Psychology as Physics

by

Clark Glymour

July 1987

Report CMU-PHIL-4

Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

Psychology as Physics

by

Clark Glymour

July 1987

Report CMU-PHIL-4

Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

Psychology as Physics

Clark Glymour[1]
Carnegie Mellon University
and
University of Pittsburgh

In one passage, responding to Wittgenstein, Noam Chomsky seems to say that cognitive science can be behaviorist, to its peril, or it can be fundamentally about neurophysiology, but it cannot be anything else:[2]

> "It is thus perfectly possible that certain psychological phenomena *cannot* be investigated physiologically, because physiologically nothing corresponds to them." It would also follow that these phenomena cannot be investigated within the framework of a theory of mind of the sort I have been discussing. Since physiologically nothing corresponds to our talking, writing and thinking, on this view, there can be no theory of mental structures and processes that attempts to formulate the properties of the nonexistent physiological mechanisms and their operation. We are left with a descriptive study of behavior, potential behavior, dispositions to behave, and so on...

Most of contemporary philosophy of mind and much of contemporary cognitive science supposes that Chomsky has missed an alternative possibility, and the most important possibility at that, *functionalism.* Linguistics and psychology can and should, on this view, aim at describing the *functional* structure of mind, and in doing so need not be (and in general cannot be) describing the *physical* structure of the brain or the nervous system. This sort of view is widely ascribed to Freud, and perhaps at times Freud even held it, although the history is unclear. It is one reading of some defenses of the "autonomy" of psychology[3], and seems to be held by many cognitive psychologists. There are really two different proposals for a functionalist alternative to the view that the study of cognitiion must be either behviorism or brain physics. The least interesting idea is that mental states are *functional states,* characterized by their causal relations, and *therefore* cannot be physical states. The second and more interesting thesis is that the proper aim of cognitive science is to describe an *abstract* feature of peoples' brains, the programs they execute. On this view, brains execute programs, and those programs, or features of them, can be described by a program in LISP or PASCAL or some other high level programming

---

[1]This essay owes so much to conversations with Kevin Kelly that any good ideas in it should be regarded as joint.

[2]*Rules and Representations*, p. 50.

[3]For example in J. Fodor, *The Language of Thought*

language. The LISP functions cannot, on this view, be identified with any physical properties of the brain, but they do describe features of a program in the language of thought, a program the brain executes. The same idea has now gained currency among those who advocate "connectionist models" of cognition; the connectionist programs do not describe physical properties of the brain, they describe the "connectionist level," the connectionist program the brain somehow executes.

All said, I shall argue that Chomsky is right.

1. Functional States and Computational Systems

Programs are generally given in some high level programming language, PASCAL or LISP or OPS5, for example, but programs can also be given in languages that make more explicit reference to a computational system. For example, one can give a program by specifying a Turing machine architecture and giving the machine table of a Turing machine, or by assuming a universal Turing machine and giving the tape encoding of a particular Turing machine it is to simulate. There are a great many other possible computational architectures: Wang machines, cellular automata, nondeterministic Turing machines, Sheperdson–Sturgis machines, Post machines and an infinity of structures that have never been and never will be specified. Each of these kinds of "machines" provides a characterization of *computation*, and for each kind, and each recursive function, there is a machine of that kind that will compute the recursive function. All of these machines are, in the first instance, formal objects, not physical objects.

A *program* can be thought of as a particular machine of a particular kind. Under appropriate input/ouput conventions, and appropriate individuations of machines, there are some machines that will in different initial states compute different functions, and there are machines such as Universal Turing machines that will, in the same sense, compute every recursive function. Another way of thinking of a program is as an initial state in an universal machine of a particular kind.

There is another sense of "program," namely a syntactical object, lines of code in LISP or PASCAL or some other programming language. But lines of code do not do any computations, they produce no output from any input. When we speak of a LISP program, there is usually nothing determinate we are denoting. The code could be compiled into many different machine architectures, into a RAM machine or a Turing machine or whatever. Different compilers in the same machine will give different programs for one and the same LISP code, and different compilers for different kinds of machines will give still further different programs for one and the same LISP code. We could imagine a specific universal machine--say a RAM machine-- and a specific compiler that translates any well formed

LISP code into a program in that machine, and adopt the convention that the program given by a piece of LISP code is the corresponding program in the "virtual" RAM machine. In any case, to the syntax and to the input/output conventions, there must be added some stipulation as to what constitutes a computation.

Turing machines represent a kind of compromise between the formal and the physical. On the one hand a Turing machine can be viewed entirely formally as a set of quintuples, possible inputs can be viewed entirely formally, and the "total state" of a Turing machine at any step in the processing of an input can be formally described. On the other hand, one tends to think of Turing machines in physical terms, and that was part of Turing's motivation. The machine has a "read–write head," a "tape," and some sort of physical mechanism whose states are the machine states.

Viewed as a formal object, a Turing machine has components which we call "machine states." Further, for any formal Turing machine and any formal input, there is a formal sequence whose elements we can call the "total states" or "instantaneous state descriptions" of the Turing machine. Suppose we call any syntactic features of any elements of any such formal sequence of an abstract Turing machine *functional states*. Then any actual physical Turing machine that instantiates that same formal Turing machine will have physical states, or aspects of physical states, that instantiate the functional states of the formal Turing machine. The physical states will have causal relations with one another, something the functional states, because they are abstract objects, cannot have. Corresponding to the formal total states there will be total physical states in the actual machine; corresponding to the formal sequence of total states of the formal Turing machine when it processes an input, there will be a sequence of physical states in the actual, physical machine, and each physical state will cause its successor. We say that under an interpretation the physical machine *executes* or *instantiates* the program of the formal machine.

The notion of instantiation bears some analogy to the familiar logical and philosophical notion of "interpreting" a formal, syntactic system. We can specify that certain syntactic elements, call them data structures, in a definite programming system, designate specific physical states or features of a particular physical system, or of any physically similar systems. When such a specification is made, a program in the language specifies a sequence of physical states for any physical input whose features are also denoted by data structures in the language. If the sequence of data structures generated by the formal program for every formal input corresponds to the sequence of physical states for every corresponding physical input, we say the physical system executes or instantiates the

program.[4] So understood, an *interpreted* program is just a computable physical theory; running the program on a computer can then give a simulation of the physical process, and that is one of the reasons why computers are useful. "Physical" is unnecessary; the syntactic structures could be taken to designate social features or economic features. Running programs can simulate economies as well as 4-bit counters.

One can imagine investigating empirically what kind of computational system a particular object--a computer on Mars--really is. Given tentative identifications of physical quantities that represent input and output, one could investigate whether particular physical structures serve particular functions--whether a piece serves as memory, for example, or as a logic unit, or whatever--and based on tentative identifications of that sort infer features of the computational structure. One could in this way hope to establish that the program of the system contains a stack, or does not contain a cue, and so on.[5]

To use a program as a description of a physical system it is unnecessary that every formal data structure that occurs in a formal execution of the formal program be interpreted as a property of a physical (or other) system. Like a formal theory, a program can be *partially* interpreted. The remainder, the uninterpreted part of the program, then states no hypotheses about the system; the uninterpreted fragment of a program simply imposes indirect constraints on the occurrence of physical properties and features designated by the interpreted fragment of the program. Instrumentalism is a special case.

Perhaps the most common sort of use of programs or procedures to characterize humans is *Ramsified*: No particular physical properties are identifed with data structures in the program, but the theorist claims that for each data structure, or each data structure of a certain kind, there *exists* a physical property instantiating that structure. Freud's conception of his psychology was of this sort, at least sometimes, and so is Chomsky's conception of theories of grammatical competence. One can imagine difficulties in carrying out this sort of conception, and yet it seems to be done more or less successfully in cognitive neuropsychology. In principle it is no more, and no less, difficult than doing physics with Ramsified theories.

---

[4] A detailed and useful illustration is given in the Appendix to Robert Cummins, *The Nature of Psychological Explanation.*

[5] Peter Spirtes points out that given an identification of "push" and "pop" one should be able to infer from behavior whether the system contains a stack, and likewise if one knows about memory limitations, one should be able to demonstrate from behavior that a system is not a cue.

2. Minimal Functionalism

The first functionalist account of psychological theory and mental states claims only that

1. a brain may instantiate a program

2. any physical state corresponding to an appropriate formal data structure in any instantiatation of a program instantiated by the human brain is an *instance* of a mental state

3. different physical properties may correspond to the same formal data structure in different instantiations of one and the same program

4. the property of being a physical instantiation of a given program, or of a specific data structure in a propram, is *not* a physical property.

This is the sort of functionalism that predominates in philosophical writing, and it does not require that programs describe programs. Most of the discussion of functionalism has focused on the second thesis, that mental states are functional states. Whether or not that thesis is true, it can only be true under a special requirement, which has to do with time. My instantiation of a functional state is whatever feature of me plays an appropriate causal role in producing my behavior. An essential feature of my behavior is the time it takes. Someone who does in a minute what I do in an hour does not do what I do; someone who does in an hour what I do in a minute does not do what I do. Equally, I think, an essential feature of the causal roles that events within me have is the time they require to produce their effects. If that is so, then the functionalist thesis must require that systems that instantiate the same program as I do have the same mental states as I do, provided the time intervals between corresponding instantaneous state descriptions in me, and in the other instantiation of my program, are the same.

The conception of psychological theory that accompanies minimal functionalism is roughly this: A theory is a complex predicate, whose internal structure is the set of hypotheses of the theory or the program; the theory or program *applies* to some systems, but not to others. A program applies to a system if the system instantiates the program. One may take exactly the same view about physical theories, and many writers have.[6] The substance to the functionalist claim is the fourth thesis above: the property of instantiating a program is not a *physical* property.

So, then, what is a physical property, and why is the property of instantiating a program, the property of satisfying a functional description, not a physical property? The conventional answer is variety: an unbounded variety of kinds of physical things can instantiate a functional description. The property of being something that likes the taste of

---

[6] Notably, P. Suppes, F. Suppe, B. Van Fraassen, and J. Sneed and his followers.

pineapple is supposed not to be a physical property but a functional property, because systems of an unbounded physical variety could, allegedly, instantiate a program that characterizes liking the taste of pineapple. What about the property of being a sphere? Systems of any physical composition can instantiate the property; we can have spheres of grass, spheres of mud, spheres of aluminum, spheres of mud and aluminum. There are no bounds. But we do ordinarily think of sphericity as a physical property. What about the property of satisfying the law of thermal expansion, according to which bodies increase in length in direct proportion to their increase in temperature? Nothing satisfies the law exactly; things of an unlimited variety of physical compositions satisfy it approximately. Does the law *not* describe a physical property, and do objects that have the same coefficient of thermal expansion not share a physical property? What about the property of being the source of the dampening force in a dampened harmonic oscillator? The physical variety of oscillator dampers is endless; they are determined not by their composition but by their causal role, but they do not seem the less physical for that. But now that we see the game, it is easy to continue. What of the property of producing plutonium by radioactive decay? Uranium 235 has the property, and for all I know so do some other nuclei. Is it not a physical property? Or consider the property of being an acid, a property originally characterized functionally--acids are those substances that are in solutions that produce a bitter taste, that tend to dissolve metals, and that react with bases to form salts. Bases and salts were equally characterized functionally, by their causal relations with acids.

I cannot imagine an adequate statement of our physical knowledge that does not make frequent use of functional locutions. If physical properties are the properties that enter into our body of physical knowledge, then properties given by causal relations are as physical as can be. And were we ever to possess a program that we could show is instantiated in someone and the instantiations of whose data structures cause behavior, we would soon come to regard it as a physical theory.

The fourth functionalist thesis looks quite false; one is tempted by it only by thinking of "physical properties" as properties of composition rather than of causal role. In fact, however, insofar as we have any clear conception of "physical properties" it includes an abundance of properties characterized by relationships, whether spatial or temporal or causal. But there is a further reason for thinking that minimal functionalism is really physicalism. The argument that functionally characterized features are not physical features depends entirely on the *variety* of physical instantiations a functional description, or program, may have. One and the same functional state, it is supposed, can be instantiated by the brain, or by silicon chips or by all of the people of China properly organized. But where functional characterizations of mental states are concerned, there is no reason to

believe that a variety of physically different but equally physical instantiations are possible.

There is some reason to think that human brains are the *only* realizations of the programs that human brains instantiate, and that the physical features of human brains that instantiate theoretical quantities (e.g., desiring to own a pineapple tree) are the only instantiations of such quantities. More, there is reason to think that in each of these cases human brains, or structures physically and chemically very much like them, are the *only possible* instantiations. We know of no other physical system that produces input/output behavior comparable to humans, let alone a system that produces it *in the same way*. Given the constraints that physics imposes, it seems reasonable to doubt that that human behavior could be reproduced by a system with causal components exactly analogous to the causal components responsible for human behavior unless that system had a chemical and physical composition essentially like that of humans. The reason most commonly given to the contrary is the Argument from Imagination. It goes as follows:

1. We can imagine a system that has a physical composition completely different from humans; say a system made of silicon. And we can further imagine that such a system has the same behavioral dispositions as a human and instantiates the same functional description as does a human.

2. Whatever is imaginable is possible.

3. Therefore it is possible for a functional description of humans to be instantiated by systems with an entirely different physical composition.

Despite its ubiquity in philosophical argument, the second premise is false. I can imagine traveling faster than the velocity of light, but I cannot travel faster than the velocity of light, and it is not possible for me to travel faster than the velocity of light. Imagining a chicken with lips does not show that one is possible, only that it does not violate our language to speak of them. What language permits is one thing, what is possible is something else.

Another argument for the variety of possible instantiations of functional descriptions of humans not only imagines a bizarre instantiation, but gives rough directions for constructing one. Ned Block imagines we have converted the government of China to functionalism, and get a billion people each with two way radios to simulate the neurons of someone's brain.[7] Imaginings like this forget that *time is an essential part of the input/output behavior and of the intermediate steps in a correct functional description of humans.* The Chinese, industrious as they are, just couldn't communicate with the speed that neurons do. Carnap maintained that if the full complexity an objects relationships were represented in a formal

---

[7] See N. Block, "Troubles with Functionalism," *op. cit.*.

theory, objects could be individuated entirely by the syntactic structure of the complete theory true of them.[8] Whether Carnap is right or not, we do expect that the more complex a set of relationships, the smaller the variety of physical realizations we can expect it to have. The relationships that produce human behavior are complex indeed.

3. Cognitive Psychology and the Program of the Mind

A key functionalist idea is that there is a programming system appropriate to *minds*; different minds can execute the same program in this system, but they will do so in different physical ways: different physical properties will correspond to the same data structures. Equally, one and the same mind will execute its program in different physical ways on different occasions. Putnam suggests that the appropriate programming system is probabalistic automata, and that instances of mental states are instances of machine states of such automata. Presumably, other functionalists need not be committed to this particular hypothesis about the programming system of minds.[9] Whatever the programming language for minds, they may hold, there are formal objects analogous to the data structures of formal Turing machines in any sequence of formal objects that we call the "execution" of a formal program. These functional states are abstract objects, like numbers or sets or propositions. They are features of programs. Mental states are functional states, and they are instantiated in a brain when the brain executes the program of which they are a feature.[10]

Functionalists who think that programs describe the mind reject any identification of program data structures or computational states with physical states; they reject even the existence of physical states that could satisfy Ramsified programs. In their view, programs can only describe the mind by describing the minds' own program: one program must describe another, and nothing is fixed between the two programs except for input/output behavior.

---

[8] See R. Carnap, *The Logical Structure of the World*

[9] In "What Psychological States Are Not," *Philosophical Review* 81 (1972), Block and Fodor argue that psychological states cannot, for various reasons, be states of a Turing machine table. Their most forceful arguments (that one can have several pscyhological states or features at once, that psychological states can have other states as components, and that pscyhological states are creative or potentially infinite whereas the machine table of a Turing machine is a finite automaton) are unavailing if psychological states are identified, not with the states of the machine table, but with data structures that can be generated by the machine.

[10] This "computational functionalism," suggested by many writers, should be distinguished from other doctrines that go by similar titles. Cummins' "functional analysis," which is simply the resolution of capacities into subcapacities, does not have the same implications. Neither does the view, implicit in Hartry Field's "Mental Representation" that functional descriptions are something like Ramsified descriptions that refer to unidentified physical states.

In this case the relation is not between a formal object, a program, and a concrete object, but instead purportedly between two programs in two distinct programming systems; let us call the latter relation, whatever it is, "describing" in contrast with the semantic relation of "instantiating" that can hold between a program and a particular system. The "describing" relation is usually glossed either by analogy to the relation between a program in a high level language and the machine language program into which it is compiled, or by appeal to a notion of "levels" of description. Thus consider the following passage from an influential textbook in cognitive psychology:

> Computers offer an interesting analogy to help us understand the need for an abstract analysis. Like the brain, a c;omputer consists of millions of components. For any interesting computer task--for example, solving a problem in mathematics such as integration--trying to understand the overall behavior of the machine by studying the behavior of each of its physical components is hopeless. However, high-level programming languages exist for specifying the behavior of the computer. The computer has an *interpreter* for converting each statement in the higher level language into a large number of low-level statements that specify what the physical components of the computer should do...A person can ofIten obtain a good understanding of the behavior of the computer by studying the high level computer program. A cognitive theory should be like a computer program. That is, it should be a precise specification of the behavior but offerd in term sufficiently abstract to provide a conceptually tractable framework for understanding the phenomenon.[11]

There is an innocuous way to read this passage: people should be described by macroscopic properties that reduce to microscopic physical properties, because a description of their behavior in terms of the microscopic properties that produce it would be incomprehensible. But there is a more literal reading of the analogy: The psychologist's program is to the program of the brain as a program in LISP is to the interpreted program running on a VAX; one is a description of the same abstract system as the other but "at a higher level of abstraction." Gripping it may be, but taken literally, the analogy is also fundamentally nonsensical.

Let the language of thought in which the program of the mind is expressed be BRAIN, and suppose someone, the agent, in fact executes a program in BRAIN, call it AGENT. The analogy with the digital computer proposes that the aim of the cognitive psychologist is to write a program in say, LISP, that describes the AGENT program, or at least describes features of AGENT in addition to the function AGENT computes. Let us grant a matching of input and output; LISP structures are mapped to physical or behavioral features that are the input and output to the agent's instantiation of the AGENT program. Suppose further that the psychologist's LISP program perfectly describes, under this convention, the input/output

---

[11] J. Anderson, *Cognitive Psychology, p. 10*

function of the agent. The question is what more there could be to *describing* AGENT in LISP than this input/output equivalence.

Describing could not mean what it means for the psychologist's LISP program to describe the machine language program into which it is interpreted or compiled on a physical computer. For that relationship presupposes a specific compiler or interpreter that takes LISP into machine language. There is no compiler or translator that takes LISP programs into BRAIN programs, or BRAIN programs into LISP programs. We don't know anything about the programming system BRAIN, and we certainly don't know *any* features of a translation between BRAIN and LISP that would suffice for a definite description of the translation function. We lack the knowledge even to *refer to* a particular translation between BRAIN and LISP.

Describing could not mean that there *exists* some effective translation that takes BRAIN into LISP (or vice versa) and under that translation AGENT is equivalent to the psychologist's program. It could not mean that because provided two programs are input/ouput equivalent, there is always *some* effective translation under which the programs are mapped into one another: A programming language is an acceptable numbering of the recursive functions; a program in such a language is an index. Any effective map that preserves input/output and maps the index of AGENT to the index of the psychologist's program is such a translation.

A more detailed proposal to answer the question turns on the idea that one program describes a feature of another if, for every input, there is a one to one mapping in the instantaneous descriptions that result from the formal executions of the first program into those of the second. (Many variations of the proposal are evidently possible.) Thee proposal seems unsatisfactory on several counts. The first is that it fails to recognize the kind of descriptions that psychologists think they can make of the program of the brain. For example, that a particular capacity is (or is not) effected by parallel processing. Second, it requires that the processing be discrete and synchronic; the computations are assumed to take place in discrete intervals of time. But many of the most popular computational models are asynchronous parallel processes, and the processing does not take place one discrete step per discrete unit of time. Third, even supposing that processing does proceed discretely, and taking the simple deterministic case, the condition is still unintuitively lenient. In that case it says only that the target program, P', must have at least as many intervening steps between input and output as does the program P which is supposed to describe P'. For the same reason it would not help to require in the deterministic case that their be a recursive function taking inputs and data structures arising from states of computation with P to inputs and data structures arisings from similarly ordered states of computation with P'. Such a condition would again only guarantee that for

any input P' has at least as many computational steps as P. So let us that add, subject to some hesitation, that programs in different systems that are input/output equivalent can also be compared for *computational complexity*. We are still a long way from what what cognitive psychologists seem to think they can do in describing the program of the brain.

It begins to look hard to say in general what it could mean for one program to describe another. Anderson and others talk of levels of abstraction, and one might think that helps somehow. What *could* it mean to say that the psychologist's LISP program describes the agent at a "higher level of abstraction" than does the AGENT program? If it means that primitive terms in LISP are translated into complex procedures in BRAIN, we are back to the previous cases. There is no translation that can be specified to make the claim true or false: it is true under some translations false under others, and there is no means of independently specifying any particular translation. If it means that the primitive terms of LISP denote complex rather than simple physical properties, and so construed the brain executes the LISP program, then the claim is a disguised claim about instantiation, not a claim about description. So understood, the cognitive psychologist does not construe the theory as making claims about abstract functional states or relations; the theory is understood as a kind of Ramsified claim about physical properties of the brain.

In defense of the usual sort of cognitive psychology one might propose something like this: If we knew the programming language BRAIN and we knew the program AGENT, we would on inspection *just see* that AGENT and the psychologist's program compute the input/output function *in the same way*. And one might add that if such judgements were impossible there would be no point to proofs of the properties of algorithms in computer science.

There are volumes and volumes of algorithms, and it is tempting to think that for any algorithm there is a property of *programs* in all programming systems. One may think: for any program and any algorithm, either the program embodies the algorithm or it doesn't, and there is therefore for each algorithm a property, other than input/output behavior, by which any two programs in whatever languages may be compared, and there is therefore a respect with which a program in one language may describe a program in another language. There are, one may also think, invariant features, such as space or time complexity, that programs must share if they are implementations of one and the same algorithm.

On examination, the story seems dubious. Algorithms are always given using some representation of the function to be computed, and the representation goes beyond the representation of the input and output. When one begins to alter the representation, intuitions about whether or not one has the same algorithm come apart.

Consider the simple issue of a matrix representation. When does a Turing machine contain a representation of a matrix? One can imagine the rows of a matrix entered sequentially on the machine tape. Does it matter if computationally relevant stuff intervenes between the rows? Does it matter whether the entires are in unary or binary notation? Does it matter if computationally relevant stuff intervenes within the rows? Does it matter if the first element of the nth row precedes the last element of the n−1th row?[12]

If intuitions about sameness of algorithms and representations begin to unravel over these few variations, one cannot plausibly claim that there are intuitions that spread over the whole gamut of programming systems. And if there are no intuitions, and there is no mathematical structure that determines program equivalence between different programming systems, then there is nothing for the property of algorithmic sameness to be. Nor can it plausibly be said that the unraveling of intuitions only shows that the properties of algorithmic equivalence are *vague*; vagueness requires that there be an unequivocal normal use that covers most cases. Algorithmic sameness is not even vague.

Does the argument prove too much, namely that the practice of computer science is pointless? When we use an algorithm in programming we take advantage of a program sketch in some representation, for that is what we find in books of algorithms. The program sketch has a psychological effect; it causes us to think of how to develop a program in the language we are using. We may say that we *intend* to implement the algorithm in the language we are using, and there is a sort of fact to the matter merely in virtue of our intent. Given particular component functions, *interpreted* as "pop" and "push" there may be a fact of the matter as to whether or not a composition of those functions is a "stack." But that identification depends upon something more than the syntactic properties of the functions interpreted as push and pop; it depends on the *interpretation*. When we verify a property of an algorithm and implement that algorithm in code, the programmer establishes an interpretation of algorithm structures in code structures. The programmer acts as a kind of informal compiler, and judges that under that compilation proofs about the complexity of the algorithm will go through for the coded program. The programmer/compiler is just what is missing in attempts to describe programs in BRAIN by programs in LISP.

Clearly there are special cases in which programs describe programs, or clearly fail in that description. When there is a physical instantiation, for example, properties of that physical system may establish or preclude description as certain kinds of computational systems. If one programming system is an isomorphic variant of another, and the programs are

---

[12] I owe this example to Kevin Kelly.

transforms under the alphabetic substitution, we have no problem saying that one describes the other. If two programs are in the same programming language, and every primitive syntactic element of another is a composition of syntactic elements of the other, and every data structure generated by the first for any input is reducible to a data structure generated by the other for the same input, we have no difficulty supposing that one program describes features of the other. None of these cases pertain for functionalist cognitive psychology. For them there is no fact to the matter in general about whether two programs in different systems are or are not the same. Programs do not describe other programs. The psychologists' programs cannot describe the programs of the brain, for there is no relation of describing. Ultimately, cognitive psychology is either physics or behaviorism; given the choice, it had better be physics.