up. I think the history of philosophy, or at least parts of philosophy, is a testament to this since in many respects it is driven by science. The issues that arose out of quantum mechanics for the theory of knowledge also arose from scientific results happening in psychology and in many respects in cognitive science (the modern term for psychology). It's only with the advent of all these notions of information processing that you once again get a resurgence in the activity of trying to build a lot more commentary on it.

So my own activity is really a kind of a commentary, and one which is probably wrong. When other commentators show the ways in which it is wrong, my response will be to say, "Let's all go to work for awhile and we'll make a redescription and maybe it will be better," since it's not clear that fixing it up at this most general level is the right way to play the game. So what I'd like to do is just give you a precis of some observations that I've made and ask you to take them in this spirit.

The starting point is that something important is happening in computer science and artificial intelligence with respect to the understanding of mind. The indicators of this are all around. There are programs that do interesting things on interesting intellectual tasks: programs like Hacker, Shrdlu, Dendral, Internist, Strips, Noah, and Molgen. No matter what one's opinion of them is in terms of some of their details, or even some of their long-term philosophic consequences, the world is very different with respect to these mechanisms now than it was 20 years ago. Furthermore, there are many new ideas which have come into cognitive psychology, general notions like semantic nets, goal structure, means-end analysis, architecture, memory structures, production systems, conceptual dependency, scripts, and so forth. Whether one can fit these developments into descriptions at higher, more general levels in which these are a reflection of certain long-term choices and options, I think is relatively unimportant. Their interest is that they really provide the detailed structure that allows one to proceed, and that is why psychology is now in a very different place than it was.

It is also true that AI and computer science generally have an extremely strong coherence, which one would not necessarily have predicted. Lots of the people in the early days—the engineering people—didn't really see it that way, but AI (i.e., the attempt to find out the mechanisms of intelligence), which derived from concerns about neurophysiology as well as psychology and concerns about complex engineering systems with feedback, has really come to be a central component of computer science. All of this indicates that something big has gone on, and that's what one wants to understand.

My own attempt to try and reflect on this has really focussed on the notion of symbol and symbol systems. The idea is that there is a class of systems which manipulate symbols, and the definition of these systems is what's behind the programs in AI. The argument is very simple: We see humans using symbols all the time. They use symbol systems like books, they use fish as a symbol for Christianity, so there is a whole range of symbolic activity, and that clearly appears to be essential to the exercise of mind. Certainly, trying to understand the nature of symbols and symbolic behavior is an approach to the nature of mind. One doesn't have to view it as an exclusive approach. In fact, at other times when I've talked about this, I've tried to lay out a whole bunch of different constraints which show that focusing on the symbols is only one of the ways one might approach the problem.

Computer science and AI have by now an extremely well-developed notion of a symbol system. Herb Simon and I have called it a physical symbol system. That extra word "physical" is there to deal with two little problems. First one wants to emphasize that a symbol system is a *physically realizable* system. Second, one wants to be sure one sees its source, i.e., if it happens that the notion of symbol systems turns out to be an adequate notion for the kind of symbolic activity that we see in humans, that's an *empirical* discovery.

The first thing that's given in a symbol system is some kind of a *physical medium*—the *symbols* are patterns in that medium. It doesn't make any difference at all what kind of physical characteristics you're talking about, except that you can distinguish some patterns, some parameters of the physical world. In fact, there might be an inexhaustible supply of distinct patterns. That seems to be a key notion, although it clearly doesn't make much difference whether you have an unbounded set or just a very large number of them.

There are also *expressions*. I could have used the term "symbol structure" to mean the same thing. An expression is simply a domain in that medium. It takes up space and time, and it contains instances of these symbols which we can call tokens. So an expression is a little chunk of that

medium that actually holds the symbols. The symbols are just patterns, but the expressions really take up space and time and have weight (if you want to think about it that way). The medium is also arbitrarily larger—that is the usual requirement of an unbounded memory.

There are *processes* that work in that medium, and these processes are things that take expressions as input. It's the physical processes that actually do something: They take tokens of symbols, and what they do then is create, modify, and destroy expressions. There is a notion here of completeness—that is to say that somehow those processes can ultimately get all the different kinds of configurations that you may need and may want to have. One of the reasons for expressing these things so generally is that it allows us to see that there are indefinitely many different ways to realize these kinds of things, i.e., how to decompose the processing into pieces so you can get the completeness back again. And there are very many different ways of realizing it.

Now we get to the notion of what the symbols *do*, and the term I have used is the word "designates." There is the notion that symbols actually have symbolic character. What this means is that symbols give access to what they designate. If you say that symbols designate X (whatever X is), then in fact what you see in the physical structure of that system is *an access relationship*. If one is given the symbol token, then a physical access relationship allows one to get access to information about it. On the input side, the access relationshp means that you get access to other expressions which themselves in fact contain information about the thing that the symbol is about. That is recursive and transitive in the sense that what you get access to may not be the thing itself, but may only be an expression which itself has symbols which give access to other information. On the output side, one finds that there are some processes which, when activated, affect the actual entity. If I have a symbol that says ADD and it shows up in an instruction, then in fact something happens called ADDing. And if I have a symbol that says TAKE ANOTHER LOOK AT THE WORLD, then that symbol occurring in the process as input causes the process to go take a look at the world. So there are both output connections and input connections and they are slightly different.

So a symbol designates in the sense that it leads to these access paths which allow one to get other information. And that's what's really built into the notion of symbol in computer science and AI. It's built in originally in terms of access to other expressions in the memory. And then it turns out that having started out with that physical access mechanism, one is able to parlay that into the ability to get designation in a much more complete sense: designation of things in the external world.

There is also something called *interpretation*. And again I use this term as it's used in computer science. In that context it means that there are some symbols in the system which designate symbolic processes. Furthermore, the system has the property that if it has a symbol that designates a process, the system can carry the process out. There is also a kind of completeness about these symbols in the sense that I am able to write an expression to tell the system to do any of the kinds of processing that it can do. What makes the whole system work and provides the generality of behavior turns out to be that one can write expressions which designate a behavioral process to be carried out. The system then has an interpreter (in the usual computer science sense of the word) which will take that expression and produce that behavior. That's the thing that provides the generality in the system.

Then, of course, there has to be (just for completeness) a control mechanism in the system. The whole physical system is built so it runs on the fetch–execute cycle or whatever, and brings in expressions one after the other, and those expressions cause new expressions to be brought in. So its behavior really does come under the control of its symbolic characters. This is the final requirement for a physical system. Any system with all of these properties is what I want to call a physical symbol system. One sees these systems under an indefinite number of different guises in all programming languages and so forth.

There is more to the story. When you look at what's happening in computer science, the systems all have a very clear level-structure. That is, there is a hierarchy of *computer system levels*. It's a brute empirical fact. It turns out that despite all of the diversity of what goes on in computer science, this kind of hierarchy is preserved. I don't know of anyone who has a theory of why this hierarchy turns out to have an immense amount of stability, but it is one of the more stable things that one sees in computer science. You all are familiar with this in a general way. The levels are, in effect, alternative descriptions of the same physical beast. There is something called the *device* level, at which the actual devices like the transistors are described in quantum mechanical terms, and where the description of signal transmission is given in electromagnetic terms, and so forth. Then one comes up a level and gets reasonably complete descriptions of these things at the *circuit* level—where the medium is voltages and currents (sometimes some different things if you don't have an electrical computer but have a hydrodynamical computer or something like that). At the next level we have the *logic* level. The logic level is one in which I now don't see the continuous circuit or device changes at all—the element that is being processed is the bit, and it is processed by logical operations of AND's and OR's and NOR's and NAND's. At this level I have a complete description of the system. There is a level above that, which is usually talked about as the *register-transfer* level, in which we don't talk about single bits, but about registers and functions which move things from register to register

and so forth. It is still fundamentally the logic level—now done on bit vectors. Then we get up to what's usually called the *programming* level, which, for the purpose of these kinds of descriptions, I call the *symbolic level.* That's the level given by the programmer's manual where you really have a machine language or other alternatives like LISP or Snobol or Fortran or whatever. You have a language description, and there, in fact, one has a description in terms of symbols and expressions with an interpreter and a lot of serial behavior, although not always exclusively serial behavior. And that is what is universally called *architecture* in computer science. A computing machine described at the register-transfer level can realize and support the architecture of the system described at the programming or the symbolic level. There is one other level above this called the *configuration* level, which is the level at which you order computers from the manufacturer—a crude anatomical description. At this level, you don't really care about the detailed structure.

This hierarchy is extremely stable, and there are several things that are known about it empirically. One of them is that given the description of a system at one level, there are many systems at the next level up. For instance, there are lots of different register-transfer level systems. They all share common values and common elements in using bits and so forth, but otherwise they are very different. It is also true that lots of different logic circuits can be realized out of the circuit level. And so you don't get a unique kind of system, you only get real family resemblances of the system at each level. Another fact is that if I give you any specific system at any one of these levels, there is an indefinite number of different ways of realizing it further down the hierarchy. In this sense each level appears to have an autonomous description. Each new technology is used to provide cheaper, better, more reliable realizations of the same systems at the next higher level up. One of the things that is rather amazing in this respect about these systems is how stable this sort of general structural view is under radical changes in the technology. Things like the development of VLSI computers still preserve all of this no matter how different they seem in other respects.

Notwithstanding Jerry Fodor's comment that no one had been paying attention to what sort of structure systems must have, there is an immense amount of work on structure in computer science. What's important is this symbol system and the structure that it brings; one then tried to understand what goes on when all the various concepts are used—concepts like knowledge, intelligence, representation, information, and so forth. These terms are quasi-technical, i.e, they are used by everybody in the AI world in order to do their day-to-day engineering and scientific business, and one needs to ask what these terms refer to.

One of the terms that I have focussed on recently is *knowledge.* I want to try and understand what knowledge is. It has become pretty obvious, at least to me, how this terminology is used. It's really used as a *competence* notion or design specification. If you have a system that you're going to build, and you've got to put some knowledge in it, then you talk about what's in that memory as knowledge *long before* you have designed any structure to hold that knowledge. So knowledge turns out to be a specification of *what's in that memory before any structure is designed to realize it.* When you ask what that amounts to operationally, it turns out to be whatever it is that the system will be able to get out of that structure to guide the rest of its behavior. And so it seems that one uses knowledge when one is trying to describe the system without specifying its internal structure. One only posits that the system has this thing called knowledge which it can use to perform the processing which it wants to achieve. At this level, talking about knowledge and talking about goals, the driving principle is really the *principle of rationality.* The system has knowledge if in the pursuit of its goals it can use that knowledge to realize those goals. This leads to the view that there is a knowledge level which is distinct from the symbol level. So I am *proposing* that there exists an autonomous knowledge level which has not normally been distinguished in any careful way, but I am doing this because I see this in the practice of all of us.

Let's go back and discuss the knowledge level a little more carefully. I'm really trying to include the knowledge level as simply another level in the system hierarchy. It is characterized, as all other levels are, by a system. The medium here is knowledge, whereas the medium at the symbol level is the symbol structures, expressions, and so forth. The components are what I have called the goals and actions. In a sense, they provide the "body" for the knowledge to be in—because it is a *physical* system. The composition of the system is a set of goals plus a set of actions plus a body of knowledge; there is no other further internal structure. There is no internal structure because the use of this system level is to describe systems *before* the structure is designed, or in other cases, to describe systems with internal structures I do not care about or do not know. And the behavior law here (corresponding to the circuit laws and the definition of how the interpreter operates) is that the system essentially operates by the principle of rationality. So this is in some respects a strange system level, and in other respects it seems to me very similar to the other levels.

This picture of the knowledge level leads to a view of representation as nothing but the knowledge plus the access structure. The principle of abstraction at the knowledge level allows us to ignore the question of how the symbols are actually going to realize all this rational behavior. In fact, just as in all other levels of the system hierarchy, the organization at the next level below must be set so that it realizes the things at the next higher level. Consequently, the symbolic processes in an intelligent agent are such that they realize rational action when viewed in terms of knowledge and goals. What I abstract away from is the access relationship that really

exists in any representation. If at the symbol level I give you a particular memory, then it turns out that drawing knowledge or information out of that memory takes processing time, and may be difficult.

There is a strong claim associated with all of this. That claim is that in the notion of a symbol system there really is the essence of the nature of mind. Let me try to be a little more explicit. When I say that there is something essential there, I do not necessarily mean that one has the full answer. This is rather like saying that in evolution there is the essential answer to the problem of how biological organisms came into existence and how they have achieved their current diversity. That does not mean that when one posits the theory of evolution one has solved all those problems, far from it. In fact we even know that Darwin had it wrong in a number of important respects, but he set the right structure within which to address those problems. A similar example would be the cell doctrine in biology, which is the guiding structural notion in biology. This states that the architecture of the biological system is really made up of collections of cells, discrete cells that interact with each other. This again doesn't solve all the problems, but it puts one on the right path. A more recent example would be in geology, where one could say that plate-tectonics was the core of how to make sense out of the geological history of the earth. It didn't solve everything, but it helps break the back of the problem. It is my belief that we are really set on the right path now, and we are going to find out that systems of the kind described above are what are needed to have mind in the physical universe. Sitting behind that structure of physical symbol systems is a structure that will do all the things that we ultimately require it to do. It explains how symbolic behavior occurs in human beings and all the rest.

A couple of historical remarks about this: This set of ideas is really very old. The issue of getting at the nature of a physical symbol system and getting enough of the details right really all happened in the last 20 years, and it happened in a way that's very different from how it usually happens in science. It seems to be impossible to identify particular scientists who have put forth that claim as a piece of theory. We all learned about the notion, and we all used it long before we had the right perspective on it. There is a strong similarity between this whole kind of system or the notion of a universal computational system and the notion of a Turing machine, and the various notions of programming languages. Much of this is embedded in the structure of programming languages; one can learn to use those with very different views of what's going on and without really understanding this notion of symbol system. Thus, one has really developed a solution to this problem without really getting a perspective on it the way normal scientific activity does. It's a very strange way to come across such a fundamental notion, but I believe that it's historically accurate.

The key thing was the invention of random access memory. Random access memory creates something called an address, which has all the basic physical properties of a symbol; it gives access to a symbol which can then be part of an arbitrary expression. Once we had the idea of an address, then one was bound to stumble upon this whole way of dealing with symbols.

The knowledge level is a way of simply talking about what *can* be done computationally with certain symbols and processes even though in general I may not be able to know exactly how the system will behave. If I find out that I can't predict the behavior on the basis of just looking at the knowledge and the goals, and assuming the rationality, then I've found out that I have to retreat to a description of the system at a symbolic level where the actual processing that takes place can be looked at. One can then try and predict what's going to happen to the system depending on what I know about its internal symbolic processing and architecture.

Let me just summarize. A key feature of this system is that if you ask about the meaning of the symbol in a structure like this, that's not really the right way to pose the question. A symbol is really a device that gives access to knowledge that needs to be used in order to make that symbol function in the context in which it has to be functioning. If you think of an expression and there's a symbol embedded in that expression, that symbol is accessed, it is the total embedding context and expression that describes the use of that symbol. That symbol is access to the knowledge that gets it to be used. So there is no stating one simple thing that is the meaning of that symbol. *A symbol is to be viewed as a device that gives access to remote knowledge.*

The second important feature in the nature of symbol systems themselves is that if you start with a system which only has these properties with respect to its internal memory structure (addresses), one can develop it to a system in which the designation really works to the outside world. That isn't given directly by the access structure; rather it is something that one builds up computationally from it. This is the discovery that is happening as we find ourselves able to build seemingly more intelligent programs.

The Link from Symbols to Knowledge

Brian Smith

I will make no *general* attempt to judge Professor Newell's theses right or wrong: With some parts of his paper I am sympathetic; with others I disagree at the deepest levels. But what I find striking is the way it fits into the emerging consensus that a sharp distinction is gong to have to be drawn between *psychology in the classical sense of that term* (broadly construed, interpreted, epistemologically and referentially grounded—a study, roughly, of what people actually think, know, believe, and so forth) and *formal, modern, computational psychology* (narrowly construed, formally defined, computationally based— a study, roughly, of *cerebral syntax*). That there is a substantial difference between these subjects—that the millennial theories may tell different stories—is a position being adopted by theorists on both sides of many academic fences.

The Boundary

Interpreted psychology sometimes treats the mind as a black box, in a presumably transparent world; the formal or computational view has been caricatured as holding that the mind is a transparent box in an opaque world. What the two views share is that the *boundary* between the mind and the world is of rather major theoretical consequence. That it is of *some* consequence would be hard to deny: Even the simplest view would presumably suggest an internal theory of a computational process defined over an uninterpreted formal language, with a corresponding model theory attributing external reference to its terms. Those that take the boundary seriously, however, argue that theories that remain inside, and those that seek to explain the culturally and socially embedded intellect, may traffic in different concepts, as well as explaining different phenomena. In the extreme, they might belong to different disciplines.

That internal and external theories must be distinguished has been argued, in various perhaps incompatible forms, by Putnam, Fodor, Stitch, Dreyfus, Chomsky, and Winograd, among others. Their arguments share the realization that the terms of lay psychology inherently involve the structure of the world in which we are embedded, are perhaps inextricably tied into our cultural and human history in a way that a purely *internal* model of what happens in the head can never reveal. Many examples have been given: The true content of the term "water" depends in part on the fact that water is H_2O; the real meaning of the statement "she believed he was a menacing intruder"—such as would matter in a court of law or ethics—depends on world knowledge about the appropriate social contracts, intertwined beliefs, and so forth. In contrast we have what we call the *computational* account: as yet without rigorous definition, but evidenced by recent research in artificial intelligence, cognitive psychology, and so forth. "Thinking is spelling," "the mind is a computer," "the syntax of thought," "methodological solipsism," and so forth: These slogans convey the hunch that a functionally construed computational account of the internal workings of the human mind, specifiable without recourse to the surrounding world, might even be true.

In order to use the term "bedfellows" for all advocates of the position that we take the boundary seriously, I would need a large bed, for there are substantial differences among their views. There are disagreements about the theories on both sides of the boundary: on whether, for example, the notions of interpreted psychology are, or ever will be, susceptible to "scientific analysis; on whether the computational project will succeed, even in its own terms; on whether a universal mental code or a more modular faculty-specific organization will ultimately be uncovered; and so forth. And finally—and this is where I want to focus, in order to locate Professor Newell's paper—there are differences in opinion on how these two subjects will turn out to relate. From the fact that they are *different* it does not follow that they are incompatible—that their theoretical categories are irreducible. One might think that interpreted psychology and cerebral syntax would mesh as thickly as organic and inorganic chemistry. It is possible to hold, however, that no type–type regularities of any sort will emerge—that, in Fodor and Block's phrase, interpreted psychology might even be a *special science* with respect to the correct computational account of psychology narrowly construed. It does not follow, just because the computational account may *itself* stand in a special science relationship to the underlying electronic or neural physiology, that we can't have *another* lack of type–type regularity. This even if one remains a mechanist: There are surely legion ways that theoretical terms can *fail* to correlate.

It is as an emergent argument for the importance of the boundary that I will attempt to cast Professor Newell's paper. In these terms I will look briefly at his conception of what he calls the knowledge level, at the "symbol system" characterization of computation, and at the correspondence he draws between them.

The Knowledge Level

With regard to his account of the knowledge level, I must confess that I have considerable technical difficulty with many of the details. I am not

inclined to accept, for example, his characterization of a computer system level in general. While it is undeniable that many different—radically different—accounts of computational devices find common use, I don't understand what it is to say, for example, that a level "really exists," rather than being a point of view. Usually there are theoretical frameworks under which we describe computational devices; in one sense these are viewpoints, but that fact doesn't make the objects viewed from those viewpoints any less *real*, or descriptions in the viewpoints' terms any less *true*. I may describe the Nahanni River Canyon as a trap for paddlers' souls, and as a glacial scar; both statements may be true, without challenging the unproblematic existence of a single river.

I find it hard to agree, in addition, with his description of what are supposed to be the *standard* computational levels. For one thing, surely they are not *physical;* the fundamental distinctions in computer science, like that between *serial* and *parallel* processes, have never been— and I see no reason to suppose that they ever will be—defined as *physical* predicates. The notion of *symbol*, in addition, which Professor Newell claims belongs internally and straightforwardly to a specialization of a physical-state-like device, I will argue below is neither straightforward, not internal, nor physical (this, as you might guess, is going to matter). Finally, the notion of *implementation*, which has to do with what it counts to be a valid "level" in this hierarchy, is a theoretical construct in sore need of explication, rather than something we can uncritically rely on. Like Professor Newell, I don't want to impugn our lack of current understanding, but I cannot accept his assumption that these are scientific terms with a clear received definition.

Turning to the knowledge level itself, I would also question his definition of *rationality* as derivative on a notion of *goal*, in part because that notion would seem to me either wrong or universal (*anything* that an agent does would seem to be tautologically a goal). I don't see why a goal need be written down in anything like an explicit language, and certainly a goal is not something that *arrives* into a computer through peripheral sensors. On another front, I don't know what to make of his notion of logic as the appropriate tool for *analyzing* knowledge, except as an admission that logic makes a good formal language in which to express one's meta-theory—likely true. I disagree, however, that logic is even a *candidate* for a *computational* representation language: Such a position embodies a virtual category error, since there is *no notion of control structure in logic*, and *no concept of accessibility*—even, really, of token identity—for the expression involved. Inference rules say what can be done, not what should be done *next*. Proof methods, like natural deduction, suggest *strategies*, but even they don't touch the question, crucial to a field of computational symbols, of what sentences are *accessible* from what others. Logic, though formal, is not concerned with actual forms.

The real question, however, is whether these technical difficulties matter. I think that they do not, given our common inchoate understanding of the overall framework. Of more concern is the general *character* of the knowledge level Professor Newell espouses, a character betrayed in the "surprises" he describes it in terms of. In brief, it would seem that knowledge, and such theoretical accounts of it as we are likely to get, is on Professor Newell's view neither *computational*, nor *formal*, nor *compositional*, nor *reductive*—not, in fact, what a strict scientism would call generative or explanatory.

That it is not *computational* is revealed in his claim that knowledge as a medium lacks *structure*. The clear meaning of "structure" here is one of *formal* structure—an organizational pattern accessible to a device or process that does not have access to the *content* of the knowledge. That it is not compositional is explicitly admitted, in the denial of composition laws. The knowledge level is also constrained, not by a behavior-engendering mechanism, but by a constraint that rationality must satisfy: It is again, by explicit admission, radically incomplete. Thus we would not expect a generative account of behavior to emerge from it. Finally, the non-computational sense is reinforced by the claim that the *medium* is not a state-like physical structure.

If knowledge is *not* all of these things, what can we say about it that is positive? First, that knowledge is *interpreted* in the sense we are using that term:

> Knowledge, in the principle of rationality, is defined entirely in terms of the *environment* that is the object of the agent's goals, and whose features therefore bear on the way actions can attain goals. (Newell, 1982; emphasis in the original)

One could argue that by environment Professor Newell *may* mean not the actual environment, but the sense impression impinging on the organism—thus defeating any attempt to connect the concept of knowledge with matters of truth and reference. His claim of equivalence between the AI use of "knowledge" and the philosophical use of "belief" supports this reading. On the other hand this suggestion can be countered by his discussion of how the knowledge level *works*, for it is plain that the knowledge he speaks of is knowledge of the sort that an observer would ascribe to another intelligent agent in understanding his plans, predicting his behavior, and so forth. It is exactly, in other words, the kind of knowledge that the lay use of the term signifies, almost by definition. The second positive claim on "knowledge," then, is that it is the knowledge of folk usage. Professor Newell, I should add, is surely right in this: Whatever it comes to to believe something, and whatever we mean when we say that someone

knows something, the way we treat other human beings is surely affected by what we believe them to know. You have to model folk psychology if you want a theory of what it is to be a folk.

Another property of Professor Newell's knowledge level is that it is infinite—a kind of Chomsky-like competence, without regard to resource limitation or performance considerations. There is an empirical claim buried here: that the proper and natural account of such phenomena as forgetting, difficulty of chains of reasoning, and so forth will receive their natural explanation at the computational level of explanation of mind—*internally*, in other words—rather than at the knowledge level! This claim, it should be noted, is at odds with that of many adherents to a proceduralist approach, who believe that the only natural essence of these terms will lie in their resource-constrained reconstruction. But there are far more issues here than we have time to explore.

In sum, it is the task of the kind of analysis of knowledge that Professor Newell is committed to, we may presume, to rationally reconstruct what is essential and paradigmatic about the use of these terms *in their lay setting*. They may of course be radically reconstructed, split apart, or unified, in the best tradition, but the *subject matter* cannot be changed. On the computational side, however—what the *symbol level* looks like—there is less reason to think we owe allegiance to any pre-theoretic body of phenomena or intuition. The judge we must submit our theoriest of computation to will be one that judges them correct articulations of expert practice. It is in this light that I will examine Professor Newell's account of a symbol system.

Symbols in Computation

My difficulties—and I have difficulties—can best be illustrated with a story. I recently moved to that part of the world known as *Silicon Valley*, where it is a favorite entertainment to tinker in one's garage for awhile, start a company, and earn a million dollars. Suppose I take up this practice, and invite you over to see a demonstration of my new world-shaking computer. "It calculates orbital trajectories," I claim, unveiling a large object made primarily of steel, but resting on what looks for all the world to be four wheels. "I will demonstrate," I add, and, taking keys out of my pocket, open the door, climb in, and drive off into the sunset.

You would have cause to claim that I, perhaps influenced by the nature of the workplace, had built a *car*, rather than a *computer*. My point is merely this: We do not yet have a workable definition of what piles of metal and silicon we choose to call *computational* as opposed to those we do not. Certainly that the device be *rule-governed* is not sufficient, for we surely hope that cars are rule-governed. The lambda calculus, Turing machines, Post production systems, and so forth aren't much better an answer, for two reasons: They are only exemplars, and they are not physical. Rather, we have an informal notion that somewhere within the concept "computation" there is a consensus that language-like structures—some constituents or patterns of constitutents—should act as causal ingredients in producing the overall behavior.

Not, mind you—and this is the crucial point—*qua symbols*. If computer science is anything it would seem to be the study of *formal* symbol manipulation, which I take it means that the symbolic ingredients are treated *without regard to their externally attributed semantical weight*. Just as a proof procedure is denied access to the interpretation of the sentences under its jurisdiction, and just as an adding machine has no access to the set-theoretic number that is encoded in the arrangements of its parts, so too the computer cannot behave in virtue of the reference of the ingredient constituents. Such a design would violate the foundational notion of computation.

It seems to me utterly clear that every computational artifact shares this property of having semantical weight attributed to its elements—both to its internal parts, and to the inputs and outputs that we call its behavior. This attribution, moreover, contributes to some of the computational magic: We are able, it so happens, to construct devices that are *semantically coherent* in spite of the fact that they are *formally defined*. It is exactly when our computational design coheres with our semantical attribution that that result can be viewed as significant. Thus when I type in "(= 3 (+ 1 2))" and the computer types back the symbol "T" I am delighted, because I know that (= 3 (+ 1 2)) is *true*, and I always take that symbol "T" to designate truth. The computer *preserved designation*, without, so to speak, knowing it.

I believe that this external attribution of significance is at the heart of the notion of computation. I trust that it is clear that a view more different from Professor Newell's is hard to imagine. I would argue that whatever I build in my garage can fairly be called a computer *just in case it is naturally understood in terms of the interaction, in virtue of their form, of a set of ingredients that you and I take to be symbolic*. Computers, in other words, are on my view just those devices *whose functional constituents naturally succumb to external semantical attribution*. My sunset device is not best so-understood; therefore it was not a computer.

What, then, are we to make of Professor Newell's claim that within computer science we have at last a notion of symbol, internal to our science, and freed from the vagaries of the human use of language? Frankly, I do not know. Striking? Yes. Radical? Yes. True? I fear not.

To substantiate this claim, we must look first at Professor Newell's characterization of symbols as *causal mediators* of some sort. On the face of it,

this characterization is surely too broad. For if I trip over the chair you left in the bathroom at night, does that mean that the chair is a symbol of you? It would be, by his account, since my subsequent floor-directed behavior, after "taking the chair as input," is dependent on you. By his account, in fact, *every* causally dependent object would be a symbol of its history. This is not the study of *symbols:* It is closer to physics. That the view is too inclusive is revealed as well in his claim that designation is transitive: Surely causal dependence is transitive, but to say that *designation* is transitive would deny the coherence of a use/mention distinction. The expression *"NILE"* designates a short word, which in turn designates a long river, but—if our education has taught us anything—the quoted term *"NILE"* does *not* transitively designate the river.

If internal causal relationships are not true designation, why might Professor Newell call it that? There are two answers that can be suggested. First, it may be that that is how he (in some sense correctly) *understands* those causal relationships. For I would even be so bold as to assert that Professor Newell attributes semantics to a large number of computational symbols. Truth values, numbers, functions, and so forth are none of them *structural* entities—they are all mathematical abstractions. And yet boolean constants, numerals, predicate letters, and lambda terms permeate our computational languages.

Another reason one might take designation to be internal to computer science—to be fair—is that the term is so used within the field. But, I would say, the lesson one draws from that fact is that by "designation" they do not mean "designation." Consider, for example, the claim that programs designate the processes they engender. Although Professor Newell is correct in claiming that that is how the word has come to be used, that usage is, so far as I can tell, simply *false*. It is certainly mathematically coherent to proceed in this fashion—it may even be singularly useful—but from neither of those facts can one argue that the relationship between program and ensuing process is *designation*— a term, after all, of English. Programming language semanticists have the unfortunate tendency, so far as I can tell, to *define* as the denotation of a program whatever is required in order to accomplish their task, which is to make manifest the total machine consequences of the occurrence of a structure within the machine. Thus even numerals are given designations of functions from cross-products of machines states and environments to tuples of input/output streams and memories and so forth. This notion, useful though it may be, is hardly to do with the natural use of the term "symbol." I should have thought it should take considerable argument to convince us that the numeral "3" designates anything other than the successor of 2.

It should be admitted as well that standard computer-ese is rife with use/mention errors. The claim that early languages worked only with *numbers*, for example, is strictly incorrect: They worked only with *numerals*. In LISP, the grandfather programming language of artificial intelligence, numerals are in fact mapped by the LISP "interpreter" *onto themselves*—onto expressions, in other words, not onto the numbers they unarguably designate. *That's* not interpretation, in the classic sense. Computational processors, in fact, can't interpret *any* term whose designation lies external to the machine, since they are functions from structure to structure. (I have always taken it as one of the great jokes in computer science that processors that perform the formal symbol manipulation are called "interpreters," since the fundamental fact about how their behavior depends on the symbols they manipulate is that they *cannot* make reference to what those symbols designate. In fact, the number of terms that are technical in computer science and philosophy of mind, with different meanings, is astounding: "reference," "interpretation," "memory," "semantics," "value," and even, I sometimes think, "theory" and "explanation." It would be fun for a conference like this to train U.N.-like experts in simultaneous translation, not between *languages*, but between *disciplines*.)

There is one final point we have to make before leaving this subject: There are, as it happens, circumstances in which it is arguable that the computational processor does interpret. These arise when the external attribution of reference to computational ingredients maps some of the symbols onto others (when, in other words, the syntactic domain is included within the semantical domain), and when the function computed by the language processor follows this designation relationship. Note first what would be required even to *show* this. First, one would have to specify a theory of semantics for the computational structures *independent* of their procedural treatment (something that is never done, except for a few languages based on logic); second, one would have to give an account of what the processor does with expression (something we might call the "procedural semantics"). Third, one has to show that the "value" or whatever it is called that is returned by the processor is the designation posited by the independently specified declarative semantics. As it happens, this is possible to do, although it is quite considerable a project.

By following such an approach, one can in fact show that computational formalisms are full of what we take to be meta-linguistic expressions. But this fact, although it increases the complexity, doesn't make the semantics *internal*—it is still mediated by our minds. Imagine by analogy that I give you two sheets of paper, one with sentences in a first-order language, and another with an axiomatization, in some appropriate meta-language, of the model theory for the language used on the first sheet. I haven't thereby given you something with internal semantics: It is still two sheets of uninterpreted formulae. As it happens, the standard model of the meta-theoretic language may take one of its predicates onto the reference relationship for

the first-order language, but that doesn't deny the fact that the semantical import of the expressions on either sheet, or on both sheets stapled together, is still in need of attribution.

Now in response to this line of argument Professor Newell might correctly reply that whereas it would be difficult to establish any causal connection between the expression on my two sheets of paper, there is in the computational case an undeniable causal route from designator to referent, since, *ex hypothesi*, the computational processor is able successfully to de-reference such terms. In fact these causal relationships are crucial to Professor Newell's theory—witness his discussion of access and assignment. However it *still* doesn't follow that we would necessarily want to call any mechanism able to mediate between symbol and referent a *user* of the symbol *qua symbol*. Imagine, for example, that I build a Tinker Toy device that holds a book by my bed, so that when I punch in a page number it opens the book to that page. Does this device count as a candidate for having *used* the language of page numbers? While I must admit that the example is not decisive, I don't think the answer is clearly affirmative. Where, for example, is the *intentionality* (I trust you heard the "t") we are all supposed to endorse? (There is an even stranger fact about de-referencing mechanisms. There is a sense in which any mechanical means by which a symbol is associated with its referent *must* be formal, tautologically, for if it were not formal, it would have to use the referent in order to find the referent, which is vacuous. I do not see any way around this conclusion, but it is curious at best.)

But these are all subtleties, inappropriate for present discussion. They apply, furthermore, only to that subset of symbols given meta-linguistic status by the pre-computational external semantics. The present moral is merely that the use of the term "symbol" in computational contexts—especially if one wants to claim some relationship between that term and its normal use in natural language—is a matter fraught with complexity, subtlety, and tacit attribution.

The Relationship Between Symbols and Knowledge

We turn, finally, with very little time left, to the question of the relationship between a computational account and an interpreted theory of knowledge. I have in fact very little to say about this, in part because Professor Newell himself does not spell out many details. He calls the relationship "representation"—for purposes of discussion, at least, I can accept that, although whether any "representing" is involved I don't know. I take it also that Professor Newell doesn't expect that representation relationship to be problematic, although I don't share his intuition here, nor do I see the argument.

I do, however, have a question about what the relationship is *between*. At one end I take it we have a computational system (as opposed to a *theory* of computational system—i.e., we are not talking about a theory reduction), but at the other we have *knowledge*? My semantical attribution function—my model theory, essentially—would take computational symbols by and large *into the world*. It is also true (this arose out of a survey . . .) that most people in the "knowledge representation" business use the term "representation" this way—as between symbols and *world*. I am happy with Professor Newell's use of "representation" as a relationship between symbols and knowledge, but it seems difficult to know what the account of a relationship between a formal, compositional system and an explicitly non-structured, non-compositional medium might come to. Is there any reason to suppose, in particular, that it is formulable?

I have another question. If I am right that the computational symbols derive their "symbol-hood" from external attribution—if, in other words, the concept of computation is itself derivative on a notion (reference) emerging from interpreted psychology—then what are the consequences for our understanding of the relationship between a computational (internal) account of mind and an interpreted psychology of mind? I myself have no idea.

Finally, I have a comment: Boundary problems have attended pure computational systems as well. Certain terms of art in computer science ("thrashing" is a common example) have developed that describe rather well the external behavior of a computational system, but that bear no straightforward relationship to any discernible interior ingredient. It is often extraordinarily difficult for the designers and constructors of computational systems to correlate compositional accounts (the programs, essentially) and manifest behavior, even when remarkable adequate *external* accounts of that behavior are at hand. It is for reasons like this, as well as for many advanced by Fodor and Putnam and others, that certain theorists —Winograd (1972), for example, is a salient case—have come to believe that *no* computational account of mind will emerge—or that if one does, no interesting relationship between it and interpreted psychology will be forthcoming. My only comment here is that if such boundary issues are problematical in the pure computational case, do we have any reason to suppose that a *simple* representation relationship can be found for theories of mind?

Conclusion

I have argued that what *distinguishes* Professor Newell from other cognitive scientists is not his identification of the knowledge level: This is strikingly like the boundary consensus I mentioned at the outset. Nor is it his

acceptance of the potential power of a computational account, although his *description* of that level is I think seriously in error. Rather, the point of difference—and this, it seems to me, is where we should focus—is on the question of how a computational account of cerebral syntax, and an interpreted theory of human knowledge, might be related.

It is here that my questions regarding Professor Newell's use of the term "symbol" acquired their force. For *if* computational symbols are not crucially semantical within the computational arena, but rather derive their symbol-hood from external attribution, then it would seem more difficult than Professor Newell has implied to draw a simple relationship across one boundary.

Commentary on Newell's Paper

Is There an Autonomous "Knowledge Level"?

Daniel Dennett

I'll try to be brief since lunch awaits us. I've rewritten my comments about five times this morning. I agree in very great measure with what Professor Newell has to say in his paper on the knowledge level. Indeed my idea of the intentional stance as a very close cousin of his idea, as he has said. Our agreement is very widespread on that point. I'm afraid our agreement even consists in our both being fuzzy in the same places about some of the really important issues. Perhaps the most useful thing I can do here is to point out some of those places and try to shore up Professor Newell's fuzziness in a few places with a little bit of slowly emerging clarity of my own. He may disagree with my attempts to shore him up, of course.

Is there a knowledge level? Do we really need to talk this way? I think the answer is "yes," and I'll try to say why; I'll also try to underline and support a few of the things Newell said about that. Here's one of the questions he's a bit fuzzy on. Is the knowledge level in fact a level of system description? On one of the slides he doesn't show it, on the next slide he does show it, as a level of systems description. But then when he says it's a level of system description, he says it's a competence model; I think that's just right. It describes the system only by saying what the system ought to be without saying how you're going to achieve that. It is a task-setting description, as it were, setting a task for the designer rather than telling the designer any feature that the design is supposed to have other than this particular competence, this particular bit of knowledge. Knowledge is an extremely abstract commodity, but we do have it. Each of us has different amounts of it on different topics. But in virtue of what do we have it? Many people when faced with a question of this sort get scared. They are afraid of becoming philosophers or they are afraid of dualism. They're afraid of abstraction. So they try to answer the question of "In virtue of what do we have knowledge?" by reducing knowledge to something a little more concrete. I won't review the history of various forms of crude materialism that were replaced with forms of less crude materialism which were replaced with forms of still less crude materialism. As we march up the ladder through Turing machine functionalism we finally get to a view such as the view in Fodor's *Language of Thought* (1975) which I think in this regard is very similar to Newell's view that what we're going to reduce knowledge to is the physical symbol system level. This question of

acceptance of the potential power of a computational account, although his *description* of that level is I think seriously in error. Rather, the point of difference—and this, it seems to me, is where we should focus—is on the question of how a computational account of cerebral syntax, and an interpreted theory of human knowledge, might be related.

It is here that my questions regarding Professor Newell's use of the term "symbol" acquired their force. For *if* computational symbols are not crucially semantical within the computational arena, but rather derive their symbol-hood from external attribution, then it would seem more difficult than Professor Newell has implied to draw a simple relationship across one boundary.

Commentary on Newell's Paper

Is There an Autonomous "Knowledge Level"?

Daniel Dennett

I'll try to be brief since lunch awaits us. I've rewritten my comments about five times this morning. I agree in very great measure with what Professor Newell has to say in his paper on the knowledge level. Indeed my idea of the intentional stance as a very close cousin of his idea, as he has said. Our agreement is very widespread on that point. I'm afraid our agreement even consists in our both being fuzzy in the same places about some of the really important issues. Perhaps the most useful thing I can do here is to point out some of those places and try to shore up Professor Newell's fuzziness in a few places with a little bit of slowly emerging clarity of my own. He may disagree with my attempts to shore him up, of course.

Is there a knowledge level? Do we really need to talk this way? I think the answer is "yes," and I'll try to say why; I'll also try to underline and support a few of the things Newell said about that. Here's one of the questions he's a bit fuzzy on. Is the knowledge level in fact a level of system description? On one of the slides he doesn't show it, on the next slide he does show it, as a level of systems description. But then when he says it's a level of system description, he says it's a competence model; I think that's just right. It describes the system only by saying what the system ought to be without saying how you're going to achieve that. It is a task-setting description, as it were, setting a task for the designer rather than telling the designer any feature that the design is supposed to have other than this particular competence, this particular bit of knowledge. Knowledge is an extremely abstract commodity, but we do have it. Each of us has different amounts of it on different topics. But in virtue of what do we have it? Many people when faced with a question of this sort get scared. They are afraid of becoming philosophers or they are afraid of dualism. They're afraid of abstraction. So they try to answer the question of "In virtue of what do we have knowledge?" by reducing knowledge to something a little more concrete. I won't review the history of various forms of crude materialism that were replaced with forms of less crude materialism which were replaced with forms of still less crude materialism. As we march up the ladder through Turing machine functionalism we finally get to a view such as the view in Fodor's *Language of Thought* (1975) which I think in this regard is very similar to Newell's view that what we're going to reduce knowledge to is the physical symbol system level. This question of

what's the relationship between the knowledge level and the physical system level is really another way of asking of Fodor whether he really means to reduce psychological features such as believing that p, for some proposition p, to being in a computational relation to a particular formula, which then, via the semantics of the symbol in the formula, can be seen to mean that p.

My view is that the attempt to reduce the knowledge level to the physical symbol system, or Fodor's attempt to reduce it to a relation between a system and a syntactic expression, is a mistake; it is the last vestige of the reductionism that we want to get rid of. We want to agree that the knowledge level is an irreducible level of characterization, and we should learn to get over our fear of that. We should be willing to accept that we can have an abstract characterization of a physical system in terms of the knowledge that it has in it without then reducing that, via the cascade of system levels, down to the hardware. What we're left with is the idea that every piece of particular knowledge, every particular intentional system or physical symbol system with knowledge, is after all just some physically realized system, and it will not be any accident that it has the knowledge it has. But we won't suppose that we can achieve anything like a type–type reduction. Newell spoke very much in passing about one of the important reasons for this. You need the knowledge level and you need it unreduced because I know things that you know. That's something we can have in common. Now at the physical symbol level we may be very different. If that's not obvious in the case of two people just consider two computer systems, one of which uses production systems and one of which uses some other LISP-based virtual architecture. At the knowledge level they may have something in common. Maybe they both "know" that *higher* is transitive, or maybe they both "know" that you shouldn't sacrifice your queen for a couple of pawns under most circumstances. These are features which they have in common and which are only describable at the knowledge level because their architecture, and their symbol systems, are different. They have different languages of thought. This, I think, has always created a problem for somebody like Jerry Fodor who has the problem of how to describe the transaction that occurs when one person speaks a sentence of natural language to another person and thereby manages to bring about a sharing of a belief. What if they don't have the same language of thought? How is he going to characterize that?

Another reason I think that the knowledge level is autonomous and irreducible (and important) has to do with the point that Brian Smith was getting at. I really can't take the line on designation and on reference that comes out of Professor Newell's account of symbols. We get the idea that a symbol designates if it gives access to a certain object or if it can affect a certain object. And this almost looks all right as long as what we're talking about is internal states. If you have a symbol that you want to say designates some subroutine and you say that the proof of the pudding that it designates that subroutine is that when it is tokened in the system it calls that subroutine, it actually brings it on the scene and gives you access to it or it changes something in it, then it looks all right. But of course the real problem is that that isn't what reference is all about. If that were what reference were all about, then what would we say about what you might call my Julie Christie symbol problem. I have a very good physically instantiated symbol for Julie Christie. I know it refers to her, I know it really designates her, but it doesn't seem to have either of the conditions that Professor Newell describes, alas. How do you solve the Julie Christie problem, the problem of intentionality, how do you get the aboutness into your physical symbol when the physical symbol is to refer to something outside the physical symbol system in which it resides?

Here I think there are two fundamentally different ways that are being considered by various people. Intentionality is an ancient problem, and it is very murky if you try to understand it in the old-fashioned Brentano sense. One response to the murkiness is to declare that intentionality is just aboutness, and we have one clean model of aboutness: the reference of terms in a language. We can take the mysterious notion of the aboutness of knowledge, or belief, and try, via something like the language of thought hypothesis, to reduce the having of knowledge to the having, quite literally, of formuli, mentalese sentences, in the head. And then the problem of the aboutness of knowledge reduces to the problem of the semantics of the language of thought. Then we call on linguistics to solve that problem for us, so if we have the semantics of the language of thought, we can do the semantics for the symbols at the symbol level.

I suggest instead we should try to solve the problem the other way around. You can't really make sense of something as a symbol in a system unless you've got lots of other symbols in the system and they're interacting in all of their various ways. The question of whether they are symbols and if so what their reference is can only be answered by going back up to the knowledge level, and seeing whether or not the activities of these symbols subserve the control activities of a system which is understandable as a system that has knowledge. Then you see what the knowledge is *about* by looking out in the world, to see what things in the world are dealt with knowledgeably; by having understood the intentionality of the system at the knowledge level, you can then go back down to your symbol level and say, "Well" (and here is where approximation comes in) "it's because these actual physical patterns play the roles they play in subserving the control of this system which has this knowledge that we claim that this little bit in this symbol really is about, as it might be, Julie Christie." What I'm suggesting is that in order to do anything remotely like procedural semantics

you can't do it directly at the symbol level at all; you have to go back up and talk about the global-knowledge level characterization of the system. Once you see what it has beliefs and desires about at that global level, you can go back down and talk about the interpretation of the physical symbols.

CHAPTER 3

Problems in Procedural Semantics

William A. Woods

Introduction

In this paper I would like to address the question, "Is there a concept of semantics emerging in the work in artificial intelligence that can supplement the classical ones?" I believe that the answer is yes and will attempt to present here a sketch of such a notion. In Woods (1981), I present arguments for an approach to the semantics of natural language involving a distinction between the natural language whose semantics is being characterized and an internal "language" or notation in which the meanings of the external language sentences as well as a vast number of prelinguistic facts, hypotheses, goals, etc. can be expressed, remembered, believed, doubted, reasoned with, etc. The characterization of a semantics of a natural language thus consists of two steps—the characterization of rules of translation from the external language into the internal notation and the characterization of the semantics of the internal notation. The first step involves dealing with the ambiguity present in natural languages and can be dealt with by mechanisms that are essentially systems of nondeterministic translation rules, while the second step requires something considerably more subtle. The critical issue then becomes how to characterize the semantics of the internal language.

Also in Woods (1981), I presented arguments for the position that a suitable notion of abstract procedure could be used as the vehicle for a semantic characterization of the internal language, a vehicle which could in principle, I claim, account for the interaction between the meanings of sentences and our sensory experience of the physical world. In this paper, I want to continue the development of those ideas and address a number of technical problems that need to be worked out to develop a comprehensive procedural semantics for natural language.